A Deep Learning Approach for Forecasting Construction Project Duration at Completion

by

Cristhian Felix Laura Portugal

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Construction Engineering and Management

Department of Civil and Environmental Engineering

University of Alberta

Abstract

Accurate forecasting of project duration is crucial during the execution phase as it affects its overall performance, timely decision-making, identification of potential delays, and resource allocation. This research proposes a proof of concept based on artificial intelligence, specifically using deep learning algorithms, demonstrating its potential application. These algorithms have shown remarkable results in finding patterns in large amounts of data and making accurate predictions. Moreover, the dataset provided to the model is treated as a time series, capturing the sequential nature of data collected throughout the execution phase. Additionally, predictions are yielded at work package level, providing to project managers granular information to make decisions.

The study follows these steps: (1) a comprehensive literature review was conducted to explore the latest advancements on related topics and underline current gaps. (2) A data acquisition model was elaborated founded on a consistent selection of duration-influencing factors. Then, actual data was collected and profiled from multiple projects using their work package names as links, thus creating datasets per work package. (3) The forecasting duration at completion model was developed, including data preprocessing and the computational deep-learning-based modelling per work package. After that, the overall project duration was modelled using the Critical Path Method and Precedence Diagramming Method and set in the Graphical User Interface. (4) The developed forecasting model was used to comparing three well-suited deep-learning algorithms with actual project data and consequently, selecting the most accurate. Next, the selected algorithm was incorporated into the Graphical User Interface. The study also was validated by comparing the proposed model with traditional methods, including the Earned Value Methodology (EVM) and Earned Schedule Methodology (ESM). Finally, the resultant model was verified through sensitivity analysis.

As a result, the forecasting model based on the Long Short-Term Memory (LSTM) algorithm demonstrated the best performance against Multi-Layer Perceptron and Convolutional Neural Network algorithms. Likewise, it performed better than intensive-used forecasting methods in the industry, such as Earned Value Methodology (EVM) and Earned Schedule Methodology (ESM). These promising results contributes to the foundation of Artificial Intelligence (AI) applicability in construction project duration at completion forecasting.

Preface

This thesis is an original work by Cristhian Felix Laura Portugal. No part of this thesis has been previously published.

Dedication

With all my love for my soon-to-arrive baby, my beloved daughter Miranda, my wonderful wife Denisse. Also, my family in Peru, my cherished parents, Nancy Portugal and Felix Laura. This work is dedicated to all of you.

Acknowledgments

First and foremost, to God for His spiritual support and strength throughout this journey.

I would like to express my thanks to my supervisor, Dr. Ahmed Hammad, for his invaluable support and encouragement during this research. His expertise and insightful feedback have been instrumental in the completion of this thesis. My sincere thanks go to the committee members, Dr. Farook Hamzeh and Dr. Ali Imanpour, for their constructive feedback and suggestions during the thesis defense, which contributed the quality of this work. In the same way, to Dr. Gaang Lee, who kindly accepted be part of this meeting as the committee chair.

I am grateful to my wife, Denisse, whose expertise in project management provided critical feedback and in turn, unwavering support. Her encouragement was pivotal to my success and kept me motivated throughout this journey. To my mother, Nancy Portugal, for her eternal support and love, and to my father, Felix Laura, for his always accurate advice, I am deeply thankful. Additionally, to Alex Laura, Tanya Belleau and Dennis Laura, their encouragement has been essential for this work.

I also want to extend my heartfelt thanks to my friends and colleagues for their support and understanding, and to everyone who contributed to this work in various ways. Your contributions have been deeply appreciated.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Background

The construction sector significantly contributes to the global economy, promoting infrastructure development, job creation, and economic growth. According to Oxford Economics, global construction spending is expected to increase from $9.7 trillion in 2022 to $13.9 trillion by 2037. (Fearnley et al., 2023). Similarly, Canada's construction industry is also essential. It supports multiple construction types, such as residential, commercial, infrastructure, and industrial, and contributed to about 7.5% of Canada's GDP in 2023 (Statistics Canada, 2024).

Regarding the construction project lifecycle, the construction execution phase is unique and critical. It transforms project plans into physical structures, fulfilling utility requirements, resource management, subcontractors coordination, and adhesion to construction codes and regulations (Oberlender & Spencer, 2022). This phase involves multiple on-site activities such as excavation, foundation, framing, electrical, finishing works, among others related. Moreover, most of the project's budget is spent in this phase (as shown in Figure 1.1) and potential delays can significantly impact the overall project timeline (Chhotelal et al., 2023; Ajayi & Chinda, 2022; Shahsavand et al., 2018).

Therefore, accurate project outcome forecasts become crucial. This research is focused on the project duration at completion forecasting, which helps to fit resource allocation, timely procurement, and coordination of various activities. It also enables construction organizations to set realistic expectations for clients, stakeholders, and project teams. In addition, timely project completion is essential for maintaining client satisfaction and enhancing the reputation of construction organizations.

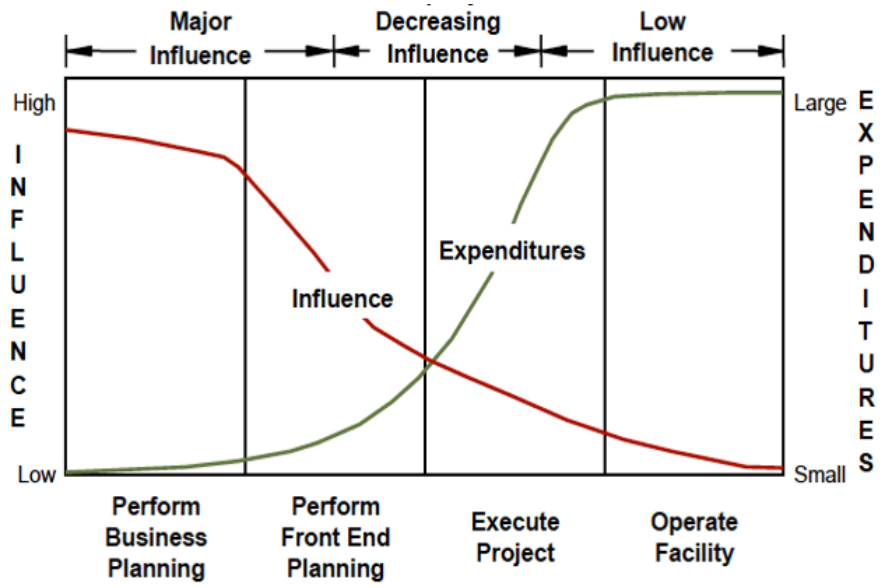


Figure 1.1 Influence and Expenditures Curve (MacLeamy curve) for the Project Life Cycle

On the other hand, Artificial Intelligence (AI) has emerged as a promising solution, revolutionizing the approach of data analysis, decision-making, and innovation (Duan et al., 2019). It can handle large quantities of data to deliver valuable data-driven outcomes, such vast amounts of data as generated by the construction industry. Thus, many construction firms have started adopting AI in their processes, as was reported by KPMG in 2023, finding that 40% have implemented AI, mostly in the early stages (Armstrong et al., 2023). The primary explored fields with AI have been worker safety, productivity improvements, and quality assurance, conversely, forecasting applications in project control still in their beginnings. Accordingly, the application of AI in project management tasks like forecasting constitutes a significant gap nowadays.

1.2 Problem Identification

Researchers have addressed project duration at completion forecasting for decades, proposing several methodologies, many of them even automated with sophisticated software to ensure accurate outcomes. Unfortunately, weaknesses have been identified during their implementations.

For instance, the Earned Value Methodology (EVM) ideally assumes that current performance still during remaining works (Chou et al., 2010; Grandage, 2022; Vanhoucke, 2012); Probabilistic methods require uncertainty inputs from experts, which are prone to subjectivity or bias (Durbach et al., 2017; Gneiting & Katzfuss, 2014; X. Yue et al., 2018); and also most of these methodologies present non-timely forecasts calculations due to their complex processes leading to delays in decision making (Ahiaga-Dagbui & Smith, 2014; Loshin, 2011).

Poor data management has been evidenced as another issue regarding Machine Learning applications. Large amounts of data are yielded in the construction sector while monitoring progress, which should be leveraged to obtain beneficial outcomes using Machine Learning (Tanga et al., 2022). To take advantage of that, it is crucial a proper data collection which enables data quality and consistency for optimal Machine Learning application performance. However, research agrees that construction companies do not have a standardized data flow (Bobrova, 2023; Matti & Antti, 2020; Pavlova et al., 2021), varying between projects and misusing valuable, reliable forecasting input data.

Similarly, another drawback identified is how machine learning has been used in addressing construction project duration forecasting. Its application has been limited to predicting the overall project level without exploring more granular levels, such as at the work package level. Additionally, most machine-learning-based applications consider solely non-time-dependent input variables, disregarding time-dependent variables like those generated during the construction project's tracking.

Accordingly, inaccurate predictions cause inefficient resource allocation or improper risk management for the remaining work. Moreover, because of time overruns, companies must incur unexpected indirect costs and expend undesirable liquidated damages. Given that, providing

accurate project duration forecasts at the work package level supported by adequate data collection and deep learning algorithms will be proposed.

1.3 Research Objectives

This research aims to enhance the accuracy of duration-at-completion forecasting for construction projects since the work package level during the execution phase, processing historical data with a deep-learning model. In fulfillment of the research objective, the present study will focus on the following goals:

- Analyze and propose a structured data collection for project duration predictions.
- Evaluate and select deep learning algorithms for project duration forecasting that can address the regression perspective and handling time series datasets.
- To integrate individual work package predictions from deep learning to obtain the overall project prediction, through a User Interface environment.

1.4 Research Methodology

The research methodology which involves a proof of concept to address the problem identified and achieve the research objectives is divided into four phases as shown in Figure 1.2 and explained below.

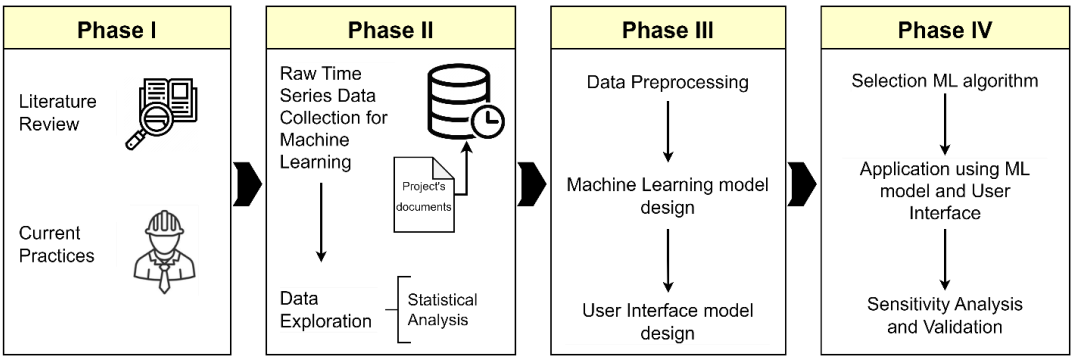


Figure 1.2 Research Methodology

In phase one, an extensive literature review on five topics about construction project duration forecasting is developed. These topics include understanding project construction in the execution phase, exploring the latest methods when managing project schedules, reviewing advancements in forecasting methods and studying machine learning developments applied to construction projects for project duration forecasting. On the other hand, this phase involves understanding industry practitioners' current practices, including project duration forecasting since work package levels. It enables the elucidation of advantages and disadvantages from an industry perspective.

The second phase encompasses a structured data acquisition model, considering the outcomes of phase one. In this phase, the input and output modelling variables are defined by an exploratory analysis of the raw data collected. As data characteristics, it should be numerical and sequential over time. The former is because this is a regression problem and the neural network application (Turban et al., 2011). The latter is related to time series requirements such as seasonality or trends. The third phase spans three stages: data preprocessing, deep learning model development and the user interface model creation. The data preprocessing handles raw data by data cleaning, transforming, feature selection, splitting and normalizing. After that, the deep learning development presents architectural aspects of its design, such as hyperparameters and performance metrics. Later, the user interface design is powered by the deep learning model for calculations and adds the methodology to integrate individual work package predictions into overall project forecasting.

An application of the User Interface (UI) integrated with the deep learning model is presented in phase 4. It sets optimal hyperparameters and selects a deep learning algorithm among the three proposed. The algorithm chosen from the third phase is incorporated into the UI, which is designed to be user-friendly for non-expert users. The interface is intuitive and sequentially logical, making

navigating easier until it produces work package and overall project prediction reports. Next, a sensitivity analysis (what-if) over input variables is deployed.

1.5 Thesis Organization

The thesis was organized as follows:

- Chapter one of the thesis provides a background description, problem identification, research objectives, research methodology, academic and industry expected contributions, and the thesis organization.
- Chapter two of the thesis presents a detailed literature review on every topic related to project duration forecasting in construction during the execution phase.
- Chapter three of the thesis develops the conceptual model and considers a process flow to propose a suitable data acquisition mode. It involves selecting proper duration-influencing factors, ERD, and analysis of available data.
- Chapter four of the thesis details the model development using machine learning, as well as the elaboration of the user interface.
- Chapter five of the thesis depicts the outcomes using the model created and passing through the selection of the best machine learning algorithm. This chapter also demonstrates the step-by-step user interface with project data.
- Finally, chapter six of the thesis comprises a research summary, limitations of the research work, and recommendations for future investigations.

Chapter 2 Literature Review

2.1 Introduction

To attain the research objectives for this study, the initial and crucial step involves establishing a comprehensive perspective through an exhaustive literature review. These objectives lie in complex construction project management fields and the application of state-of-the-art technology such as Artificial Intelligence (AI). In this context, the investigation draws support from the following four specialization fields: Firstly, this chapter explains schedule management in construction projects. Secondly, scheduling techniques are described. Thirdly, an exploration of forecasting techniques is presented. Fourth, a brief overview of machine learning (ML) techniques, data clustering, regression and classification tasks, types of machine learning, a revision of Artificial Neural networks, representative Deep Learning algorithms, and current applications on Construction are studied. Lastly, a summary of previous research and the research gap was explained.

Each previously described field drives to expand the knowledge of these matters, understanding their synergy and elucidating their contribution to the present research. By doing so, this chapter will expose the fundamentals behind this study.

2.2 Schedule Management in Construction Projects

Many studies agree that schedule management is a meaningful component of construction project management (Faghihi et al., 2014; Meng et al., 2022; Yu et al., 2021). It is defined as those processes needed to complete the project on time. These processes are planning schedule management, activities definition, activities sequencing configuration, activities duration estimation, schedule development, and schedule control (Project Management Institute, 2017). By

executing these processes, one of the important outcomes is the *Schedule Model*, which results from applying project scheduling tasks during the Schedule Development Process. This task involves tools, techniques, and the project team's experience (Project Management Institute, 2019). The schedule model is then controlled and monitored under preset conditions established in the schedule project plan.

Unlike the PMI, for the Association for the Advancement of Cost Engineering International (AACEi), schedule management includes only three phases: planning, developing, and controlling project schedules (Stephenson, 2015). By matching both perspectives about schedule management processes, AACEi considers that defining, sequencing, and estimating activities are subprocesses of the Planning Project Schedule. It also adds that Controlling Project Schedule encompasses measuring, evaluating performance, *forecasting*, and initializing the change management process whether the project requires it. From both frameworks, the forecasting task is placed within the controlling schedule process in the overall project schedule management, as shown in figure 2.1. As can be seen, the forecasting task outcomes will depend on the scheduling technique adopted because different techniques consider diverse factors and limitations that can impact forecasting management. The following studies the scheduling techniques used in the construction industry.

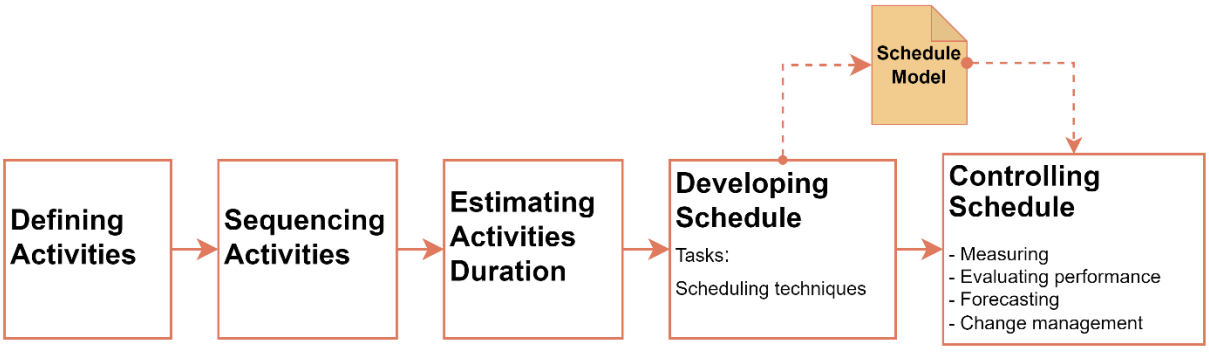


Figure 2.1 Schedule Management Processes (Project Management Institute, 2019).

2.3 Scheduling Techniques: Network-based, Constraint-based, Line of Balance, Pull-Driven.

The schedule model encompasses three critical aspects: defining the scheduling approach to adopt, selecting suitable scheduling tools, and considering wide-ranging project information (Project Management Institute, 2019). The four most used scheduling approaches have been studied, as shown in the Figure below.

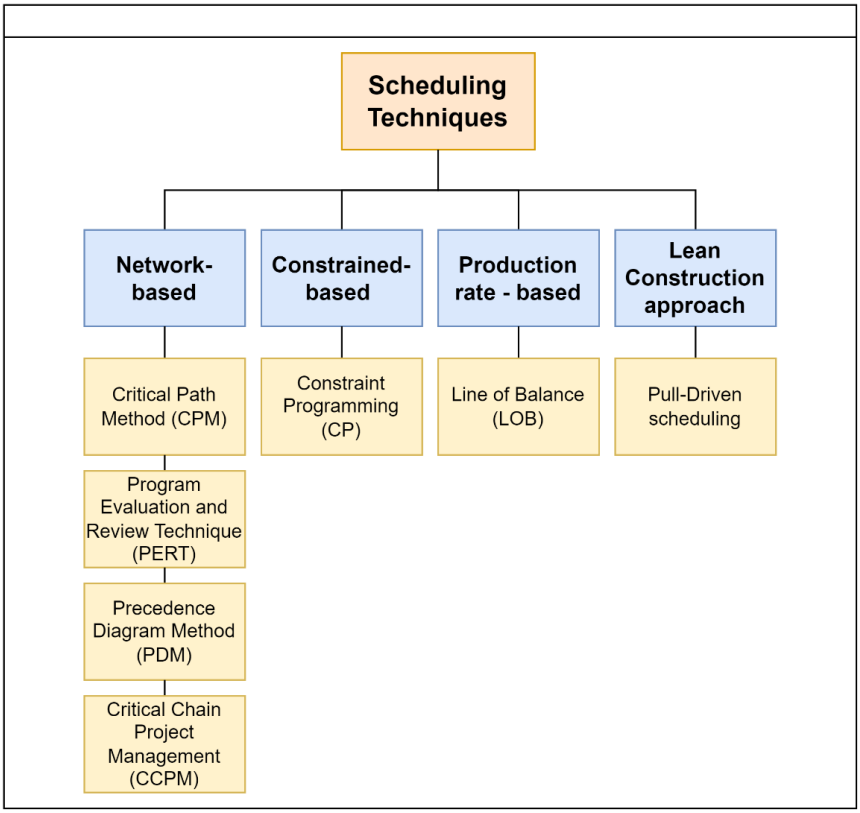


Figure 2.2 Most Used Scheduling Techniques encompassed on this Research

1. Network-based scheduling methods.

Network-based scheduling is arguably the most-known approach used by the construction industry. It was raised in the 1950s in response to the limitations of bar chart scheduling techniques (Baldwin & Bordoli, 2014). This method is characterized by using a graph that portrays nodes

interconnected logically by arrows, each oriented to a specific direction (Hajdu, 1997). In the construction industry, several approaches have been widely applied and fall under this category, including the Critical Path Method (CPM), the Program Evaluation and Review Technique (PERT), the Precedence Diagram Method (PDM) and the Critical Chain Project Management (CCPM). Below, these methods are delved into detail.

- **Critical Path Method (CPM)**

Hajdu (1997) stated that this method is arguably the earliest method that came with innovative techniques shifting from traditional non-network techniques to a network approach. In 1959, Kelley and Walker introduced this method as part of the research conducted by DuPont Company ("The Origins of Schedule Management," 2018). Initially, this method consisted of three rules by drawing the network (Hajdu, 1997). First, it considered one starting event (node) and one terminal event (node) only, represented by 's' and 't,' respectively. Second, the loops were not part of the network; otherwise, it would imply a returning path to the origin node, or even successor nodes would condition to the predecessor nodes. Third, the network did not consider multiple activities (represented by arrows) arriving in the same node conversely in most real-life situations; however, this could be overcome by imputing fictitious nodes or 'dummies.' Today, the CPM algorithm is well-known for the forward and backward pass calculations. When making a forward pass, it computes the earliest start and finish dates, and making a backward pass calculates the latest start and finish dates. This logic is the CPM's potential to discover the critical path (Lu, 2020).

- **Program Evaluation and Review Technique (PERT)**

The U.S. Navy Special Projects Office introduced the PERT in 1958 when addressing complex projects (Baldwin & Bordoli, 2014). For Kerzner (2017), this period marked the "age of massive engineering." In the beginning, the PERT followed the next steps (PERT, 1958)

a. Defining coherent events aligned with specific objectives at the core of planned progress.

b. Sequentially arranging these events to establish logical relationships among them as they unfold.

c. Providing an initial estimate of each activity's duration while gauging the variability by comparing the associated events.

d. Leveraging computational tools to process and effectively manage this data.

e. Creating a structured, systematic communication framework to capture progress and facilitate data updates.

This technique has spurred the development and adoption of other scheduling methods, such as the Precedence Diagram and Critical Chain Method.

- **Precedence Diagram Method (PDM)**

The Precedence Diagram Method visually shows project activities through nodes, delivering essential information for each task. Baldwin et al. (2014) outlined five critical considerations for creating a PDM:

a. *Sequential Time Flow:* Ensure that time flows from left to right in the diagram.

b. *Flow Direction:* Indicate the flow direction using arrowheads.

c. *Arrow Length*: Pay attention to the relative length of arrows within the graph.

d. *Arrow Orientation*: The orientation of arrows should be unambiguous.

e. *Activity Descriptions:* Each node should comprehensively describe the associated activity.

Kerzner (2017) also referred to this method as Activity-on-Node (AON). In PDM, activity relationships and constraints are represented by arrows in the graph. Activities encompass slacks, while lags are calculated between activities. Kerzner emphasized the use of leads, particularly when addressing resource constraints. Lu (2020) noted that PDM, a variation of the Critical Path

Method (CPM), features intelligent relationships. In PDM, non-Finish-to-Start (non-FS) relationships may include restrictions related to resources or technologies. Currently, the PDM is a widely adopted scheduling technique and is commonly implemented in various project management software tools. PDM is sometimes mistaken for CPM in practice due to their similarities (Project Management Institute, 2019).

- **Critical Chain method**

This method centers on activities and resource interaction (Project Management Institute, 2019). In its application, two crucial factors are considered: firstly, an analysis of the original critical path and, secondly, a thorough evaluation of resource availability for the successful completion of project activities (Baldwin & Bordoli, 2014). After completing the tasks, the subsequent phase involves the reduction of the estimated durations for activities, as highlighted by Raz et al. (2003):

a. *Inherent Uncertainties:* Recognizing that uncertainties are inherent in all activities.
b. *Common Overestimation:* Acknowledging the tendency for overestimation in the duration of activities.
c. *Safety Time Considerations:* Understanding that activities typically incorporate safety time, leading to constrained resources for subsequent activities related to the original one. In practice, the activity owner relies on the planned time due to these restrictions.

In this context, the Project Management Institute (2019) emphasizes that the critical chain is identified as the longest resource-leveled path, considering the presence of buffers.

While Critical Chain Project Management (CCPM) offers notable advantages, as articulated by Leach (1999): Ensuring timely delivery, averting scope creep, and adhering to budgets with diligent application, a significant drawback lies in the lack of clarity regarding buffer dimensions.

This concern is underscored by Baldwin et al. (2014), who also note the method's reluctance to update the project baseline, a practice viewed as essential by some project managers.

- **Comparison among the Network-based Scheduling approaches**

Table 2.1 provides an overview of the characteristics of each method across various criteria, as discussed earlier:

Table 2.1 Comparison of the Described Network-based Scheduling Approaches

Criteria	Critical Path Method (CPM)	Program Evaluation and Review Technique (PERT)	Precedence Diagram Method (PDM)	Critical Chain Method
Nature of Network Representation	Node and Arrow Diagram	Node and Arrow Diagram	Node and Arrow Diagram	Node and Arrow Diagram
Activity Dependency Representation	Finish-to-Start (FS)	Finish-to-Start (FS)	Various dependencies allowed	Finish-to-Start (FS)
Focus on Resource Constraints	Limited emphasis	Limited emphasis	Limited emphasis	Central emphasis
Activity Duration Estimation	Single duration estimate	Three-point estimate (optimistic, pessimistic, most likely)	Single duration estimate	Single duration estimates with buffer
Uncertainty Consideration	Limited	High	Limited	Moderate
Buffer Utilization	No buffers	Incorporates buffers	No buffers	Buffers are a vital element
Critical Path Definition	Longest path in terms of time	Probabilistic critical path	Longest path in terms of time	Longest resource-leveled path
Management of Resource Constraints	Limited focus	Limited focus	Limited focus	Central focus, buffers manage constraints
Handling Project Changes	Rigid	More adaptable	Adaptable	Adaptable with buffer management
Commonly Used in Practice	Yes	Yes	Yes	Increasing adoption

2. Constrained-based scheduling based on Constraint Programming (CP)

Unlike the Network-based Scheduling approaches, Constrained-based Scheduling utilizes Constraint Programming (CP) within the Operations Research (OR) domain to tackle scheduling challenges. Moreover, it employs efficient propagation algorithms to enhance model performance (Baptiste et al., 2006). By concurring rigid activity durations, interdependencies, and construction's dynamic and uncertain nature, this approach introduces flexibility to the model. Activities are treated as variables with defined relationships, allowing for adding or eliminating constraints as needed (Zupančič et al., 2007).

In constraint programming, it is classified as constraint satisfaction, where the activities' requirements become constraints in the model (I.-C. Wu et al., 2010). These constraints are categorized into hard and soft constraints, with technological dependencies and resource availability as attributes. Next, the goal is to establish an objective function that satisfies all project restrictions. This model includes activities, resource constraints, temporal constraints, an objective function, and extensions to the basic model. It enables addressing typical construction sector scenarios, such as resource availability, variable times and costs, breakable activities, and activities left undone due to resources (Baptiste et al., 2006).

Supporting this scheduling method, Fromherz (2001) underscores that scheduling is both a constraint satisfaction problem and an optimization problem. While scheduling problems are considered NP-hard, complexity can be mitigated with specific considerations such as pre-empting tasks in the programming model. These kinds of conditions boost the CP method's efficiency and benefits in scheduling (Müller et al., 2022). They also suggest that machine learning can predict the most suitable CP solver based on each unique scenario, considering factors like activity relationships and resource availability.

Lorterapog and Ussavadilokrit (2013) suggest using constraint programming in construction projects because it offers greater flexibility in representing constraints and makes it more accessible to manage project networks than traditional CPM. The authors recommend prioritizing alternative schedules, incorporating search algorithms, and adopting constraint relaxation and collaborative scheduling for better efficiency.

3. Line of balance (LOB) scheduling

Line of Balance (LOB) is a prominent method within the family of Linear Scheduling methods (LSMs) (Ammar, 2020). The Goodyear Tire and Rubber Company initially proposed the technique, which the U.S. Navy later adopted during the Second World War and the Korean War to coordinate mass production strategies (Frandson et al., 2015). The LOB scheduling technique is well-suited for applying the learning effect due to the repetitive nature of its activities. When the learning effect is applied to the LOB schedule, productivity rates improve as the project progresses. This improvement leads to a decrease in the duration of activities. As a result, the output schedule will change from inclined parallel bars to curves (Matey et al., 2017; Zahran et al., 2016). Initially, it is represented linearly, incorporating production rates and available resources, and focusing on sequencing activities. The results display project deliverables' completion times and a production schedule of significant sub-elements (Baldwin & Bordoli, 2014). Notable projects suited for LOB include highways, high-rise buildings, pipelines, and tunnels (Bayhan et al., 2020).

In his research, Ammar (2020) emphasizes the importance of considering crews when implementing the Line of Balance (LOB) scheduling method. He suggests incorporating work interruptions to simulate real-life conditions and optimizing the LOB model to improve its

performance. Ammar also acknowledges that recurrent activities typically make up a significant portion of the project construction process.

The Line of Balance (LOB) method offers a graphical representation of the correlation between productivity and the time it takes to complete project activities. Also, the combination of CPM and LOB can deal with repetitive work, as demonstrated by Hegazy and Mostafa (2021). Conversely, achieving a "natural rhythm" is a significant challenge, as suggested by the LOB method (Tang et al., 2018), requiring a high commitment from construction practitioners. Ammar (2020) added that determining the optimal crew per activity to achieve the ideal project duration under typical project constraints would become another challenge by implementing this method.

4. Lean Construction approach: Pull-Driven Scheduling

From Lean Construction, employing "pull-driven" scheduling emerged as a prominent technique which lies in achieving optimal outcomes, considering factors such as quality, time, cost, and client demands (Tommelein, 1998). This approach advocates for the intensive and strategic use of resources as inputs, aiming to minimize wait times in queues and strategically selecting activities (processes) to obtain products needed further in the process, enhancing system fluency. Concerning the implementation of pull-driven scheduling, Tommelein (1998) highlights the need to reinforce a selective control process on resources for assignment to any activity. In this vein, Fukushima (2000) identifies three crucial factors for implementing pull-system scheduling: solving limited space problems, reducing inventory, and embracing change based on agility.

Unlike the push-system utilized in Critical Path Method (CPM), the pull-system prioritizes a continuous workflow (I.-T. Yang & Ioannou, 2001). Ghanem et al. (2022) pointed out that production planning, crew-level assignments, and decisions on resource mobilization are remarkable differences between push and pull systems. Firstly, in production planning, the push-

system aims to reduce project duration based on Critical Path Method techniques, while the pull-system focuses on stabilizing crew work and production rates. Secondly, when deciding assignments, the push method relies on the previous sequence, whereas the pull method is based on real-time scenarios and considers empty locations. Thirdly, in decisions on resource mobilization, the push method is more reactive, putting resources based on deviations. In contrast, the pull method compares the actual production rates of predecessor and successor tasks.

However, challenges arise when implementing pull-system scheduling, as Yang and Ioannou (2001) summarized. These challenges include difficulties mapping resources' work for various activities, crew splitting during progress activities, variable production rates, and intermittent activities. Furthermore, after conducting a case study using pull-driven scheduling, Ghanem et al. (2022) concluded that while this method might improve productivity, reduce idle time, and diminish task interruptions, project time can be overrun.

5. Detailed Comparison of Studied Scheduling Techniques

The table below summarizes each scheduling method's key characteristics, principles, strengths, and challenges.

Table 2.2 Comparison of Scheduling Techniques

Scheduling Method	**Description**	**Key Principles**	**Strengths**	**Challenges**
Network-based Scheduling	Utilizes network diagrams (e.g., PERT, CPM) for planning.	Time-oriented, critical path identification.	Clear visualization, critical path analysis.	Sensitivity to changes, may not handle resource constraints well.
Constrained-based Scheduling (CP)	Applies Constraint Programming (CP) for scheduling.	Constraint-driven, handles rigid activities and relationships.	Effective for rigid constraints, adaptable.	Complexity in modelling, challenges in dynamic environments.

Line of Balance (LOB) Scheduling	Suited for repetitive construction activities.	Emphasizes continuous workflow, resource focused.	Effective for repetitive projects, resource optimization.	Challenges in achieving a "natural rhythm" may require substantial commitment.
Lean Construction - Pull-Driven Scheduling	Focuses on the just-in-time flow of work, minimizing waste.	Pull-oriented, real-time decision-making, continuous workflow.	Enhances efficiency, minimizes waste.	Challenges in real-time decision-making, potential for project time overrun.

Despite significant advancements in schedule management, the outcome performances of construction projects, particularly in terms of timely completion, have still not been encouraging.

2.4 Forecasting Methods for Project Duration

As described earlier, construction scheduling techniques are closely linked to the forecasting task. Forecasting involves modelling techniques that rely on historical data to predict the future (Turban et al., 2011). In the context of time series datasets, Petropoulos et al. (2022) highlighted that forecasting is based on past knowledge to generate future predictions. These datasets represent sequential records over time, often exhibiting dependent characteristics crucial for establishing relationships between past inputs and future outcomes (Box et al., 2016). However, Litsiou (2022) pointed out that time series models work on a black-box system basis, which means that inputs and outputs are known while internal working (relationships) is not visible or fully understood.

Project forecasting is close related to risks, uncertainties, and bias, as was evidenced by Flyvbjerg et al. (2003). His research emphasized the crucial role of risk management at various levels, such as safety, cost, and environmental factors. Similarly, De Andrade et al. (2019) concluded that risks and uncertainties are common causes of project delays when studying the efficiency of the earned

schedule and earned duration management for forecasting. Given these reasons, Lovallo and Kahneman (2003) researched the forecasting bias, finding that decision-makers are experimenting with the *planning fallacy,* a typical behaviour labelled by psychologists. They described this behaviour adopted by decision makers when they over-optimize the best outcomes, such as profits or benefits, and underestimate potential costs, mistakes, etc. In this line, Flyvbjerg et al. (2009) performed extensive experiments, concluding that the consistent gaps between predicted outcomes and actual outcomes are due to "strategic misrepresentation." Its concept matches with Lovallo and Kahneman's about planning fallacy, which means that project planners tend to overemphasize the benefits while downplaying the potential costs to increase the chances of getting approval and funding for the project.

Within the realm of project control processes, Azeem et al. (2014) placed the forecasting at the end of the project control whole map, which includes monitoring actual project performance, deviations contrasting and evaluating, and outputs at project completion forecasting. Similarly, the Association for the Advancement of Cost Engineering International (AACEi) stresses that forecasting should be a set process on ongoing projects, which should actively use project control plans and baselines to verify schedule deviations. A forecasting method's categorization proposed by Montgomery et al. (2015) is qualitative and quantitative. The present research primarily delves into judgmental methods within qualitative approaches alongside deterministic, probabilistic, and machine-learning methods within quantitative techniques.

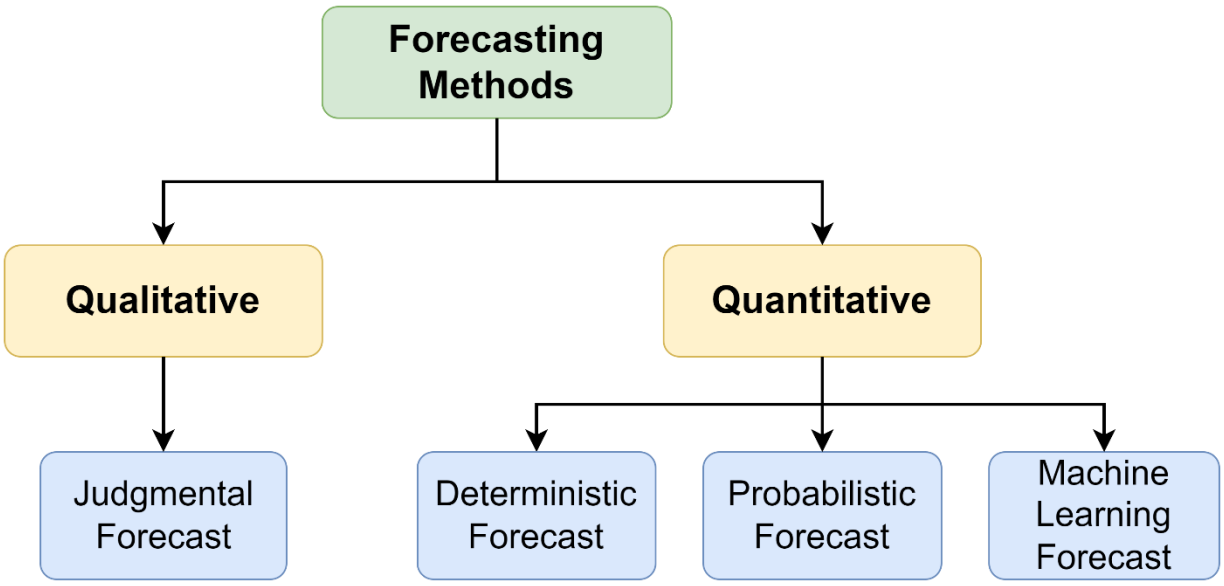


Figure 2.3 Types of Forecasting Methods

2.4.1 Judgmental forecasting

Montgomery et al. (2015) pointed out that judgmental methods are often subjective and require expert opinions for their development. These methods become particularly significant when historical data for forecasting is lacking, such as when executing a new project requiring cutting-edge design technology. In the early phases, educated guesses from experienced engineers, architects, and construction workers are crucial. Makridakis and Gaba (1998) added that judgmental forecasting relies not only on historical data but also on the biases acquired by forecasters through their practice and training. Despite judgmental methods being criticized for their nature, Caniato et al. (2011) stated that experimental studies demonstrated the significant impact of managerial decisions on expectations. For instance, Sanders and Manrodt (1994) noted that outcomes often align with judgmental elements provided by industry experts after applying any quantitative method.

A management tool for this qualitative forecasting method is the well-known Delphi method, introduced by the RAND Corporation, which engages a group of experts. Its application initiates with experts completing individual questionnaires to mitigate bias. Results from each round are reviewed and fed back to the panel with a new question, repeating the process cyclically to achieve consensus. This iterative method may reveal documented output differences, as Montgomery et al. noted (2015).

2.4.2 Deterministic forecasting

This method is part of the quantitative forecasting methods. According to Box et al. (2016) a mathematical model is deterministic if it can produce future results precisely. For instance, a cosine-based function with a time series dataset for prediction can help determine exact future values. One benefit of using deterministic models is their simplicity, which makes them understandable and implementable for practitioners. It also requires less data and facilitates straightforward output processing to achieve the primary goal of understanding project performance (Ballesteros-Pérez et al., 2020).

In this vein, Barrientos-Orellana et al. (2021) emphasized Earned Value Management as the most prominent method in the construction sector due to its easy implementation. Also, Wilson et al. (2003) noted that both Gantt charts and Critical Path Method (CPM) as deterministic methods hold a strong position in construction practice despite their significant limitations, such as overlooking the lack of variability control in activity durations and resulting impacts. Kim's (2007) research compared the reliability of deterministic and probabilistic methods when forecasting the final project performance. He tested the Earned Value Management (EVM) and the Critical Path Method (CPM) as deterministic against the Kalman filter forecasting method (KFFM) and the Bayesian adaptive forecasting method (BAFM) as probabilistic. Initially, this study found that the

EVM outperformed the CPM in the realm of deterministic methods because the CPM method lacks dynamic updating of original estimates with project performance data, reducing its capacity to predict future activities. As a result, probabilistic methods demonstrated superior performance compared to deterministic ones. This is attributed to their capacity to forecast future outcomes, leveraging a combination of historical data from similar projects, judgmental insights, and preliminary project information obtained at the early stages.

2.4.3 Probabilistic forecasting

Unlike the forecasting deterministic methods, the probabilistic one addresses situations where numerous unknown inputs exist to compute predictions. In some cases, combining deterministic with probabilistic models yields multiple forecasting values that are bounded according to specific restrictions. This derived model is the probability or stochastic model (Box et al., 2016). Similarly, Montgomery et al. (2015) stressed that probabilistic forecasting results are intervals instead of a unique value, and this feature is worthy in a risk and uncertainty construction environment. For example, Barraza et al. (2004) applied probabilistic methods to the S-curve to obtain stochastic S-curves. These S-curves would provide possible solutions between preset upper and lower limits, which enable considering other parameters to tackle project uncertainties.

In regards to the accuracy of predictions, Abdel Azeem et al. (2014) compared the Kalman Filter Forecasting Model (KFFM) with the Earned Schedule (ES) model, finding that the former (probabilistic method) showed better performance than the latter (deterministic method). Research also compared probabilistic performance against machine learning models. Makridakis et al. (2018) assessed a large set of traditional probabilistic methods mentioned as follows: (1) Naïve 2, (2) Simple exponential smoothing, (3) Holt exponential smoothing, (4) Damped exponential smoothing, (5) SES, Holt and Damped (Comb), (6) Theta method, (7) automatic model selection

algorithms for ARIMA and finally (8) exponential smoothing (ETS). Consequently, probabilistic methods exhibited superior performance compared to Machine Learning algorithms. It is essential to note that this research focused solely on Machine Learning and did not include an assessment of Deep Learning algorithms, implying certain assumptions and limitations in the study.

2.4.4 Machine Learning for forecasting

Abioye et al. (2021) stressed that forecasting using Artificial Intelligence models has the potential to be used in many fields within the construction industry. According to Pan et al. (2021), the applicability of machine learning in construction has grown substantially, particularly in forecasting tasks, owing to its ability to process extensive datasets from diverse sources and return approximate outcomes. Whereas machine learning algorithms often yield accurate results individually, it is also a common practice to ensemble multiple algorithms, combining their strengths to enhance predictions. For instance, the support vector machine (SVM) and fast-messy genetic algorithms (fmGA) produce the Evolutionary Support Vector Machine Inference Model (ESIM) for the prediction of construction management problems (Cheng & Wu, 2009). In this case, the SVM addressed learning and curve fitting, while fmGA deals with the optimization task. Among the extensive list of forecasting models, primarily ensembled models, the Support Vector Machine (SVM) and Artificial Neural Network (ANN) algorithms have had significant attention in the literature. The former works into higher-dimensional space, building optimal hyperplanes which give global solutions, while the latter, also classified as a subset of Machine Learning algorithms, comprises interrelation neurons, activation inputs, cyclic processes so that it can imitate the process of human learning.

The differences between Machine Learning and Probabilistic models were illustrated by Nielsen (2019) using Time Series datasets. Probabilistic models require a theory to represent time series

data coupled with a parameter to monitor deviations and uncertainties. Once established, they can be utilized for prediction purposes. On the other hand, Machine Learning models rely on identifying patterns through complex mathematical algorithms to set up the behaviour, enabling it to predict future outcomes. Their prediction results were also compared. Makridakis et al. (2023) recently compared Machine Learning, Probabilistic and Deep Learning (a Machine Learning subcategory) models. They selected representative models for each method and ranked their accuracies in the context of the well-known M3 forecasting competition, encompassing 3003-time series datasets. The outcomes revealed that Deep Learning models outperformed others by using monthly data and yielding long-term predictions. Also, as an aspect to be improved, the Deep Learning algorithms compromised considerable Computational Time under this competition conditions.

2.4.5 Detailed Comparison of Forecasting Methods

Table 2.3 below shows relevant aspects of each forecasting method according to the previous discussion.

Table 2.3 Detailed Comparison of Forecasting Methods

Method	Key Characteristics	Advantages	Limitations
Judgmental	Subjective, relies on expert opinions	Quick implementation, functional with limited data	Prone to biases, lacks objectivity
Deterministic	Uses mathematical models to predict future outcomes	Easy to understand, suitable for projects with less data	Ignores uncertainties, may not capture complex patterns
Probabilistic	Considers uncertainties and deviations in predictions	Captures variability, provides a range of possible outcomes	Requires supporting theory, may be data-intensive

Machine Learning	Utilizes complex algorithms to identify patterns and behaviors	Adapts to complex data, can handle large datasets	Requires significant computational resources, may lack interpretability

2.5 Machine Learning (ML) Overview

The purpose of this study is to delve deep into machine learning forecasting. Therefore, this section will describe the essential aspects of this topic. The most accepted definition of Machine Learning by research is credited to Tom M. Mitchell (2013), who stated that learning starts from experience "E" in tasks "T," measured by performance "P" when its ability in these tasks improves through the accumulation of experience "E." Likewise, Machine Learning is considered a foundational aspect of Data Mining, enabling the extraction of valuable insights from raw data in databases for various purposes (Witten & Witten, 2017). Chollet (2018) highlighted that whereas traditional programming inputs *data* and *rules* to yield *answers* through explicit programming, Machine Learning inputs *data* and *answers* to yield *rules* through training, as shown in Figure 2.4.

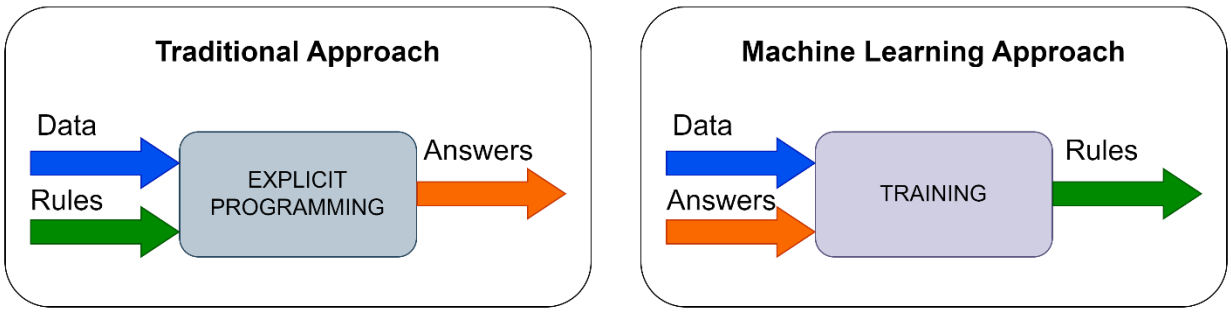


Figure 2.4 Comparison Between Traditional and Machine Learning Approaches

Also, it is essential to notice that Artificial Intelligence (AI), the parent branch of Machine Learning, is a non-new technology; on the contrary, they have gained momentum in the last three decades because of the intensive increment of computer power to handle vast quantities of data

(known as Big Data), which have enabled them to deploy high-demanding computational algorithms (Y. Zhang et al., 2014).

When talking about Machine Learning, mentioning the Deep Learning algorithms is unavoidable. They are a subfield of Machine Learning, as represented in Figure 2.5. Deep Learning is centred on constructing expansive Neural Network models capable of making precise data-driven decisions. It involves the creation of intricate architectures with multiple layers, enabling the system to learn complex patterns and representations from data autonomously, ultimately enhancing its capacity for accurate decision-making (Kelleher, 2019). Indeed, "Deep" refers to many Neural Network layers that can solve specific and complex problems (LeCun et al., 2015).

These neural network–based algorithms are characterized by mimicking the human brain's functioning by replicating biological components such as neurons or dendrites. They resemble stimulations among neurons through *feed-forward* and *back-forward* propagations, as these are known in deep learning terms (Auffarth, 2021). Prominent subcategories of deep learning algorithms are Recurrent Neural Networks (RNN), Modular Neural Networks, Convolutional Neural Networks (CNN), and Radial Basis Function Neural Networks.

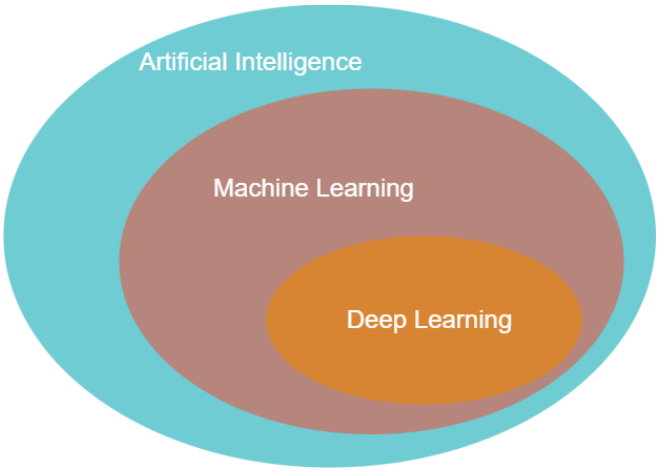


Figure 2.5 Artificial Intelligence, Machine Learning and Deep Learning

In the forecasting field, deep learning algorithms have delivered promising results in contrast to conventional pure machine learning models; because of that, they have become a center of attention, especially for time series prediction problems in the last years (Chandra et al., 2021; Makridakis et al., 2023).

2.5.1 Machine Learning Types: Supervised, Unsupervised and Reinforcement

There are three main types of machine learning: Supervised, Unsupervised, and Reinforcement Learning (Auffarth, 2021; Lazzeri, 2021; Turban et al., 2011). *Supervised Learning* is a kind of induced learning. This process occurs when the set of observations *knows* their outputs. It means that both inputs and outputs should be provided. Haque et al. (2022) stressed that *Supervised Learning* aims to define a relationship between inputs and output variables from the training dataset. In addition, Abioye et al. (2021) added that supervised learning is concentrated on an algorithm that makes decisions based on previous knowledge acquired, explicitly, prior understanding of the relationship among variables to deliver a specific output. "Supervised" refers to data scientists supervising the learning process as they know actual outcomes and can share iteratively to enhance the learning performance (Alachiotis et al., 2022; Lazzeri, 2021). Research in this field has categorized chiefly Supervised Learning into regression and classification, further explained in the next section. The mathematical expression to represent the said relationship is described as follows. If x is a feature and y an output, the equation in supervised learning would be $y = f(x)$.

Unsupervised Learning is a type of machine learning that deals with datasets where the output is unknown (Nelles, 2001; Turban et al., 2011; Vermeulen, 2020). In a Classification problem context, this would mean that the dataset is unlabelled, and no classes are associated with the observations, making it difficult to solve. In Unsupervised Learning, the goal is to identify

relationships between the observations and create a function $f(x)$ (where x represents inputs) without the use of labels (Abioye et al., 2021). As examples may be mentioned, the Clustering and Dimension Reduction techniques are two main categories of Unsupervised Learning (Wang, 2016), while Lazzeri (2021) suggested that anomaly detection and principal component analysis might also be included as examples.

Reinforcement Learning, on the other hand, is quite different from Unsupervised Learning (Turban et al., 2011). It does not rely on historical data to start the learning process, and there are no natural groupings in the dataset. Instead, the learning process is driven by interacting with the environment to create experience-based outputs, and the model's efficiency is improved through a trial-and-error process (Auffarth, 2021). The following figure broadly categorizes the types of Machine Learning and their main subcategories.

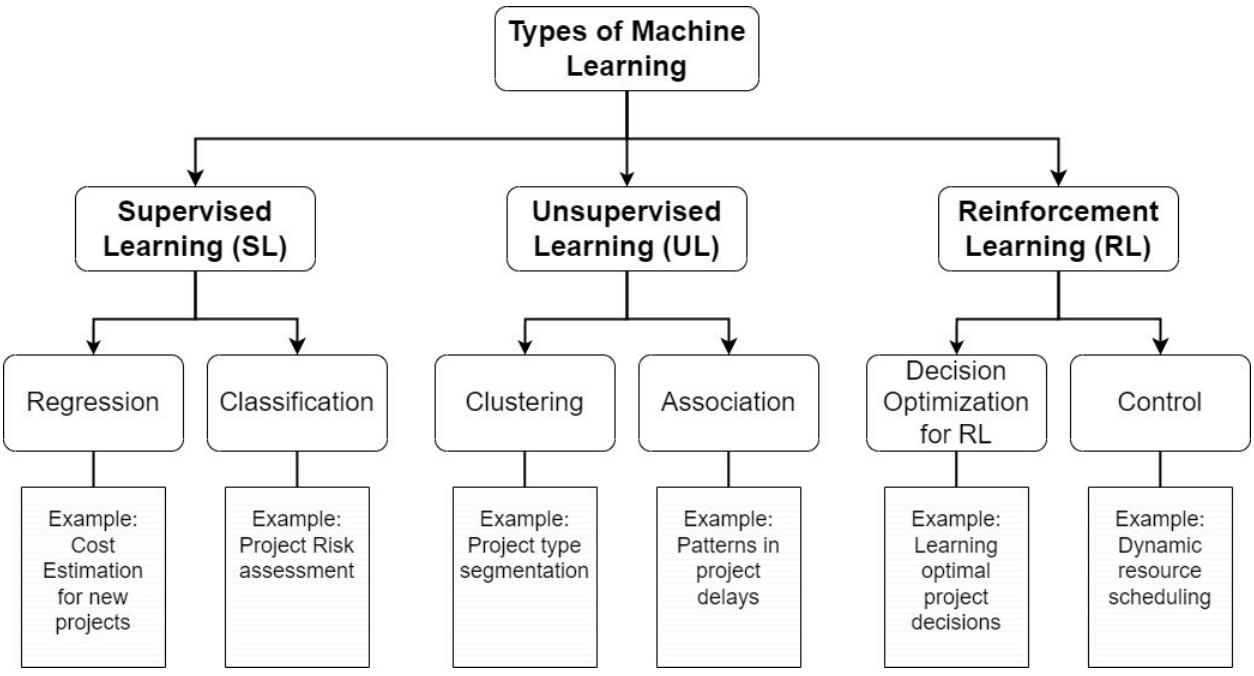


Figure 2.6 Types of Machine Learning

2.5.2 Data Mining Tasks: Classification, Regression and Clustering

1. Classification.

The classification is arguably the most widespread task in data mining when seeking valuable outcomes (Kesavaraj & Sukumaran, 2013; Linoff et al., 2011; Umadevi & Marseline, 2017). They added that it is a common practice by human beings during the communication process when categorizing or establishing grading to be understood by one another. In this way, classification is grouping objects into *predefined classes*. Aggarwal (2015) argued that, unlike clustering, the classification is developed on a training dataset containing one or more "target variables." Bhattacharyya et al. (2020) outlined that classification might be based on endless criteria. For instance, Kotsiantis et al. (2006) proposed two categories based on their development process: firstly, artificial intelligence-based development, which in turn encircles logic-based techniques and perceptron-based techniques and, secondly, statistic-based development, encompassing Bayesian networks and Instance-based techniques. As examples in the artificial intelligence category, logic-based techniques comprise decision trees and rule-based classifiers whereas perceptron-based algorithms are neural networks.

2. Regression

The regression task is one of Supervised Learning, which struggles to find relationships between variables that affect the output variable. These variables can be independent or dependent (Miller, 2017; Yildiz et al., 2017), and the function that encircles this relationship is called the Regression Function. Moreover, the dependent variable is numeric and continuous (Harrington, 2012). An algebraic representation of the variable's relationship is given as follows:

$$\boldsymbol{y' = a_1 \times x_1 + a_2 \times x_2 + a_3 \times x_3 \ldots\ldots + a_n \times x_n}$$

$$\boldsymbol{y = y' + error}$$

Where, $a_1, a_2, \dots a_n$ are coefficients, $x_1, x_2, \dots x_n$ are independent variables, y' is the output or dependent variable and y es the actual output so that the difference would be the *error* (Yildiz et al., 2017). This simplistic representation assumes only one output instead of multiple outputs. It also includes various independent variables, which are known as *multivariate* variables. Furthermore, the regression on the time series dataset is closely related to forecasting and is supported by machine learning algorithms (Lazzeri, 2021).

3. Clustering

Gan et al. (2007) defined data clustering as creating groups of objects called clusters based on similarities. Also, such objects from separated clusters are different from each other. They added that data clustering is known as segmentation analysis, cluster analysis, taxonomy analysis or unsupervised classification. Linoff and Berry (2011) highlighted that there are no predefined classes in clustering; therefore, users may determine the denotation of each cluster. In addition, clustering is characterized by often being executed in the early stages compared to others during the data mining. It is useful, for example, for market segmentation before launching any market research to know habits from the objective group of people.

Clustering can be categorized into two groups (Gan et al., 2007): hard clustering and soft clustering, based on the objects' belonging. Objects in hard clustering are likely to belong only to one cluster, while soft clustering could go to two or more clusters. Diving hard clustering contains two types: hierarchical algorithms and partitional algorithms. Thus, hierarchical algorithms can be divided into divisive and agglomerative hierarchical algorithms. The first one creates clusters from top to bottom direction, which means that it starts with a big cluster encompassing all the objects, and then more clusters will be made starting from this primary cluster by partition. On the contrary,

the agglomerative hierarchical algorithm works from bottom to top direction, meaning that each cluster only encloses one object, then clusters will be created, unifying these.

Below is a comparison of the data mining tasks described.

Table 2.4 Comparison of Data Mining Tasks

Task	Objective	Output	Example
Classification	Assign input data to predefined classes or labels.	Discrete categories or classes.	Classifying project risks as high, medium, or low.
Regression	Predict a continuous numerical value based on input data.	Continuous numeric values.	Predicting the estimated completion time for a project phase.
Clustering	Group similar data points based on features or characteristics.	Discrete clusters or groups.	Grouping similar project tasks for resource optimization.

2.5.3 Time Series Datasets for Machine Learning Forecasting

As construction projects yield overwhelming amounts of diverse data, it is essential to understand time series datasets, which are the specific types managed in this research.

1. Structured and Unstructured Dataset

Datasets can be categorized by their ordering criteria, Structured and Unstructured. The structured data is highly organized and recognized, mostly in rows and columns, which matches most conventional relational database management systems (RDBMS) (Mishra & Misra, 2017). Hopkins et al. (2022) underlined that it could be easily understandable for human beings regarding amount and organization. An example of a structured dataset is the Time Series Dataset, arranged in a tabular format where each row represents a timestamp, collecting observations over time. On the contrary, unstructured datasets are initially hard to comprehend. Despite this characteristic,

most business data is generated in an unstructured form, accounting for around 80%. To make datasets applicable to machine learning tasks, it is necessary to transform them into structured data. However, this transformation can be intricate and computationally demanding (Cropper et al., 2016; Mao et al., 2023; Mishra & Misra, 2017).

2. Univariate and Multivariate Time Series Dataset

Another relevant aspect of datasets involves time series analysis, oriented toward forecasting. According to each machine learning problem goal, one input or multiple inputs can be considered to represent the analysed event better. Thus, they might be univariate and multivariate. In many cases, these events are essentially multivariate; for example, temperature forecasting implies the concurrence of diverse variables such as pressure, humidity, location, etc. (Karimi-Bidhendi et al., 2018). Analogically, diverse data is gathered during the project tracking, which finally affects the project duration. Regarding the complexity of managing Univariate and Multivariate TS datasets, the latter is usually more complex because many internal or external factors affect each variable (G. Li & Jung, 2023).

2.5.4 Artificial Neural Networks

Artificial Neural Networks (ANN) are those algorithms within the Machine Learning field. The ANN represents the learning process, simulating the human learning process and adapting elements like neurons and synapses (Aggarwal, 2015). A simplistic view would describe the ANN as a multilayer way to learn over an input dataset (Chollet, 2018). Specifically, it works as follows: first, input variables must be weighted based on their incidence over the expected solution. Sometimes, weights are also known as parameters. Second, the weighted inputs are collected by a "neuron cell" and passed on to an "activation function." However, the complexity of this process is rooted here because this network will yield hundreds of weights and then should select the

correct values to obtain the calculated output. Third, the loss function was introduced to monitor the process, scoring the error between the calculated and expected values. Fourth, ANN algorithms allow the optimizer to implement a backpropagation algorithm to address high errors. It is the hearth of the ANN. This iterative training process involves adjusting weights multiple times to minimize the error. Finally, it is obtained as an outcome of ending the process, represented as follows (Chollet, 2018; Dinov, 2023).

$$\boldsymbol{y(x) = f\left(\sum_{i=1}^{n} w_i\, x_i\right)}$$

Where w means the calculated weights and x are the independent variables. Furthermore, Dinov (2023) proposed three crucial parts when building a Neural Network algorithm: an activation function, a network topology concerning neuron and layer quantities and a training algorithm to polish weights/parameters of the input variables.

1. Multi-Layer Perceptron (MLP)

The Multi-Layer Perceptron is a subtype of the Artificial Neural Network algorithms and is the most common neural network type. Moreover, it is an extension that enables the creation of networks with multiple layers because of its architecture (Singh & Banerjee, 2019). Its creation is rooted in the Perceptron model proposed by Rosenblatt in 1950, which included linear input and output layers to solve problems (Ramchoun et al., 2016). However, most issues, such as classification and regression, look for the fittest curve to represent the behaviour that is not necessarily linear (See Figure 2.7). The MLP overcame this linear drawback by incorporating layers in between the input and output layers. They are called *hidden layers.* (Taud & Mas, 2018). For some, the MLP can be recognized because it uses three or more layers in its architecture.

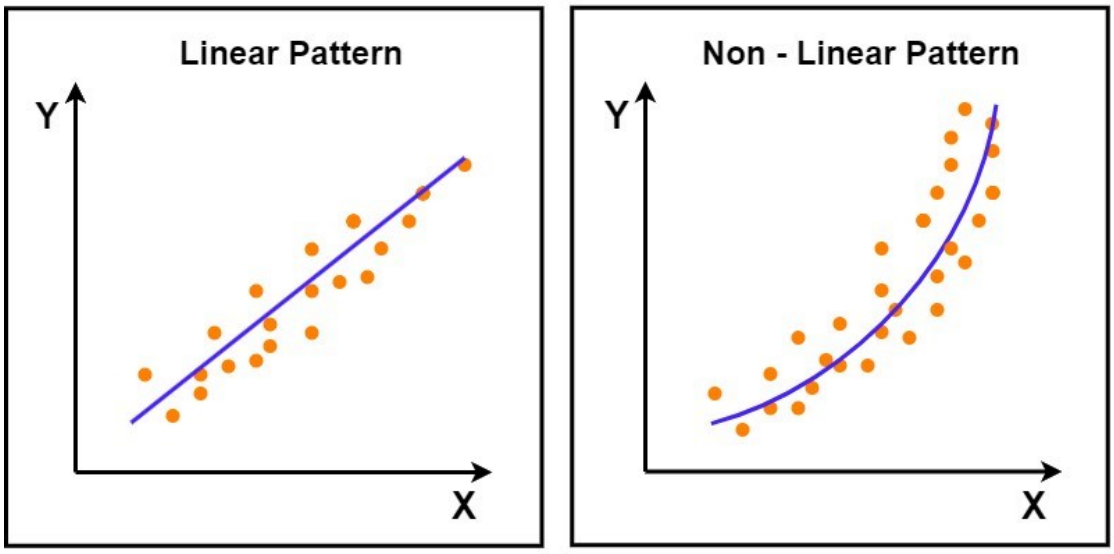


Figure 2.7 Linear and Non-linear Data Patterns

Accordingly, the MLP is defined as a mapping function fitted between input and output variables, which, in turn, is very useful for solving Time Series regression problems (Brownlee, 2018). This model also follows the ANN's main features as feedforward and backpropagation to find the minimal error, but this time, it includes a nonlinear activation function in one or more neurons (Dilipkumar & Durairaj, 2022). The mathematical expression for the MLP is the connection of several fully connected layers represented as an input matrix called $X_{n\,x\,m}$ to yield an output matrix $Y_{n\,x\,k}$. During this process, a weight matrix $W^{l}_{m\,x\,k}$ for layer l that contains i rows is yielded. Each row corresponds to the weights leading from all of units i in the previous layer to all of units j in the current layer. Lastly, the product matrix $X\,x\,W$ has dimensions $n\,x\,k$. Also, it should be considered the bias vector $b_{k\,x\,1}$ as part of the final mathematical expression. Each layer produces an output that can be represented as:

$$Y^{(l)}_{n\times k} = f^{(l)}_{k}(X_{n\times m}W_{m\times k} + b_{k\times 1}).$$

2. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are specialized neural networks that process data with a known grid-like topology. They excel in handling various types of data, such as time-series data (1-D grid) and image data (2-D grid); typically, in the literature, they are found as CONV-1D and CONV2D, respectively, even though the CNN can handle more dimensions. The term "convolutional" indicates using a mathematical operation called convolution, a specialized linear operation. In practical applications, CNNs have demonstrated remarkable success by incorporating convolution instead of general matrix multiplication in at least one of their layers (Goodfellow et al., 2016).

As CNN is primarily applied to image classification problems, it can be used to Time Series datasets, making some customizations. Some authors called this specific model Temporal Convolutional Networks (TCN) based on Bai et al. (2018) research (Gridin, 2022). These adaptations start by considering Time Series problems as *one-dimensional* convolutional Neural Networks represented as CONV-1D. Secondly, CNN's native properties, Causal Convolution and Dilation, are leveraged. On the one hand, the Causal Convolution, also called Equivariance, ensures that the output at a certain time depends only on the current and past inputs, not future inputs. It is crucial in tasks like time series forecasting or any sequence prediction scenario where the model cannot access future data points. (Goodfellow et al., 2016; Gridin, 2022).

On the other hand, dilation is the interval in the input sequence that produces the output values. It is also known as steps or cadence. For example, some time series datasets identify as a pattern the fact that a group (kernel) of intercalated inputs is linked to a reliable output (see figure 2.8) (Gridin, 2022; Gutman & Goldmeier, 2021).

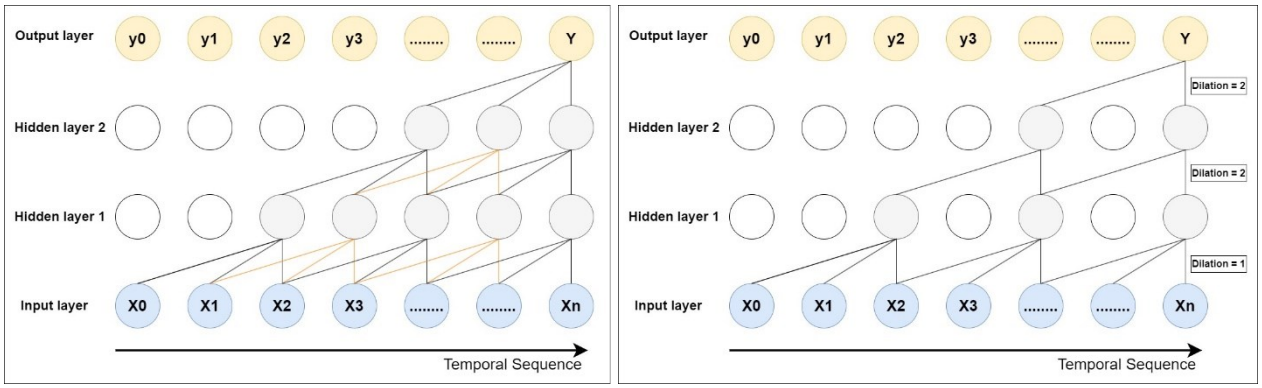


Figure 2.8 Causal Convolution (left) and Dilation (right) CNN Properties (Gridin, 2022)

3. Recurrent Neural Networks: The Long Short-Term Memory

A subset of ANN is the Recurrent Neural Networks, which implement *memory* in each neuron. It is represented by an internal loop which stores information in each iteration to polish the internal output (Chollet, 2018). Figure 2.9 represents a recurrent connection.

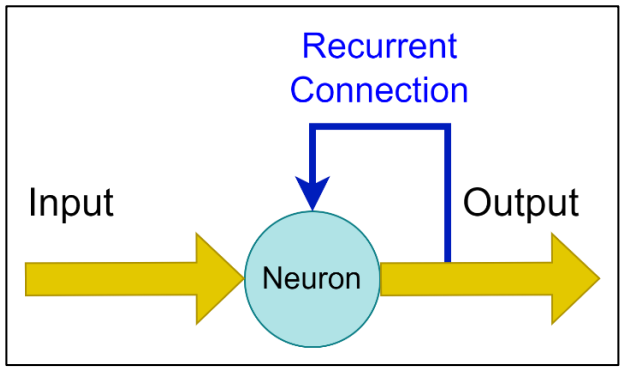


Figure 2.9 Neuron's Recurrent Connection Representation

Hochreiter and Schmidhuber developed the Long Short-Term Memory (LSTM) in 1997, an improved variant of the vanilla RNN. Typical RNN models vanish or explode parameters, having a reduced scope of memory (Nowrin, 2022). The LSTM performs much better Time Series datasets than conventional RNN algorithms due to its memory capability (Chandra et al., 2021; Staudemeyer & Morris, 2019), which is carried out by the incorporation of two additional

components known as *hidden and cell states* to memorize patterns in a short and long term, respectively (Gridin, 2022; Lazzeri, 2021; Nowrin, 2022). The standard representation of an LSTM neuron (also known as a cell or unit) is shown in Figure 2.10.

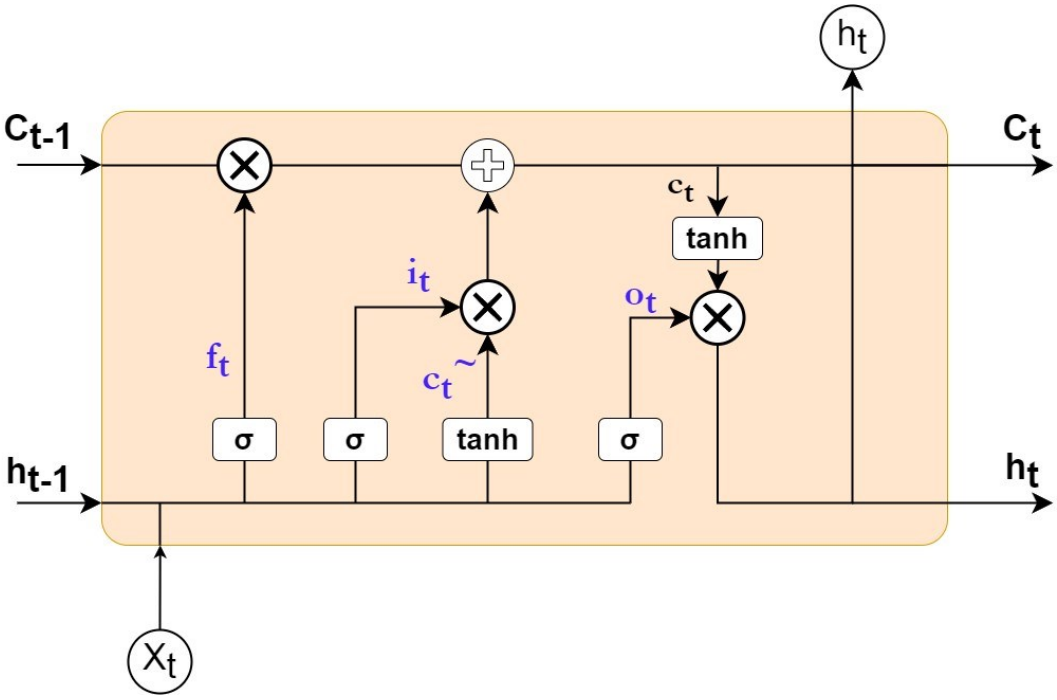


Figure 2.10 Long Short-Term Memory (LSTM) Neuron

Figure 2.10 identifies three gates known as forget, input and output gates, represented as f, i and O, respectively. While the forget gate decides what information should be unused from the cell state, the input gate decides which values will be used from new inputs. Then, the output gate decides what the next hidden state, h_t,, should be. Also, the hidden state is used for predictions and passed to the next step. The mathematical expressions that show how the LSTM is depicted below (Ling, 2023):

$$f_t = \sigma(W_{fh} \odot h_{t-1} + W_{fx} \odot x_t + b_f)$$

$$i_t = \sigma(W_{ih} \odot h_{t-1} + W_{ix} \odot x_t + b_i)$$

$$o_t = \sigma(W_{oh} \odot h_{t-1} + W_{ox} \odot x_t + b_o$$

$$\widetilde{c_t} = tanh(W_{ch} \odot h_{t-1} + W_{cx} \odot x_t + b_c)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \widetilde{c_t}$$

$$h_t = o_t \odot \tanh(c_t)$$

The state cells, denoted as c and h in the Current and Hidden Cells, as illustrated in the figure above, store both long-term and short-term memory. These cells also establish connections between the current timestamp information and a broader range of previous ones, allowing the model to memorize previous details. The LSTM can polish each iteration's output several times, considering extensive past observations in its calculations. Thus, this model is suitable for time series problems.

4. Detailed Comparison of explained ANN algorithms.

Table 2.5 Comparison of MLP, CNN and LSTM Algorithms

Feature	**MLP**	**CNN**	**LSTM**
Architecture Type	Feedforward	Feedforward with Convolution	Recurrent
Data Type	Structured Data	Grid-like Data (Images, Sequences)	Sequences (Time Series, Natural Language)
Layer Types	Input Layer, Hidden Layers, Output Layer	Convolutional Layers, Pooling Layers, Fully Connected Layers	Input Layer, Hidden Layers with Memory Cells, Output Layer
Parameter Sharing	No	Yes (through convolutional kernels)	Yes (through recurrent connections)
Feature Extraction	Manual Feature Engineering	Automatic Feature Learning	Automatic Feature Learning with Memory
Use Cases	Tabular Data, Numeric Predictions	Image Classification, Object Detection, Spatial Data	Time Series Prediction, Natural Language Processing
Memory Handling	No Memory Handling	Limited Memory Handling	Explicit Memory Handling for Sequences
Applicability	Generalized for Various Tasks	Specialized for Grid-like Data	Specialized for Sequential Data

2.6 Application of ML for Forecasting Construction Project Duration

Machine learning, as a broader concept, has emerged as a promising solution for predicting project duration in the construction industry. This is primarily due to its ability to handle large amounts of data, identify meaningful patterns, and generate valuable insights that align with the need to manage the vast amounts of data generated by construction activities. This data is often poorly leveraged for reasons such as lack of proper data collection, lack of standardized processes, resource constraints, risk aversion to subsequent processing tools like machine learning, and so on. However, notable advancements in academia might encourage industry practitioners to apply it to large-scale construction projects.

For instance, creating an integrated model, which creates an adaptative ANN model using a genetic algorithm. Lishner and Shtub (2022) addressed the need to adapt an ANN model to different construction organization features such as uniqueness, management techniques, organizational cultures, etc. by using a genetic algorithm that optimizes the generic ANN model to various scenarios. It means that hyperparameters can be modified to match the number of hidden layers or neurons. The model was tested with two organizations as predictors for the first company were the project type, business units, project risk levels, start dates, project actual duration and planned start dates, while stability of project scope, projected estimated duration, importance of time, among others for the second organization. Finally, they obtained 25% and 17% as MAPE (error) by predicting the project duration, respectively.

Wu et al. (2022) developed a framework using a backpropagation (BP) artificial neural network (ANN) to forecast the power grid project duration, involving as predictors quantities, voltage level, number of callable units (resource availability), and construction conditions (climate and environmental). By analyzing project Gantt charts and leveraging historical data, the research

determines key project execution paths through network node diagrams. In another study, Sanni-Anibire et al. (2022) used survey data from 48 construction practitioners against 36 potential risk delay factors previously selected from a literature review. This study used K-Nearest Neighbors (KNN), Artificial Neural Networks (ANN), Support Vector Machines (SVM) and Ensemble methods, concluding that ANN shows the best accuracy (93.75%) in the duration prediction for building construction projects.

Similarly, Yudhi (2022) could predict project completion with supervised machine learning, reaching an accuracy of 98.6%, while applying the traditional methods, the accuracy was 40%. This study also highlighted that predictions were obtained at construction task levels instead of at the whole project level as usual. On the other hand, Lawal et al. (2023) compared the multilayer perceptron (MLP) and a radial basis function (RBF) model to forecast the project duration of building renovation. This study encompassed 121 questionnaires from specialized construction firms, finding that the MLP was overcome with an accuracy of 86% against 80% for the RBF. The industrial building construction duration was addressed by Leu and Liu (2016) by using the Principal Component Analysis (PCM) algorithm to select duration influencing factors, so then, apply a Backpropagation Neural Network (BP-NN) to predict the duration. This study involved 1538 industrial projects, with more than six months of duration each, and considering four categories for the modelling: case type, participant, location, and time. Likewise, Cheng (2019) addressed this problem by analyzing two types of factors that affect project duration: sequential and nonsequential. The former was handled with the LSTM algorithm, while the latter was addressed with a typical ANN network. The model was named NN-LSTM. Cheng tested 226 historical information from 11 construction projects and 14 factors. After testing, the results

displayed good performance, with a mean absolute percentage error (MAPE) of less than 5% and a mean absolute error (MAE) of 2%.

Applying Machine Learning to predict construction project duration is a transformative approach, but it is still a developing field. So far, Deep Learning has been more prevalent than Machine Learning, and among the Deep Learning algorithms used, ANN has been the most explored compared to RNN models. The research has mainly focused on classification problems like the schedule risk assessment approach rather than regression through predictive modelling. Although challenges such as data quality and model interpretability persist, ongoing research and technological advancements offer promising results.

2.7 Schedule Delays in Construction Projects

A common setback in construction projects related to project duration is delays, which lead to failed projects (Park, 2021; Shahhossein et al., 2017; Yates & Lockley, 2002). That is why it is helpful to understand their causes, relevance to the project duration and the existing gap in the construction industry. In this sense, Merrow (2011) found that the average schedule slip is 28% in industrial megaprojects. This study encompassed 318 projects worldwide, including oil and gas production plants, petroleum processing and refining megastructures, mineral and metals plants, and chemical plants. Another comprehensive study by Ansar et al. (2014) delved into the performance of 245 large dams executed between 1934 and 2007 worldwide. The findings indicated that 8 out of 10 large dams experienced delays, accounting for a substantial 44% increase in their schedules. Ayalew et al. (2016) conducted a study in the Ethiopian construction sector, revealing schedule delays ranging between 61% and 80%. Some evidence was listed among many more. These prevalent delays were also exhaustively studied by Flyvbjerg (2003), calling this

phenomenon a ‘paradox,’ where achieving timely completion remains a pervasive issue despite advancements in project management practices.

2.7.1 Schedule Delay factors

The analysis of delays in construction projects centers on identifying causes and effects impacting the project's critical path, consequently influencing the final project duration (Al-Saggaf, 1998; J.-B. Yang & Kao, 2012). Research on delay factors started with data-gathering methods, which can utilize both literature reviews and surveys or rely solely on literature reviews. Then, they were usually categorized according to project types (e.g., infrastructure, industrial, residential), and project locations (typically categorized by countries or regions), among other aspects. Regarding data gathering, an approach adopted by academia involved selecting delay factors from prior studies and ranking them through surveys of industry practitioners. Another approach relied on a comprehensive literature review and elaborated a prioritized list of delay factors. In this sense, Assaf and Al-Hejji (2006) surveyed 23 contractors, 19 consultants, and 15 owners in Saudi Arabia’s construction environment. Focussing on Large construction projects, this study spanned 76 projects where 45 were delayed. This study also categorized delay causes according to Owners, Contractors, and Consultants. As a result, this research detected 73 delay causes, the most frequent of which were “change orders” among the three parties.

Another relevant study was performed by Sanni-Anibire et al. (2022), reviewing analytically representative past worldwide studies in the last 15 years. As a result, 36 factors of delays were reiterative in the construction sector, the top five below: "Financial challenges faced by the contractor," "approval delays for completed work," "slow material delivery," "ineffective site organization and coordination among involved parties," and "inadequate resource planning and scheduling estimation.". In addition, it is worth mentioning that this study was focus on building

construction type. A similar scope was observed in the research conducted by Durdyev and Hosseini (2019). This research systematically reviewed worldwide studies developed between 1985 and 2018. From 149 identified causes, the top ten most recurrent ones were composed of "weather/climate conditions," "poor communication," "lack of coordination and conflicts between stakeholders," "ineffective or improper planning," "material shortages," "financial problems," "payment delays," "equipment/plant shortage," "lack of experience/qualification/competence among project stakeholders," "labour shortages and poor site management".

Sepasgozar et al. (2019) selected 94 research studies from 29 countries worldwide, identifying the causes and effects of delays in the construction industry. Consequently, 30 critical factors among the most relevant are scheduling (Improper resource planning, inaccurate budgeting, procurement, unreal scheduling), Payment delays to labourers or contractors, Design and scope changes, unqualified workforce (workers, technical staff) and Financing and cashflow issues (insufficient contingency allowance, penalties or loan gaining problems)—also, this research comprised among residential, building, industrial and infrastructure types of constructions. Unlike previous investigations, Selcuk et al. (2024) assessed 70 journal articles developed in 33 developing countries. This study identified the 30 most frequent delay causes, such as material procurement, change orders or uncertainty in project scope, problems in supplying labour and technical staff, delayed payments to contractors by owners, unforeseen weather conditions, deficient or incomplete design documents and specifications, deficient management skills of contractors, equipment procurement issues, poor communication among the parties, lack of proper budgeting and planning of the contractor. Likewise, the most recurrent type of construction considered was building and infrastructure projects; conversely, one industrial project only was analyzed in this research.

Similarly, Kermanshachi and Pamidimukkala. (2023) spanned main project lifecycle phases like design, procurement, and construction. This research comprises 44 case studies from a literature review on heavy industrial projects, with the authors defining three main categories: project general aspects, project-specific features, and best construction practices. After a survey of over 140 construction practitioners, this study finds the following delay factors in the design phase: size of project team, agreement with penalty clauses for project delays, providing of part of engineering at the beginning of the project, number of budgeting stages, proper financing procedures, interaction among designer, engineer, and contractor parties, count of vendor and subcontractor entities, regulations, adequate project team interactions, appropriate rate of employees, skill workers on fields, matching between project objectives and physically accomplish components, efficient change management process, inspections by external entities, clarity of owner requirements, proper resource management implementation, conflict resolution method implemented, impacting of best practices strategies.

Likewise, the procurement phase shows the following indicators: level of project engineering schedule performance, complexity of design and technology, collaboration among project parties, amount of implementation locations for procurement activities, quality of resources (labour and bulk materials), and impact of change order timing. In the construction phase main aspects, they identified cost overruns from the engineering phases, actual duration of the engineering phase, involvement of the project team in procurement, close relationship with technologies by organizations, Level of completion of engineering/design project, difficulty of procurement of machinery due to project location, time gap from required changes.

Despite the myriad schedule delay factors found by the Academia, the lack of a consensual categorization is predominant. A standardized categorization would help take appropriate steps to

reduce the impact of such delays (Selcuk et al., 2024). A classification proposed considering the liability of delays as the foundation: compensable, excusable, non-excusable, and concurrent. Compensable refers to delays caused by the owner, while excusable means unforeseeable causes. Non-excusable delays are caused by third parties such as contractors or subcontractors, and concurrent delays are a combination of various types (Kraiem & Diekmann, 1987). Another classification proposed by Enshassi et al. (2009), categorizes delay factors as critical or non-critical based on their impact on the project's critical path. Although numerous attempts have been made so far, there are no consensus reached yet.

2.8 Summary and Research Gap

Numerous studies have found that schedule management is essential to construction project management. In schedule management, the schedule model plays a crucial role in forecasting. The schedule model is created by adopting a scheduling technique to control scheduled tasks, including forecasting. Several scheduling techniques, such as CPM, PERT, PDM, CCPM, and the 'Pull-System' approach, are available. However, each technique has its own set of drawbacks. For instance, these techniques may face challenges such as inefficiency in overlapping situations, timeliness issues, accuracy concerns, buffer sizing difficulties, and coordination demands.

On the other hand, current most extended forecasting methods such as Judgmental, Deterministic (e.g., EVM), and Probabilistic (e.g., ARIMA or Markov Chain) exhibit limitations. Specifically, despite its prevalence, judgmental forecasting has known shortcomings, such as being prone to bias, which is a consequence of its high dependency on expert opinions. Moreover, deterministic forecasting, particularly EVM, struggles with time forecasting, especially during project completion. Probabilistic Methods, while effective, often require a large quantity of input

variables. Amidst these challenges, Machine Learning emerges as a promising alternative for forecasting. It offers flexibility in handling large amounts of data and identifying patterns from the past to predict accurate durations. Moreover, Artificial Neural Networks (ANN) overcome typical Machine Learning algorithms in time series forecasting. Likewise, its application requests accomplished some requirements previously, such as obtaining structured data and performing exhaustive data preprocessing.

Academic research on Machine Learning in the construction industry and time series forecasting evidenced some gaps. Most Machine Learning models have been focused on cost prediction rather than duration. Also, data mining tasks primarily approached the problem as Classification instead of Regression, reducing its applicability in project control activities. Additionally, there is limited exploration of Time Series (TS) datasets for the duration of completion forecasting, and the analysis often lacks granularity, like at the Work Package level, which is crucial for proactive decision-making.

This research addresses these gaps by prioritizing the accuracy of project duration forecasting using Deep Learning algorithms. This study approaches this problem as a regression and handling time series datasets, enabling real-time monitoring of project duration forecasting. The proposed framework collects data related explicitly to time forecasting, and the analysis operates at the Work Package level, facilitating more detailed identification of potential causes of delays.

Chapter 3 Methodology for Project Duration at Completion Forecasting using Machine Learning

3.1 Introduction

Many completed construction projects tend to have poor performance related to time management. This chapter aims to identify the main factors that affect project duration performance, define relevant metadata, establish a Data Acquisition Model (DAM), and discuss the data required and collected from the industry. As a result, the dataset required for the forecasting model was obtained and used as input for the forecasting model, which is detailed in the following chapter.

To achieve this goal, two types of analyses were conducted: an analysis of current construction projects from a business organization perspective and a comprehensive project lifecycle analysis. Both studies provide a better understanding of the actual problem. A relational model was also created to manage data relevant to the project duration calculation. Moreover, a detailed analysis of factors influencing project duration was conducted based on completed projects. It enables the identification of entities, attributes, relationships, and cardinality, as well as the configuration of relevant metadata. The latter helps identify the numeric data required for subsequent Time Series Forecasting models.

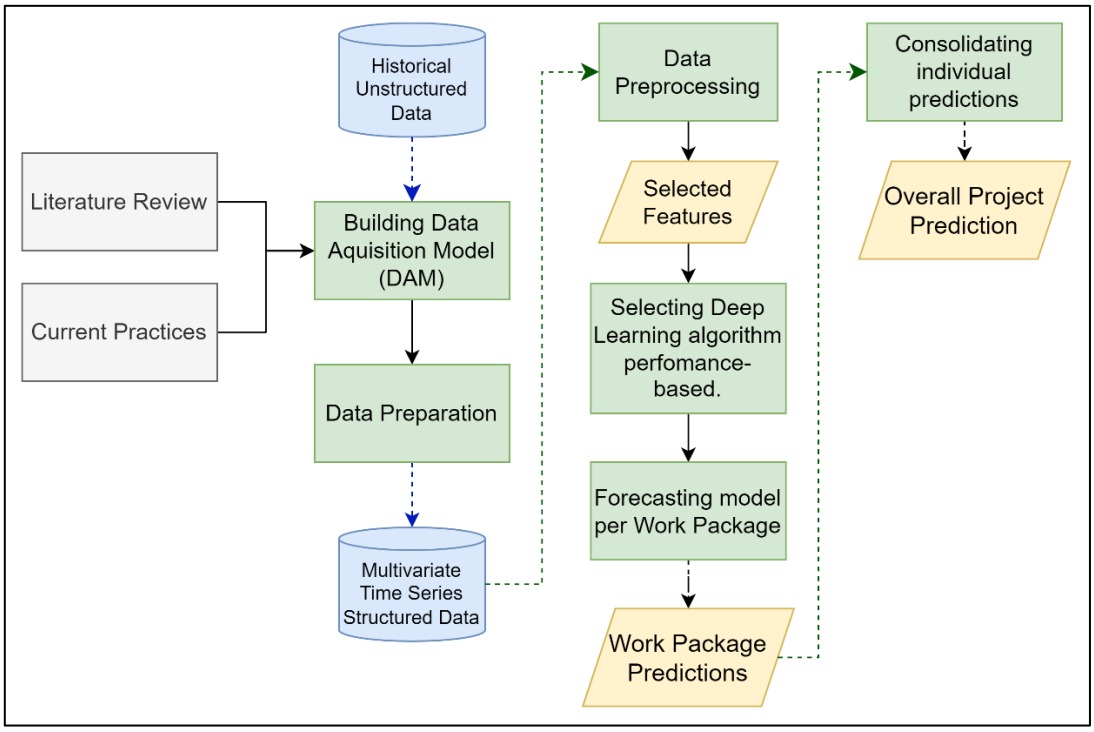


Figure 3.1 Methodology for Developing the Data Acquisition and Forecasting Models

3.2 Factors Influencing Project Duration Forecasting

3.2.1 Vertical and Horizontal Analysis for Construction Projects

Construction companies often adopt typical organizational structures to achieve their project goals, so it is crucial to understand these structures during the projects' execution phase to determine duration-influencing factors. To accomplish this, a vertical and horizontal analysis was conducted. The vertical analysis was approached from an operational management perspective, which provides a company's transversal view. An operational management view drives construction organizations toward their strategic goals while focusing on resources such as people, materials, equipment, practices, and management tools. The vertical analysis divides the construction organization into various levels, such as business units, portfolios, programs, projects, and work packages (*The Standard for Organizational Project Management (OPM)*, 2018). Likewise, other

departments, such as Accounting, Human Resources, Technology and Information, Logistics, and Legal, support them during the project execution phase. The usual operational breakdown structure for a construction organization is shown in Figure 3.2.

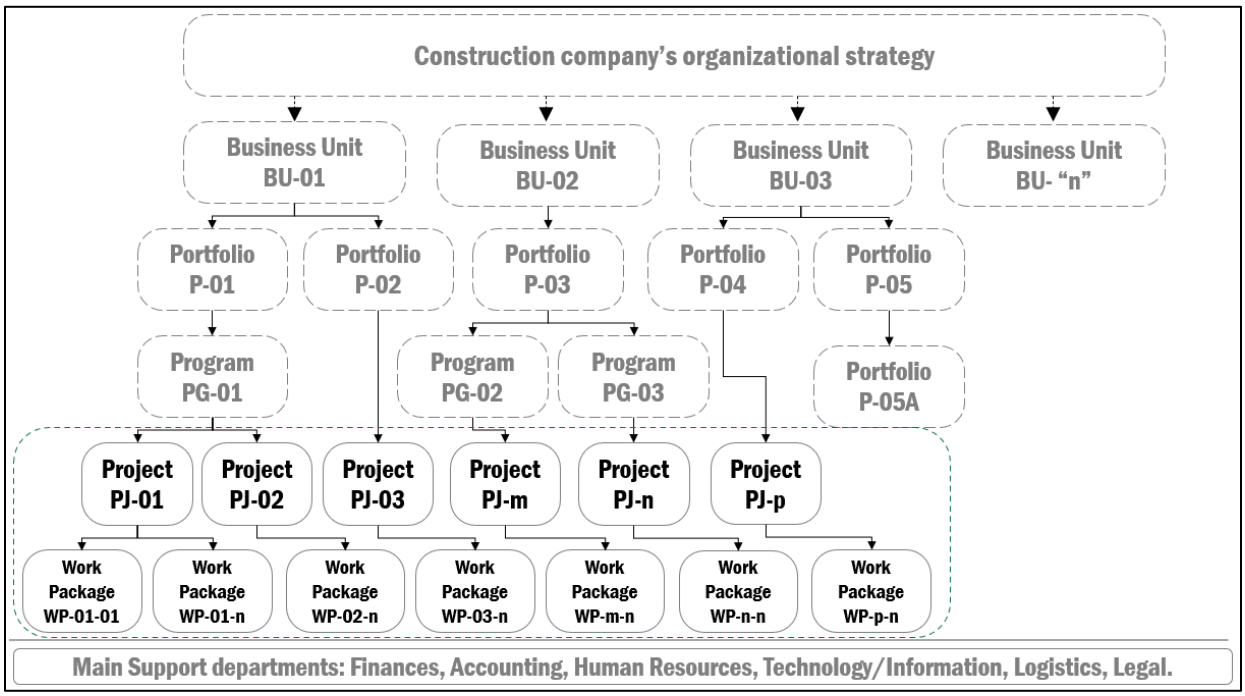


Figure 3.2 Schematic of Operational Breakdown Structure for a Construction Company

The Horizontal Analysis provides a long-term view of construction organizations when executing projects. It allows stakeholders to monitor technological adoption, timely resource allocation, regulatory changes over time, and more. These aspects are developed within the project lifecycle of a typical construction project. The project lifecycle comprises project phases, which may be set sequentially or iteratively, and overlapping among them is possible. The Construction Industry Institute (CII) has defined eight project phases: Feasibility, Concept, Detailed Scope, Detailed Design, Procurement, Construction, Commissioning & Start-up, and Handover & Closeout. Additionally, each project phase or the entire project life cycle has stages known as Process Groups, such as Initiating, Planning, Monitoring and Controlling, and Closing (Project

Management Institute, 2023). The interaction between project phases and process groups is shown in Figure 3.3.

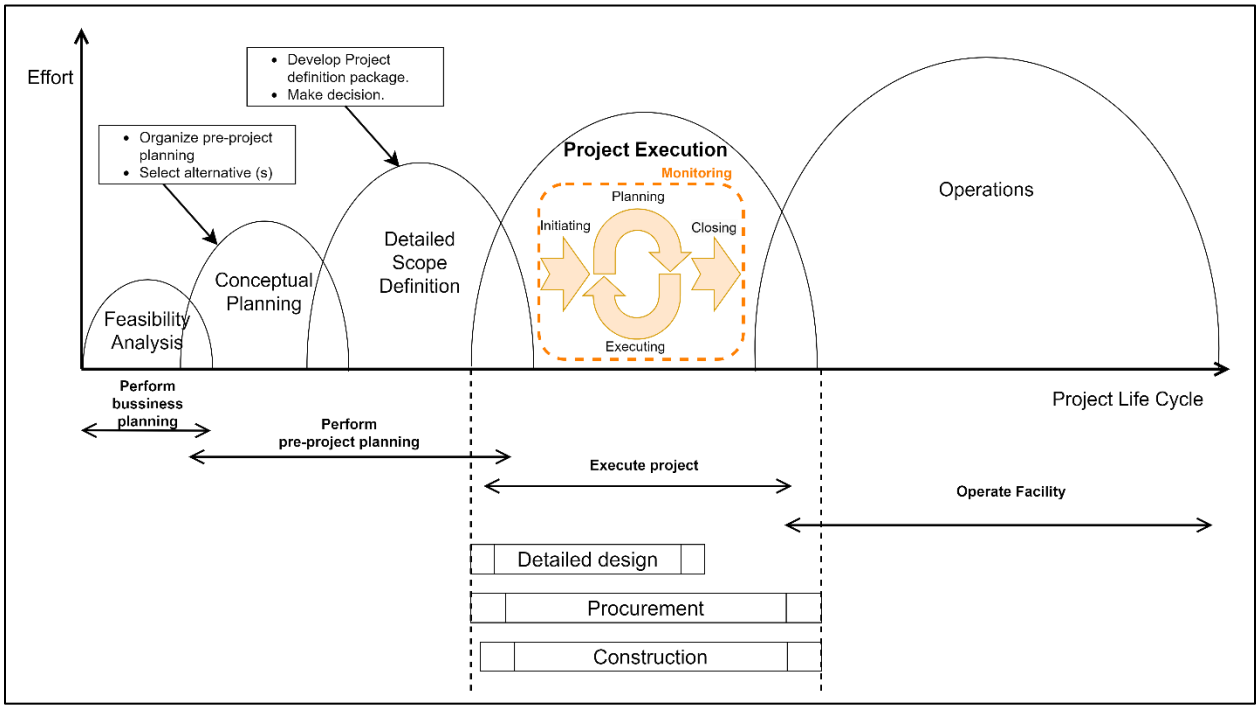


Figure 3.3 Interaction Between Project Lifecycle Phases and Process Groups for Project Execution, adapted from Construction Industry Institute (2019)

3.2.2 Current Practices in Project Duration Forecasting

This section describes the most common techniques used by practitioners in the context of the execution phase. It is also relevant to mention that any forecasting technique follows a sequence well compiled by the AACEi, referring to it as the forecasting map, which industry professionals mainly accept. The forecasting map is shown in Figure 3.4.

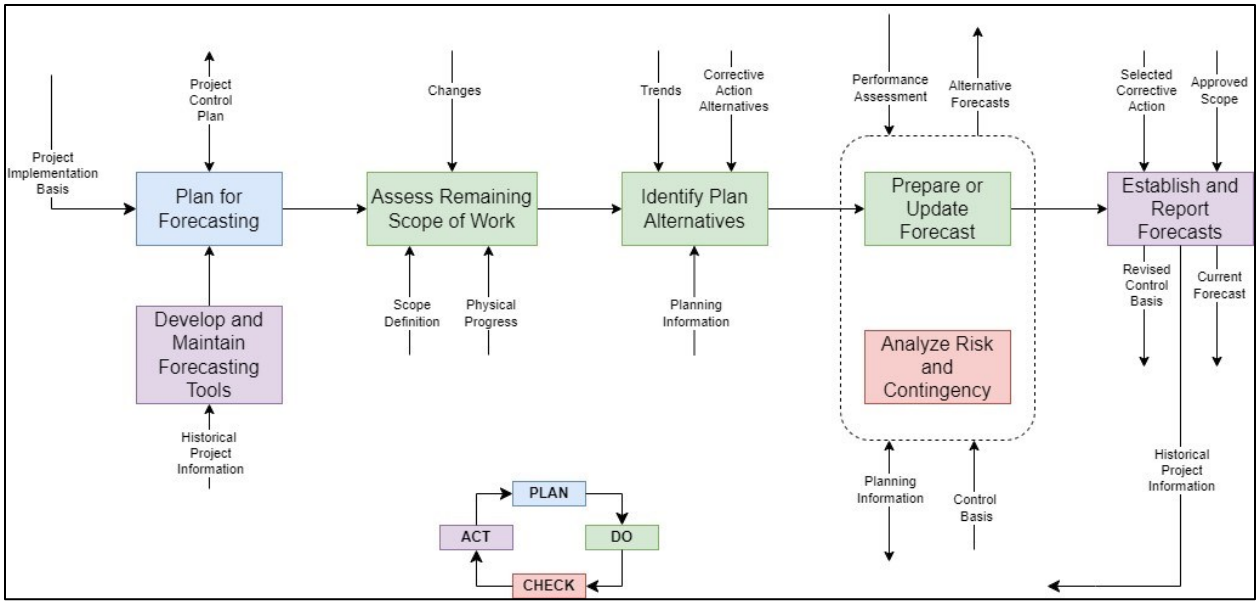


Figure 3.4 Forecasting Map on the Industry (Stephenson, 2015)

Today, the Monte Carlo Simulation is mentioned among the most applied techniques for project duration forecasting, typically used on scheduling network-based techniques like CPM. Similarly, the Earned Value Methodology (ESM) is often used by construction projects to forecast outcomes using progress performance metrics.

- **Overview of the Monte Carlo Simulation for Duration Forecasting**

Monte Carlo Simulation, named after the Monte Carlo Casino, is a probabilistic method that originated in the 1940s through the work of scientists Stanislaw Ulam and Nicholas Metropolis. Initially designed for solving mathematical problems using statistical sampling, it has evolved into a powerful tool for analyzing uncertainties in various fields, notably project management (Carlo, 2017; Sallabi, 2011). This technique leverages its probabilistic nature, considering a range of potential project durations and providing a distribution of results rather than a single deterministic forecast. This approach provides a more accurate depiction of uncertainty (Papadopoulos &

Yeung, 2001). Due to likely duration inputs, it facilitates scenario analysis, enabling project managers to assess potential outcomes and make informed decisions based on a range of scenarios.

Additionally, it contributes to comprehensive schedule risk assessment by addressing uncertainties in various project parameters, such as task durations, resource availability, and external factors. This holistic approach enhances the understanding of project schedule risks. Likewise, it's crucial to note that the effectiveness of Monte Carlo Simulation relies on detailed input data (Kroese et al., 2011); therefore, result accuracy yields on the quality of the data provided. Notable Montecarlo-based software options include Oracle Primavera Risk, Microsoft Project Risk Analysis, and @Risk by Palisade. While these tools offer various strengths, the choice depends on project complexity, user expertise, and organizational preferences. When using Monte Carlo simulation with the Critical Path Method (CPM) for project duration forecasting, the input variables are displayed in Table 3.1.

Table 3.1 Input Variables by Applying Montecarlo Simulation with CPM for Duration Prediction

Input variables	**Description**
Activity Duration Estimates	Use probabilistic distributions (e.g., normal, triangular) for each activity's duration.
Probability Distributions	Select appropriate distributions to model task duration uncertainties based on data or expertise.
Dependencies and Sequencing	Accurately model all task dependencies and logical relationships (finish-to-start, start-to-start).
Resource Allocation and Availability	Include constraints on resources (labor, equipment, materials) that affect task timings.
Risk Factors	Integrate potential risks that could impact task durations or sequencing.
Project Milestones and Critical Path	Define project milestones and dynamically identify the critical path for minimum duration analysis.

- **Overview of the Earned Value Management (EVM) for Duration Forecasting**

Earned Value Management (EVM) was initially developed in the 1960s by the U.S. Department of Defense as a financial analysis tool to track and manage project performance and progress (Dibert & Velez, 2006; Vertenten et al., 2009). It has become a standard industry practice for monitoring project costs and schedules. EVM combines project scope, price, and schedule measures to accurately describe project performance and progress (Khamooshi & Golafshani, 2014; Mayo-Alvarez et al., 2022). The method uses three key data points: Planned Value (PV), which is the budgeted cost of work scheduled; Earned Value (EV), which is the budgeted cost of work performed; and Actual Cost (AC), which is the actual cost incurred for the work performed. These metrics help assess project status and efficiency.

For predicting project duration, EVM incorporates the Schedule Performance Index (SPI) (Khamooshi & Abdi, 2017). SPI is computed by dividing EV by PV and reflects how closely the project is following its scheduled plan. An SPI of less than 1 suggests that the project is behind schedule. By using the SPI and project performance data, project managers can forecast the likely completion time and adjust schedules or resources as needed. A suggested calculation for the total project duration is dividing Planned Duration by the SPI (Iranmanesh et al., 2007). However, it should be noted that EVM's prediction reliability relies on the baseline plan, so any flaws in the baseline can mislead performance evaluations. Additionally, EVM can be less effective in the early stages of a project where EV and PV are too small to yield meaningful insights (Chen et al., 2016). Likewise, it does not directly account for the impact of resource allocation changes in its calculations, reducing its interactions with resource management.

Despite the mentioned drawbacks, EVM is recognized as one of the most effective project management tools for monitoring and forecasting project performance, particularly in industries

like construction, where projects are complex and multi-faceted. It is endorsed by various standards bodies, including the Project Management Institute (PMI), and is mandated for use in U.S. government contracts. When using EVM for duration prediction, input variables are detailed in Table 3.2.

Table 3.2 Input Variables by Applying EVM for Duration Prediction

Input variables	**Description**
Planned Value (PV)	Scheduled work value as per the baseline.
Earned Value (EV)	Value of the actual work completed.
Actual Cost (AC)	The real cost spent on the completed work.
Schedule Estimates	Original and ongoing estimates of task durations.
Performance Indices	Such as SPI (Schedule Performance Index) and CPI (Cost Performance Index), indicating current project status against the plan.

3.2.3 It Project Duration-Influencing Factors

Forecasting the project duration during the construction phase is fraught with complexities due to a wide range of factors that can influence it. Additionally, once factors are identified, they require proper categorization based on their origin (internal or external), impact scope (overall project or specific components), and temporal presence (specific stages or throughout the project). Overall, the factors analysis follows three phases: (1) identification, which analyzed extensive investigations related to duration-influencing factors and also, those factors sourced from current industry practices; (2) categorization, which is supported by the vertical and horizontal analysis previously performed and (3) quantification, which details calculation methods per factor to drive a scientific analysis and develop a forecasting model.

1. Factors identification.

Multiple studies have delved into the identification of duration factors in construction projects. The process was based on rigorous criteria, including publication types such as journals or books, recent publication years, and authors' professional or academic background, which were considered in the comprehensive analysis. Many studies that didn't meet those requirements were discarded, resulting in thirteen (13) primary studies spanning various global contexts. The table below depicts the factor selected.

Table 3.3 Duration Influencing Factors per Author

Factor	Kermanshachi et al. (2022)	Deep et al. (2022)	Durdyev et al. (2019)	Emam et al. (2015)	Hansen et al. (2023)	Hossain et al. (2022)	Merrow (2012)	Selcuk et al. (2022)	Sepasgozar et al. (2019)	Tafazzoli et al. (2017)	Tebeje (2016)	Wang et al (2022)	Zidane et al. (2018)	**Frequency**
Excessive change orders by the owner during construction	1	-	-	1	1	-	-	1	1	1	-	1	1	**8**
Financial difficulties of the owner	1	-	1	-	1	1	-	1	1	-	1	1	-	**8**
Incomplete/improper design	1	-	-	1	1	-	1	1	1	1	-	1	-	**8**
Ineffective communication among parties	1	-	1	-	1	1	1	-	1	1	1	1	1	**10**
Lack of owner's commitment	-	-	-	-	-	-	-	-	-	-	-	1	1	**2**
Lack of project stakeholders' experience/ qualification/competence	-	-	1	-	-	-	-	-	-	-	-	1	-	**2**
Late approval process of design documents by owner	-	-	-	1	-	-	-	-	-	1	-	-	-	**2**
Late design	1	-	-	1	-	1	-	-	-	1	-	-	-	**4**
Poor execution management on site (organizations)	1	1	-	-	1	-	-	-	-	-	1	-	1	**5**
Poor planning and scheduling of project	1	-	1	1	1	1	-	1	1	1	1	1	1	**11**
Realism of obligations	-	-	-	-	-	-	-	-	-	-	-	1	-	**1**
Slow decision-making by owner	1	-	-	1	1	-	-	-	-	1	1	-	1	**6**

Slow quality inspection process by owner	-	-	-	-	-	-	-	-	-	-	-	-	1	**1**
Unskilled Construction Project Management Team	-	1	-	-	1	1	-	-	1	-	-	-	-	**4**
Unskilled Engineering Project Management Team	1	-	-	-	-	-	1	-	-	-	-	-	-	**2**
Unskilled Procurement Project Management Team	1	-	-	-	-	-	-	-	-	-	-	-	-	**1**
Bureaucracy within project organizations	-	-	-	-	-	-	-	-	-	-	-	-	1	**1**
Risk identification process for execution	1	-	-	-	1	-	-	-	-	-	-	1	-	**3**
Challenges of the physical location	-	1	-	-	1	-	1	-	-	-	-	1	-	**4**
Poor site-office conditions	1	-	-	-	-	-	-	-	-	-	-	-	-	**1**
Project complexity	-	-	-	-	-	-	-	-	-	-	-	1	-	**1**
Project size	-	-	-	-	-	-	-	-	-	-	-	1	-	**1**
Technology availability	-	-	-	-	1	-	-	-	-	-	-	1	-	**2**
Weather/climate conditions	1	1	1	-	1	-	-	-	-	-	-	1	-	**5**
Delays in equipment procurement (shortage, delivery, quality)	-	-	1	-	1	-	-	1	-	-	1	-	1	**5**
Delivery of materials	1	1	-	1	1	1	-	1	1	-	1	-	-	**8**
Errors in contract documents	1	-	-	-	-	-	-	-	-	-	-	-	-	**1**
Quality of materials	-	-	-	-	-	1	-	1	-	-	-	-	-	**2**
Shortage of manpower (skilled, semi-skilled or unskilled)	1	-	-	-	1	-	1	1	-	-	-	-	1	**5**
Shortage of materials	-	-	1	-	1	1	-	-	-	-	1	-	-	**4**

The more significant ones were taken from this preliminary list of influencing factors by performing the median significance test on the frequency of occurrence (Field, 2024; Wheelan, 2014). Given the frequency values observed in Table 3.3, the median is 3.5. Thus, table 3.4 reflects the more significant factors with occurrence greater than 3.5. Additionally, many studies agree the Earned Value Methodology is an assessment of the project performance and progress in the scope, schedule, and cost aspects, helping to identify outcome deviations (Fayad et al., 2019; Khafri,

2018; Kostelyk, 2012; Priyo, 2021). Hence, it is considered in the influencing-factor list due to its influence on the project schedule performance.

Table 3.4 More Significant Factors (Median > 3.5 on Frequency of Occurrence)

ID	Factor	Description
F-1	Excessive change orders by the owner during construction	Additional work or modifications requested by the owner
F-2	Financial difficulties of the owner	Owner's inability to finance the project as planned
F-3	Incomplete/improper design	Design documents that are incorrect or incomplete
F-4	Ineffective communication among parties	Lack of clear, timely information exchange among stakeholders
F-5	Late design	Delivery of final design documents after the scheduled date
F-6	Poor execution management on site (organizations)	Inadequate management leading to inefficiencies on site
F-7	Poor planning and scheduling of project	Inaccurate project timelines and resource allocation
F-8	Slow decision-making by owner	Delays in making crucial project decisions
F-9	Unskilled Construction Project Management Team	Lack of necessary skills and expertise in the project team
F-10	Challenges of the physical location	Difficulties arising from the project's geographical or environmental conditions
F-11	Weather/climate conditions	Adverse weather affecting construction activities
F-12	Delays in equipment procurement (shortage, delivery, quality)	Late arrival of necessary equipment for construction
F-13	Delivery of materials	Late or incorrect delivery of construction materials
F-14	Shortage of manpower (skilled, semi-skilled or unskilled)	Insufficient or inadequately skilled labor force
F-15	Shortage of materials	Lack of necessary materials for construction
F-16	Project performance and progress	Performance and progress evaluation that encompasses cost, time, and scope.

2. Factors categorization.

The absence of a standardized categorization methodology leads to a lack of coherence and integrated analysis; therefore, a pressing need exists to systematically assess, pinpoint, and formulate a comprehensive assessment framework for the project duration-affecting factors. In the literature, this classification is focused on project delays instead of project duration-influencing

factors. For example, the delay classification proposed by Ansah and Sorooshian (2018), called the 4-P categorization, underscores internal project sources of delay such as Participants-related, Procurement-related, Project-related, and Practices-related which follows a typical construction project environment from an operational perspective.

Given this gap, the identified factors have been classified based on their level of impact, either work package or overall project, and the probable phase of occurrence according to the vertical and horizontal analysis previously performed. The level-of-impact analysis considered that if a duration-influencing factor has effects on more than one work package, it has effects on the overall project, for instance, weather or site conditions. Likewise, another category that was included is related to time dependency, which indicates whether the factor is susceptible to changes during the project execution timeline.

Table 3.5 Factors Categorization based on Level of Impact, Probable Phase and Time Dependency

ID	Factor	Factor description	Level	Level description	Probable phase	Probable phase description	Time dependent?
F-1	Excessive change orders by the owner during construction	Additional work or modifications requested by the owner	Project	Changes affect overall budget and timeline	Construction	Changes often result from unforeseen issues during construction.	Yes
F-2	Financial difficulties of the owner	Contractor's or Owner's inability to finance the project as planned	Project	Financial issues typically impact the entire project's financial health	Engineering & Construction	Financial difficulties can arise during planning or construction.	No
F-3	Incomplete/improper design	Design documents that are incorrect or incomplete	Work Package	Design issues usually pertain to specific components	Engineering	Design flaws are typically identified during the engineering phase, requiring revisions.	No
F-4	Ineffective communication among parties	Lack of clear, timely information exchange	Project	Communication affects all aspects of	All Phases	Effective communication plays a crucial role at every	No

		among stakeholders		project management		stage of the project's lifecycle.	
F-5	Late design	Delivery of final design documents after the scheduled date	Work Package	Design delays often affect specific parts of a project	Engineering	Design completion delays often occur in the engineering phase, affecting subsequent stages.	No
F-6	Poor execution management on site (organizations)	Inadequate management leading to inefficiencies on site	Work Package	Execution issues are often localized to specific tasks or areas	Construction	Execution management issues are most prevalent during the construction phase.	No
F-7	Poor planning and scheduling of project	Inaccurate project timelines and resource allocation	Project	Planning and scheduling affect the entire project timeline	Engineering & Procurement	Planning and scheduling flaws typically originate in the early phases, affecting procurement and execution.	Yes
F-8	Slow decision-making by owner	Delays in making crucial project decisions	Project	Affects pace and efficiency across all project stages	All Phases	Decision-making processes can slow down any project phase, from engineering to construction.	No
F-9	Unskilled Construction Project Management Team	Lack of necessary skills and expertise in the project team	Project	Management capability affects all project dimensions	All Phases	A skilled management team is critical throughout the project, from planning to execution.	No
F-10	Challenges of the physical location	Difficulties arising from the project's geographical or environmental conditions	Work Package	Location challenges are typically specific to site-related activities	Construction	Physical location challenges are most impactful during the construction phase.	Yes
F-11	Weather/climate conditions	Adverse weather affecting construction activities	Work Package	Weather impacts are often localized to outdoor activities	Construction	Weather conditions directly affect outdoor construction activities.	Yes
F-12	Delays in equipment procurement	Late arrival of necessary	Work Package	Equipment procurement usually affects	Procurement	Equipment delays are typically linked	Yes

	(shortage, delivery, quality)	equipment for construction		specific phases or tasks		to procurement challenges.	
F-13	Delivery of materials	Late or incorrect delivery of construction materials	Work Package	Material deliveries are typically linked to specific construction phases	Procurement & Construction	Material delivery issues can arise during procurement and directly affect the construction phase.	Yes
F-14	Shortage of manpower (skilled, semi-skilled or unskilled)	Insufficient or inadequately skilled labor force	Work Package	Manpower needs vary across different stages of the project	Construction	Manpower shortages are most acutely felt during the intensive labor demands of the construction phase.	Yes
F-15	Shortage of materials	Lack of necessary materials for construction	Work Package	Material shortages typically affect specific construction activities	Procurement & Construction	Material shortages can occur during procurement and have immediate impacts on construction activities.	Yes
F-16	Project performance and progress	Evaluating work package performance. Utilize EVM metrics to evaluate and predict.	Work Package	Changes affect overall budget and timeline at WP level	Construction	EVM is a continuous evaluation tool applicable throughout the project lifecycle for performance assessment, however it has relevant usage during the construction.	Yes

The present study is based on the analysis at the work package level during the construction phase using timeseries dataset to solve a regression problem; consequently, the listed factors should be filtered under those considerations. As a result, the duration-influencing factors are described in the table below.

Table 3.6 Duration-Influencing Factors Selected per Work Package, Construction Phase and Time Dependency

ID	Factor	Description
F-10	Challenges of the physical location	Geographical or environmental difficulties.
F-11	Weather/climate conditions	Adverse weather affecting activities.
F-13	Delivery of materials	Late or incorrect delivery of construction materials
F-14	Shortage of manpower	Insufficient or inadequately skilled labor.
F-15	Shortage of materials	Lack of necessary materials for construction
F-16	Project performance and progress	Evaluating work package health and future performance. Utilize EVM metrics to analyze and predict.

3. Factor quantification.

A factor perse is ambiguous, hindering a forecasting regression problem which claims numeric values by nature. By quantifying them, they can offer valuable insights into historical patterns, thereby contributing to more robust datasets and enhancing the accuracy of forecasts. Accordingly, a calculation method was implemented to facilitate the understanding of the identified duration-influencing factors. Thus, they were categorized by level of analysis with the corresponding calculation method, as it is shown in Table 3.7.

Table 3.7 Calculation Method per Duration-Influencing Factor

ID	**Factor**	**Calculation Method Description**
F-10	Challenges of the physical location	It is calculated based on **resource availability**, **suppliers' delivery times**, and **transportation time**, all of which are directly related to the project site's physical location. These metrics help assess the practical difficulties and constraints associated with the project's location.

F-11	Weather/climate conditions	It involves **historical weather data** and **weather monitoring stations**, which focus specifically on analyzing and forecasting weather patterns and conditions that could impact project activities. Thus, any potential hindrances to the project's duration are addressed.
F-13	Delivery of materials	Assess the percentage of **project time spent** (deviation) waiting for materials to arrive.
F-14	Shortage of manpower	It revolves around **resource allocation tracking** and **productivity metrics**, which help monitor the availability and efficiency of manpower resources. This factor assesses internal workforce-related challenges that could lead to delays or bottlenecks in project execution.
F-15	Shortage of materials	Measure the impact of material shortages on project schedule by comparing **planned versus actual** progress.
F-16	Project performance and progress	It is assessed through the EVM metrics, detailed below: Planned Value (PV) = (Planned Percentage of Completed Work) x (Budget at Completion) Earned Value (EV) = (Actual Percentage of Completed Work) x (Budget at Completion) Actual Cost (AC) = Total Costs Incurred for the Work Performed Schedule Variance (SV) = EV - PV Cost Variance (CV) = EV - AC Schedule Performance Index (SPI) = EV / PV Cost Performance Index (CPI) = EV / AC

3.3 Data Acquisition model (DAM) for forecasting project duration

Decision makers ultimately seek a unified and accurate representation of reality from their information systems. In many organizations, data and information are scattered across various sources. Consequently, when developing a singular forecasting model, the process involves collecting data from diverse sources and consolidating it into a cohesive, particular version. It ensures homogeneity among stakeholders, facilitating uniformity in information exchange, promoting coherence and informed decision-making.

The Data Acquisition Model (DAM) is constructed during a database application's analysis and design phases to ensure a comprehensive understanding and accurate capture of the requirements before creating the actual database (Biskup & Menzel, 2007). Beyond their primary purpose, DAM serves additional functions, including:

Grasping Business Dynamics: Data Acquisition modelling is crucial for understanding business processes like the construction project schedule management and forecasting before developing supporting applications.

Facilitating Team Understanding: Data Acquisition models serve as practical educational tools, conveying information visually at different levels of detail. Walking through data models is a valuable practice for quickly educating new team members on concepts and rules.

Subsequently, creating a DAM becomes crucial to collecting adequate information for the Prediction Model application. DAMs can be created in two main ways: Relational and dimensional. According to Hoberman (2015), Relational data modelling involves capturing a business's operational essence by precisely representing its rules. In contrast, Dimensional data modelling focuses on capturing how a business is monitored by precisely depicting aspects like navigation through data. A comparison table is shown below to encircle that.

Table 3.8 Comparison Between Relational and Dimensional Databases

Feature	**Relational Database**	**Dimensional Database**
Data Structure	Tables with rows and columns	Star or snowflake schema
Schema Design	Structured schema with predefined relationships	More flexible, denormalized schema for better query performance
Query Type	Suited for transactional processing (OLTP)	Optimized for analytical processing (OLAP)

Query Flexibility	Supports complex queries and transactions	Optimized for read-intensive operations, especially complex analytical queries
Operations	Inserts, updates, deletes, and complex transactions	Aggregations, summaries, and complex analytical queries
Data Integrity	Strong emphasis on data integrity through normalization	May sacrifice some aspects of ACID properties for improved query speed
Use Case Examples	Tracking tasks, team assignments, and project progress in real-time	Analyzing historical project data for performance trends, resource allocation, and risk assessment in business intelligence and reporting

3.3.1 The Relational Data Model for Project Duration forecasting.

The relational data model relates the project's intricacies and yielded data. In the context of the project duration forecasting, the project complexities can be operationally organized. The project duration forecasting during the project execution phase involves widely known project management levels such as portfolio, program, project, and work packages. These components have been organized in a Relational Data model to facilitate collecting, organizing, and periodically analyzing data tailored explicitly for project duration forecasting. Moreover, Relational Models excel in handling structured data, making them particularly well-suited for periodic data collection required for machine learning time series forecasting. Their robust transactional support guarantees accurate data storage and updates, reinforcing their reliability in dynamic and evolving time series datasets.

This database model type has also been selected to represent and interconnect various elements that influence project schedules, defining the relationship between entities and offering a clear understanding of dependencies and their impact on project duration. A prominent visual representation of a Relational Database is the Entity Relationship Diagram (ERD).

3.3.2 The Entity Relationship Diagram (ERD)

An Entity-Relationship Diagram (ERD) is a tool used in database design and systems analysis to model and understand the structure of information and how different entities interact (Bagui & Earp, 2011; Q. Li & Chen, 2009). Figure 3.5 displays the ERD for project duration forecasting using Chen's notation. Key components of an ERD are:

- **Entities:** Represent real-world objects or concepts, such as "Work Package", "Resources", or "Project Phase". Each entity is depicted as a rectangular box in the diagram.
- **Attributes:** They are the qualities associated with entities. They are illustrated within ovals and are connected to the respective entity. For example, a "Work Package" entity may include "WorkPackageID," "WorkPackageName," and "DueDate."
- **Relationships:** Illustrate the connections between entities. Lines connecting entities indicate the nature and type of association between them and they can have labels to describe the nature of the association, such as "manages," "assigned to," or "belongs to."
- **Cardinality:** It establishes the numerical relationship between entities in a given relationship, indicating the quantity of instances of one entity associated with another. Standard cardinality notations include "1" for one, "0…1" for zero or one, and "0…n" for zero to many. For instance, the relationship between "Work Package" and "Resource" could have a cardinality of "1...n," indicating that one work package can be assigned to multiple resources. Still, each resource is assigned to one work package at a time.

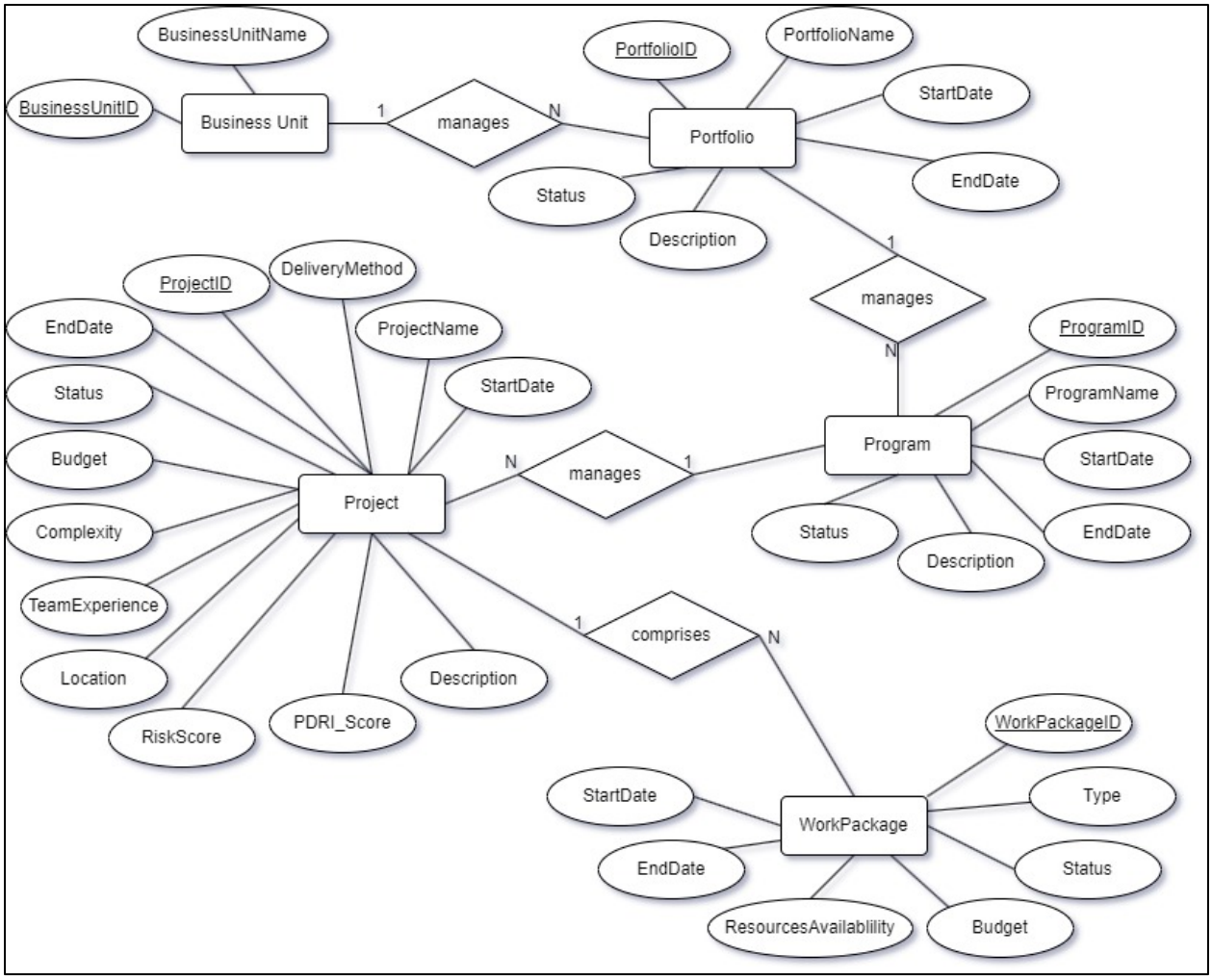


Figure 3.5 ERD for Project Duration Forecasting

The table below provides descriptions of the components in the ERD used for predicting project duration.

Table 3.9 Description of the ERD components for predicting project duration

Entity	Attribute	Description
Business Unit	BusinessUnitID	Company Identification Code assigned to Project's Business Unit
	BusinessUnitName	Company Identification Name of the Business Unit
Portfolio	PortfolioID	Company Identification Code assigned to Project's Portfolio
	PortfolioName	Company Identification Name of the Portfolio
	StartDate	Start Date of the Project's Portfolio. It involves planned and actual.
	EndDate	Finish Date of the Project's Portfolio. It involves planned and actual.
	Status	It contains the physical progress status and can be Non-Started, In-Progress, or Finished

	Description	A brief description of the portfolio containing benefits, challenges, major risks, and contribution to the organization.
Program	ProgramID	Company Identification Code assigned to Project's Program
	ProgramName	Company Identification Name of the Project's Program
	StartDate	Start Date of the Project's Program. It involves planned and actual.
	EndDate	Finish Date of the Project's Program. It involves planned and actual.
	Status	It contains the physical progress status and can be Non-Started, In-Progress, or Finished
	Description	A brief description of the program containing planned outcomes, goals, and contribution to the portfolio.
Project	ProjectID	Company Identification Code assigned to the Project
	ProjectName	Company Identification Name of the Project
	StartDate	Start Date of the Project. It involves planned and actual.
	EndDate	Finish Date of the Project. It involves planned and actual.
	Status	It contains the physical progress status and can be Non-Started, In-Progress, or Finished
	Description	A brief description of the project including expected outcomes, interaction with other project or phases.
	DeliveryMethod	It refers to the strategy used to plan, design, and execute the project. It includes Design-Bid-Build, Design-Build, Construction Management at Risk, and Integrated Project Delivery.
	Budget	It encircles the amount allocated to execute the project.
	Complexity	It refers to the outcome of analyzing the strategy adopted to execute the project which considers many subfactors as risk, resources, or legal restrictions. It can be low, medium, or high.
	TeamExperience	It is the experience of the Project Team in similar projects.
	Location	It contains the location remoteness, which is close related to resource availability, climate conditions, etc.
	RiskScore	It is the result of comprehensive Project Risk Assessment.
	PDRI_Score	PDRI stands for Project Definition Rating Index. It is a tool used in the field of project management and construction to assess the level of definition and completeness of a project during the early stages of planning. The PDRI score is a quantitative measure that helps project teams evaluate and improve project definition to reduce the likelihood of changes and problems during later stages of the project life cycle.
Work Package	WorkPackageID	Identification Code assigned to each Project Work Package.
	Type	It refers to nomenclature assigned to each Work Package.
	Status	It contains the physical progress status and can be Non-Started, In-Progress, or Finished
	Budget	It encircles the amount allocated to execute specific Work Package.
	ResourcesAvailability	Numeric representation about resources availability difficulties. It uses a Likert scale with low, medium, and high.
	StartDate	Start Date of the Work Package. It involves planned and actual.
	EndDate	Finish Date of the Work Package. It involves planned and actual.

3.4 Data Preparation for forecasting model

Data is the most valuable resource for the forecasting model, and it is closely related to outcomes quality and consistency (Carney et al., 2006; L. Yang et al., 2023). The data preprocessing process can start once the forecasting data needed within the project environment is set up through the ERD. This latter consists of taking the previous data collected and preparing it for data mining methods. Data Preprocessing is an iterative process subdivided into two sequential stages: Exploratory Data Analysis (EDA) and Dataset for Forecasting Model Improvement. The former is oriented to understanding historical data collected, and the latter to improve the forecasting model performance. EDA comprises data collection, data cleaning, and data inspection. While Data Collection aims to gather data from different sources (unstructured data) such as databases, spreadsheets, or repositories in an organized tabular structure, Data Cleaning is focused on tasks such as handling missing values by imputing new ones or deciding a strategy for handling them, removing duplicates, correcting errors, or handling outliers. After that, the data inspection struggles to understand the distribution, patterns, and relationships within the data when performing statistical analysis. The "Dataset for Forecasting Model Improvement" stage is explained in Chapter 4. The pipeline for the EDA process is shown in Figure 3.6.

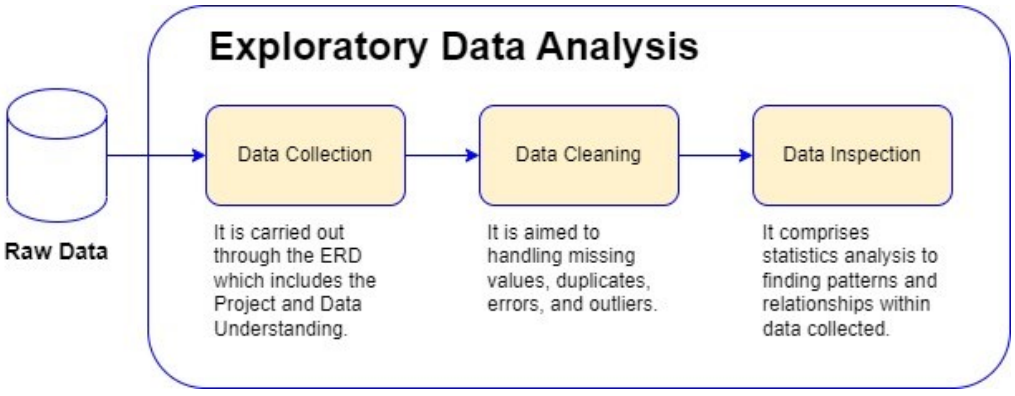


Figure 3.6 Pipeline of the Exploratory Data Analysis

3.4.1 Managing Historical Data Collected

The historical data is from a project portfolio that a contractor company managed. This company undertook twenty-two construction projects for a mining company located in a remote area in the high mountains of Peru between 2011 and 2012. The objective of these projects was to increase the mine production. They included demolishing old structures, constructing new facilities, and relocations to set up a new site layout. Each project was divided into five work packages: Concrete, Excavation, Backfill, Demolition, and Ground Mesh, to address different aspects of the project.

As discussed earlier, the duration-influencing factors needed for the project duration forecasting model were identified, as shown in Table 3.7. Accordingly, the raw data from these projects, such as project schedules, letters issued to the client, detailed budget and weekly three-week lookahead, was encountered from various channels, predominantly relying on MS Excel Spreadsheets and MS Project. Complementarily, Progress Weekly Reports in MS Word format and pertinent details from MS PowerPoint were crucial to cross-reference information. This compilation process faced challenges in extracting the most critical data on duration-influencing factors. Such data as "resource availability," "suppliers delivery time," "transportation time," "historical weather data," "weather monitoring stations data," "resource allocation tracking data," and "productivity metrics" were not possible to gather. Conversely, most data related to the project performance have been compiled. This data is associated with the project control, such as performance indicators or progress status. Also, this information was managed weekly at the work package level.

- **Historical Data Integration: from unstructured to structured data.**

All data was integrated in an MS Excel spreadsheet in a tabular way, considering its weekly time sequence per Work Package and Project. The following steps point out the integration process:

Step 1. Reviewing all available documents: *three-week lookahead, project budgets, Progress reports, project letters between contractor and client, project schedules, and project status presentations.*

Step 2. Gather data needed according to sources available:

- Unit Price per Work Package from the project planned budget.
- Weekly Actual Quantities from the Percentage of Activities Completed (PAC) weekly report.
- Weekly Percentage of Planned Cost incurred from the Project Cost-flow.
- Total Planned Quantities from the project planned budget.
- Actual and Planned Physical Progress from weekly project "S" Curve.
- Actual and Planned durations and start and finish dates from the weekly project schedule.

3.4.2 Data Cleaning

- **Missing values**

After initial compilation, notable missing values were raised in the novel integrated table. Each row containing zeros or erroneous values were removed or replaced. As it deals with a time series dataset, it is essential to know the timestamp sequence. In this sense, those values from the time series sequence were removed, and values within the sequence were replaced following just the previous row data behaviour. In addition, as Machine Learning models highly depend on the quantity of data, the Demolition and Ground Mesh work packages were removed due to less data available, leaving Concrete, Excavation and Backfill work packages.

- **Verifying consistencies of values**

Some attributes, like planned values, were contrasted by comparing the planned budget with weekly progress reports. Others, like the work packages' start and finish dates, were compared

with project letters, weekly progress reports, and the weekly work planned completed report. Also, the cumulative actual quantities against the actual physical progress weekly. Likewise, the planned duration.

3.4.3 Feature engineering

The creation of new attributes is a common practice in data preparation. New attributes were created based on the existing ones within the historical data available. Such attributes are the Earned Value starting from the Planned Unit Price and the Actual Quantity, the weekly Planned Value from the total planned quantity, and the weekly Percentage of Planned Cost. Also, cumulative values for the Actual and Planned Quantities were inserted, cumulating the Earned and Planned Value. Additionally, the planned physical progress from the planned quantities. Subsequently, the Schedule Performance Index (SPI) and Schedule Variance (SV) since Planned and Earned Values are known, were included due to their close relationship with the project duration.

Moreover, after analyzing the gathered data, additional factors that are closely related to project time performance were also considered. Research has shown that Earned Schedule Management (ESM) is the most reliable method to forecast project schedules compared to other Earned Value-based approaches (Vandevoorde & Vanhoucke, 2006). Thus, ESM metrics were introduced. It is a deterministic method that emerged as an extension of Earned Value Management (EVM) and focuses explicitly on schedule performance. The concept of ESM was introduced by Walter Lipke in the early 21st century (Cho & Lim, 2020). Unlike EVM, ESM aims to provide a more accurate representation of project schedule performance. ESM metrics included are Earned Schedule (ES), Time Performance Index (TPI), and Time Variance (TV). ES is derived from Planned Value (PV) and Earned Value (EV). Furthermore, it interacts with the actual project time elapsed to yield TPI

and TV through ratio and difference calculations, respectively. The Figure below represents the ES graphically.

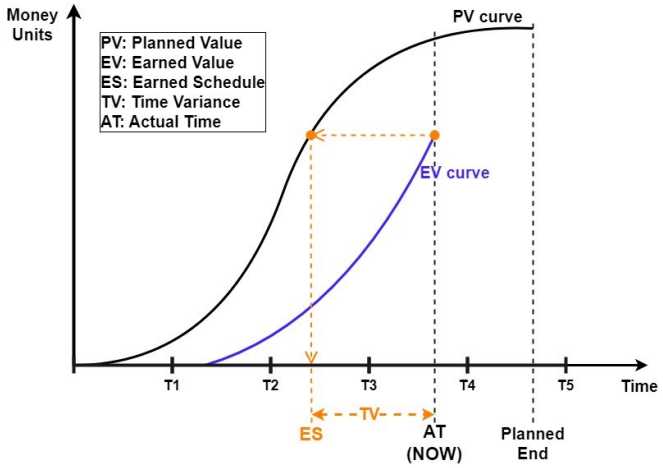


Figure 3.7 Graphical Representation of Earned Schedule

3.4.4 Data Inspection on Selected Attributes

The data inspection plays an essential role in the entire Data Mining pipeline (Augenstein et al., 2019; Chekanov, 2016). It enables (1) identifying data quality issues by uncovering outliers, duplications, or inconsistencies, (2) understanding data characteristics by providing patterns and particular distributions, (3) ensuring data relevance by verifying their potential impact in the forecasting of Duration to Complete DTC (target variable) and, (4) therefore, choosing appropriate mining techniques by enabling match suitable algorithms with the dataset. In this sense, a description of each attribute collected is explained. As this research is handling a time series dataset, it is essential to identify patterns along the timeline.

- **Planned and Actual Quantities:**

The Planned Quantities (PQ) encompasses the initial estimations of materials on the work packages such as concrete in cubic meters, excavation in cubic meters, backfill in cubic meters, existing structure demolition in cubic meters and ground mesh in square meters required for the

construction project. It is relevant for forecasting duration because accurate planning of quantities is crucial for determining the project's timeline and ensuring resource availability. The Actual Quantities (AQ) represent the real, executed quantities of resources during the construction project. It aids in understanding the actual resource consumption, informing future planning, and forecasting potential budget variations. Comparing actual quantities to planned quantities helps identify variances and adjust the project duration accordingly. Figure 3.8 (left side) shows the Box-and-Whisker plot for Planned and Actual Quantities, indicating that the Actual Quantities dataset exhibits more outliers and a larger spread, indicating higher variability than the Planned Quantities.

- **Planned Value (PV)**

PV is the estimated value of the work planned to be completed at a specific time. Serves as a foundational benchmark for cost forecasting. Project managers can use PV to gauge whether the project is on track regarding planned cost expenditure. It requires frequent re-evaluation and adjustment in response to project changes rather than being a static figure. Figure 3.8 (right side) shows fewer outliers in this plot, suggesting that the Planned Value data are more tightly clustered around the median.

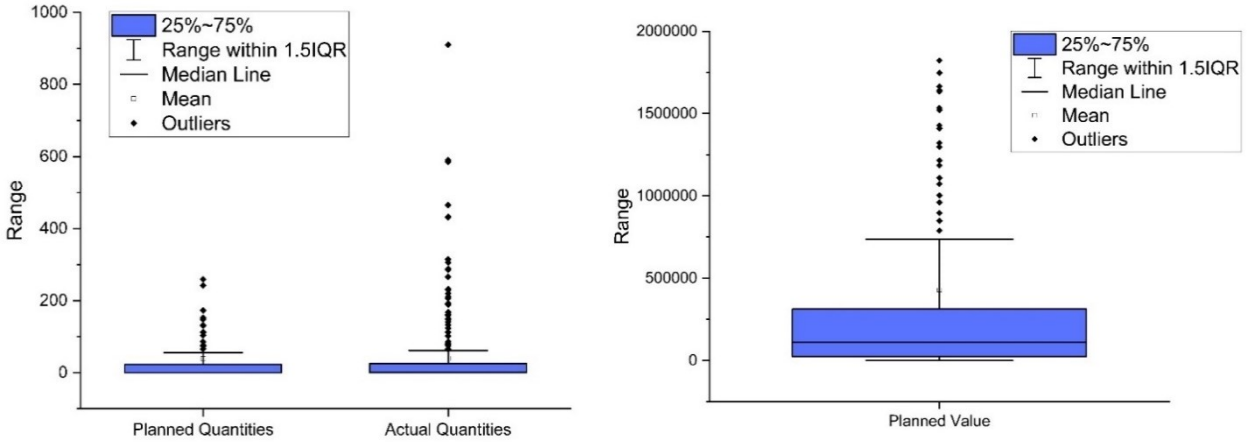


Figure 3.8 Box-and-Whisker Plot for Planned And Actual Quantities and Planned Value

- **Earned Value (EV)**

EV represents the value of the work performed and completed at a specific point in time. Provides a tangible measure of actual progress. Forecasting with EV assists in understanding how efficiently resources are being utilized and aids in predicting future cost and schedule trends. Figure 3.9 (left side) shows that the Earned Value data has significant variability, as indicated by the large number of outliers and the extensive range of the whiskers.

- **Earned Schedule (ES)**

ES is the time-phased measure of the value of work performed. Offers a nuanced time-based perspective for earned value. On the project duration forecasting, the ES shows project managers schedule variations and potential delays based on earned value achieved over time. By comparing the Box-and-Whisker plot (Figure 3.9 – right side) with the EV, the data points are notably less spread out, implying a tighter distribution with less variability in the Earned Schedule metrics.

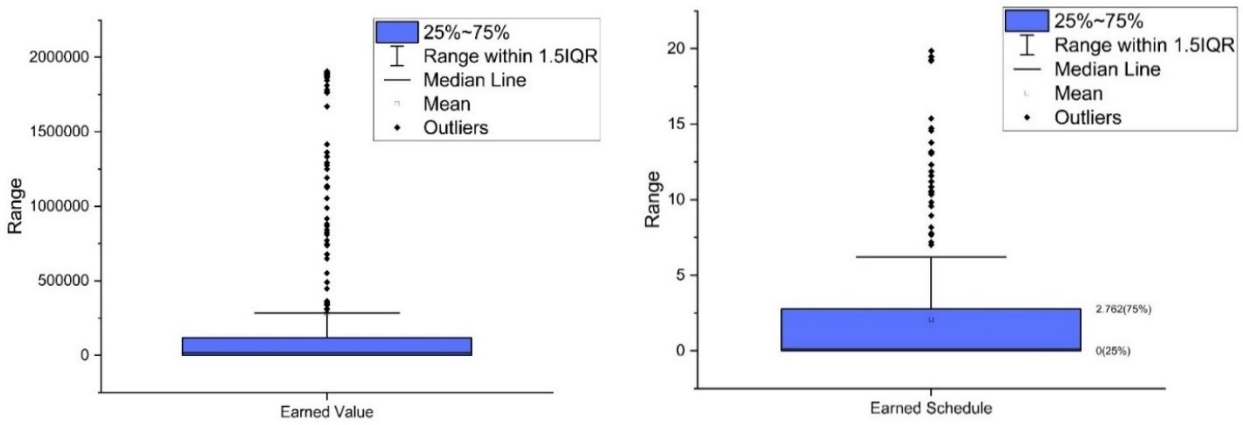


Figure 3.9 Box-and-Whisker Plot for Earned Value and Earned Schedule

- **Time Performance Index (TPI)**

TPI is a measure of schedule efficiency, calculated as the ratio of earned schedule to actual time spent. Assists in forecasting the efficiency of the project in terms of time. A TPI greater than 1 indicates efficient progress, while a TPI less than one may signal potential delays, prompting

proactive adjustments to the schedule. From Figure 3.10 (left side), it is evidenced that the data shows that the mean is slightly higher than the median, which may indicate a slight positive skew. Outliers are depicted as individual points located outside the whiskers, signifying that these values are unusual compared to the rest of the data. Additionally, the distribution of data points, especially with outliers on the higher side, suggests that there are instances where the Time Performance is significantly above what is typical for the dataset.

- **Time Variance (TV)**

TV represents the difference between the earned schedule and actual time spent. Identifying time deviations allows project managers to forecast potential delays or accelerations. Forecasting with TV provides insights into whether the project will likely meet or exceed its time objectives. Similarly, from Figure 3.10 (right side), the dataset has a relatively symmetrical distribution around its center, as the mean is very close to the median. However, some outliers below the lower whisker represent significant negative time variance. Moreover, the range of TV is considerably more comprehensive than that of TPI, with many large negative numbers present. It indicates that there were instances of substantial delays or deviations from the plan.

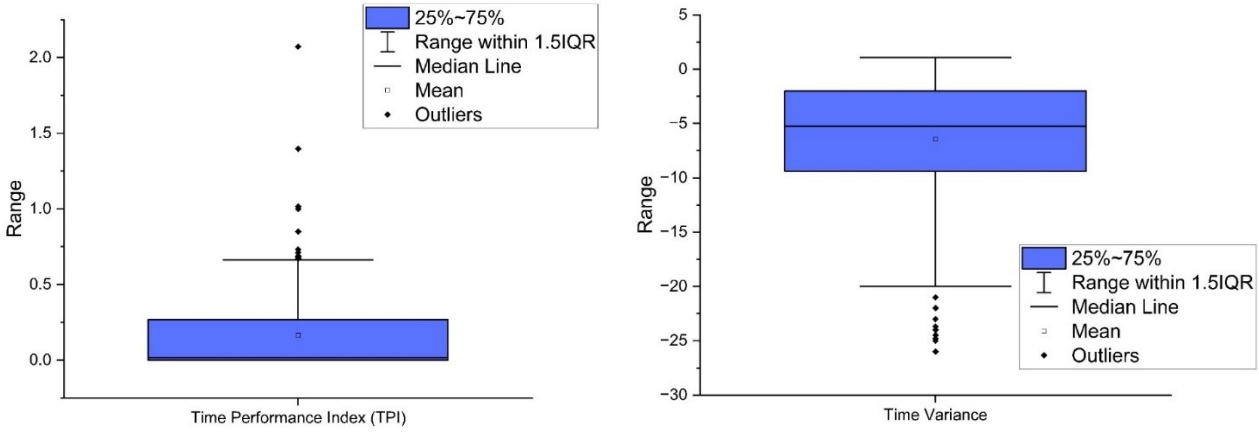


Figure 3.10 Box-and-Whisker Plot for Time Performance Index and Time Variance

- **Schedule Performance Index (SPI)**

SPI quantifies the schedule efficiency and is computed as the ratio of earned value to planned value. SPI assesses how well the project adheres to the planned schedule. Forecasting with SPI helps project managers anticipate future schedule trends, enabling timely adjustments to optimize project performance. Figure 3.11 (left side) indicates that the mean value is negatively skewed, lying below the median value. Additionally, the figure depicts outliers above the upper whisker, which suggests that there are instances of SPI values significantly higher than the average.

- **Schedule Variance (SV)**

SV offers a direct measure of whether the project is ahead or behind schedule at a specific point. It is calculated subtracting the planned value from the earned value. Forecasting with SV helps project managers anticipate schedule variations and make informed adjustments to keep the project on track. From Figure 3.11 (right side), the median and mean values of the dataset are very similar, indicating that the data is evenly distributed around the central value. Additionally, the SV plot has a much more comprehensive range when compared to the SPI. It suggests a significant number of data points with large negative values, indicating some deviations from the planned schedule.

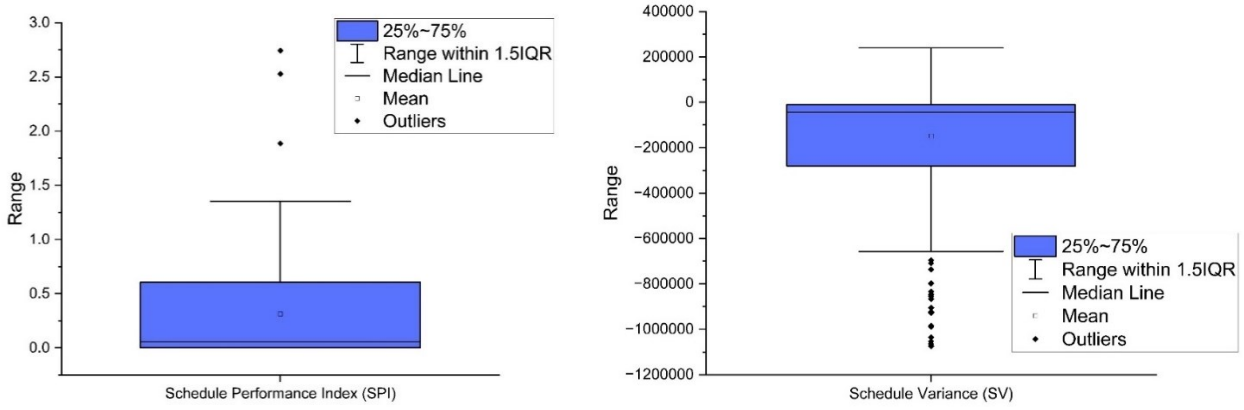


Figure 3.11 Box-and-Whisker Plot for Schedule Performance Index and Schedule Variance

- **Planned and Actual Physical Progress**

The planned physical progress encompasses the initial physical progress, including milestones and expected completion percentages. A critical metric for forecasting overall project progress. On the other hand, the actual physical progress represents the real, executed physical progress achieved during the construction project. Forecasting with actual physical progress aids in understanding the project's current state. It allows project managers to adjust plans based on observed progress, facilitating accurate duration forecasting and timely decision-making. Comparing planned and actual physical progress helps project managers anticipate potential delays and ensure alignment with the planned timeline.

From the chart below (Figure 3.12), numerous outliers are shown below the lower whisker, indicating significantly lower planned progress for the Planned Physical Progress. In contrast, the plot for Actual Physical Progress shows consistent actual progress data with no significant anomalies.

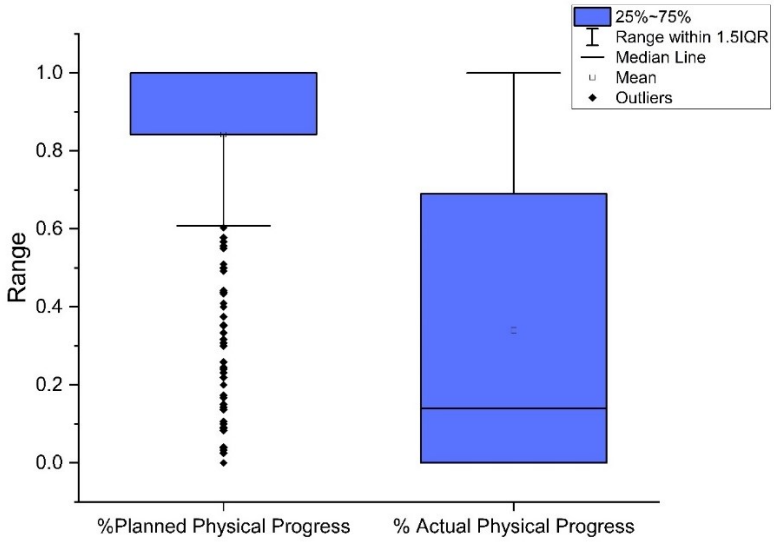


Figure 3.12 Box-and-Whisker Plot for Planned Physical (%) and Actual Physical Progress (%)

Chapter 4 The Deep Learning Forecasting Modelling for Project Duration and the Graphical User Interface

4.1 Introduction

After analyzing the historical data, this chapter elaborates on the project duration forecasting model using Deep Learning and produces a Graphical User Interface (GUI) for its usage and application. A high-level pipeline is shown in Figure 4.1. It comprises data preprocessing and the forecasting model itself. The preprocessing involves feature selection to address multicollinearity by applying the variance inflation factor (VIF) and correlation model and data splitting and normalization to address the model performance and data patterns. Then, the forecasting model considers managing the algorithm hyperparameters, which highly depend on the data characteristics and influence the model performance.

After getting individual duration predictions at work package levels, they are integrated into a methodology to calculate the overall project duration at completion prediction, using the CPM and PDM methodologies. In turn, this forecasting model is the core of the GUI, which, due to the deep learning algorithm's complexity, the user interface plays a pivotal role for non-expert users. Additionally, this chapter described complementary, relevant subprocesses, such as data augmentation, which addresses eventual small time series datasets, and present the performance metrics to monitor the fittest accuracy of deep learning algorithms by handling time series datasets.

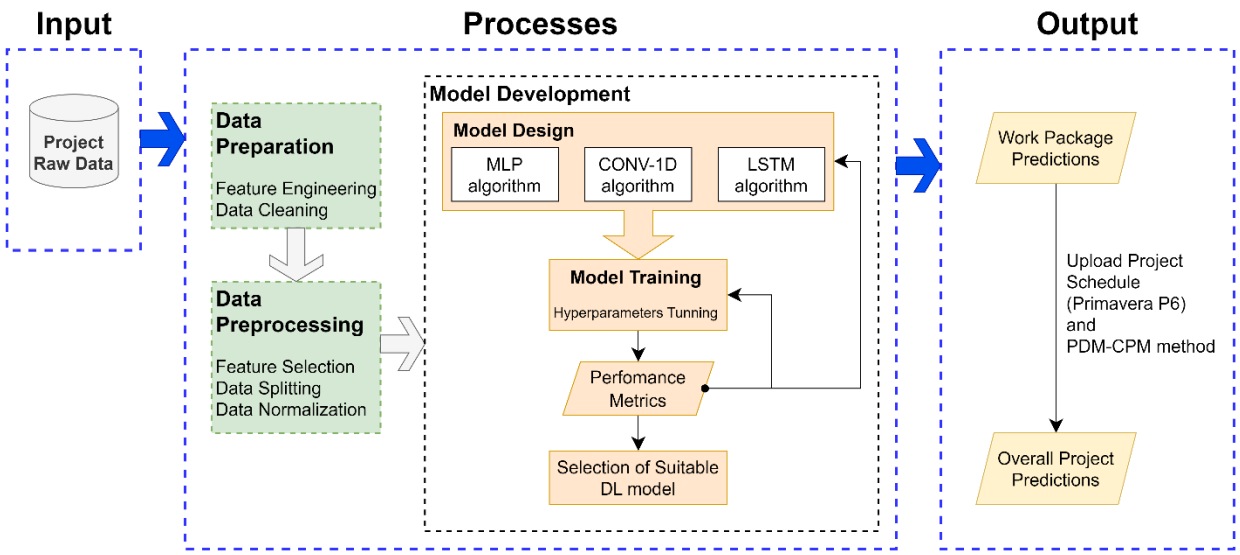


Figure 4.1 Development of Forecasting Model

4.2 Dataset Setup for ML Forecasting model.

The axiom 'More data equates to better model forecasting performance' holds in the context of machine learning (Lara-Benítez et al., 2021; Passalis et al., 2020; Torres et al., 2020). However, the first challenge lies in organizing available data to optimize the learning process by deep learning algorithms. A methodical approach is crucial when handling data from multiple projects, further divided into Work Packages (WPs). In this context, three possible scenarios for data setups to augment machine learning efficacy were experimented with, as described below. A graphical representation is shown in Figure 4.2.

a. Isolated Work Package Analysis:

This method involves analyzing data separately for each WP within a project. Contrary to machine learning's preference for large datasets, this compartmentalized approach limits the chronological data points, which hinders the algorithm's learning capacity. Furthermore, repeated execution of

the machine learning algorithm for each WP can lead to increased computational time and resource consumption.

b. Sequential Work Package Data Integration:

This strategy exploits the increased volume of data while preserving individual work package (WP) characteristics by grouping sequentially the same work packages from multiple projects, for example, grouping the work packages 'Concrete' from projects A, B, C, etc. likewise for the rest of work packages. In turn, it becomes more efficient when leveraging the autocorrelation inherent in Time Series Datasets. Autocorrelation is a statistical measure describing the degree of likeness between a specified time series and a lagged version over successive interval. It's used to identify repeating patterns or dependencies in data over time. Overall, this second setup allows for project-specific forecasting through separated algorithms for each WP, potentially leading to more targeted insights.

c. Aggregated Project Data Without WP Segregation:

This approach consolidates all data across projects without distinguishing between WPs. Although it maximizes the dataset size, it may not facilitate the identification of patterns specific to individual WPs due to the homogenization of data.

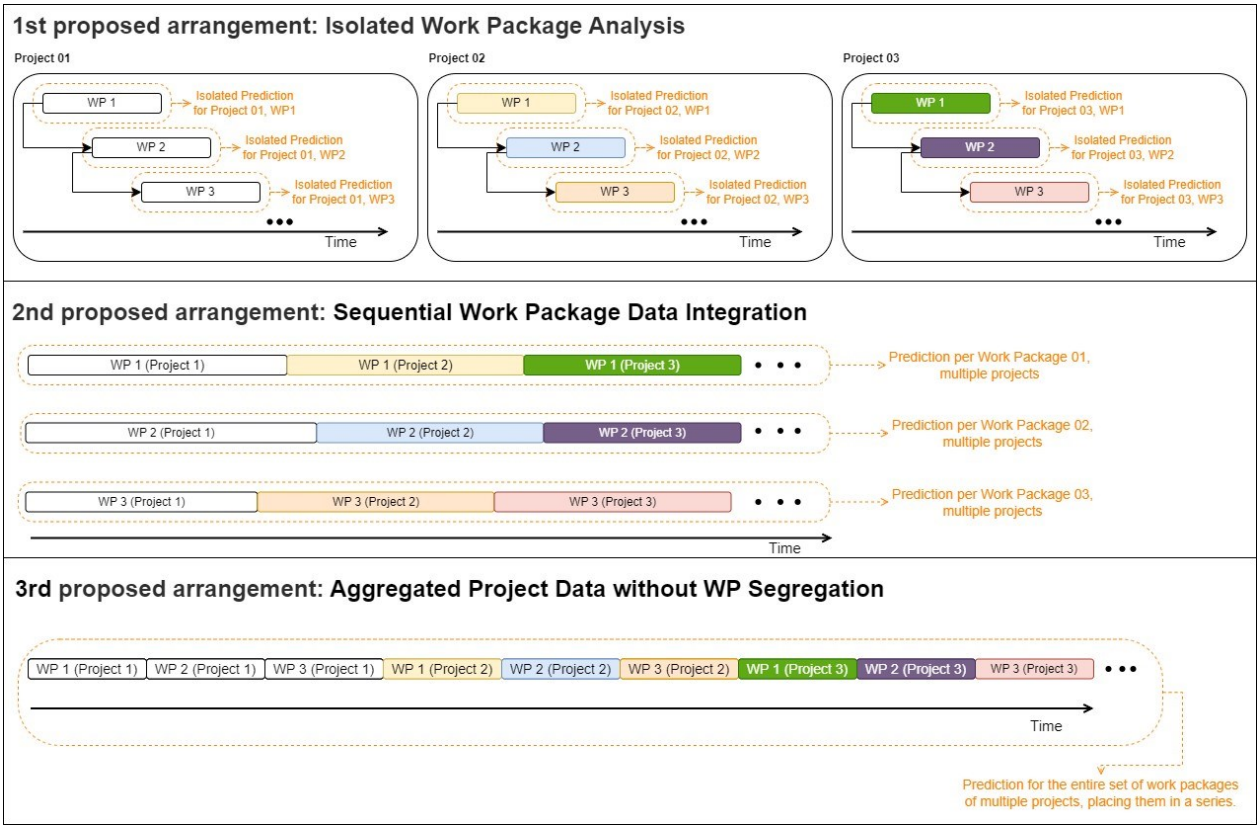
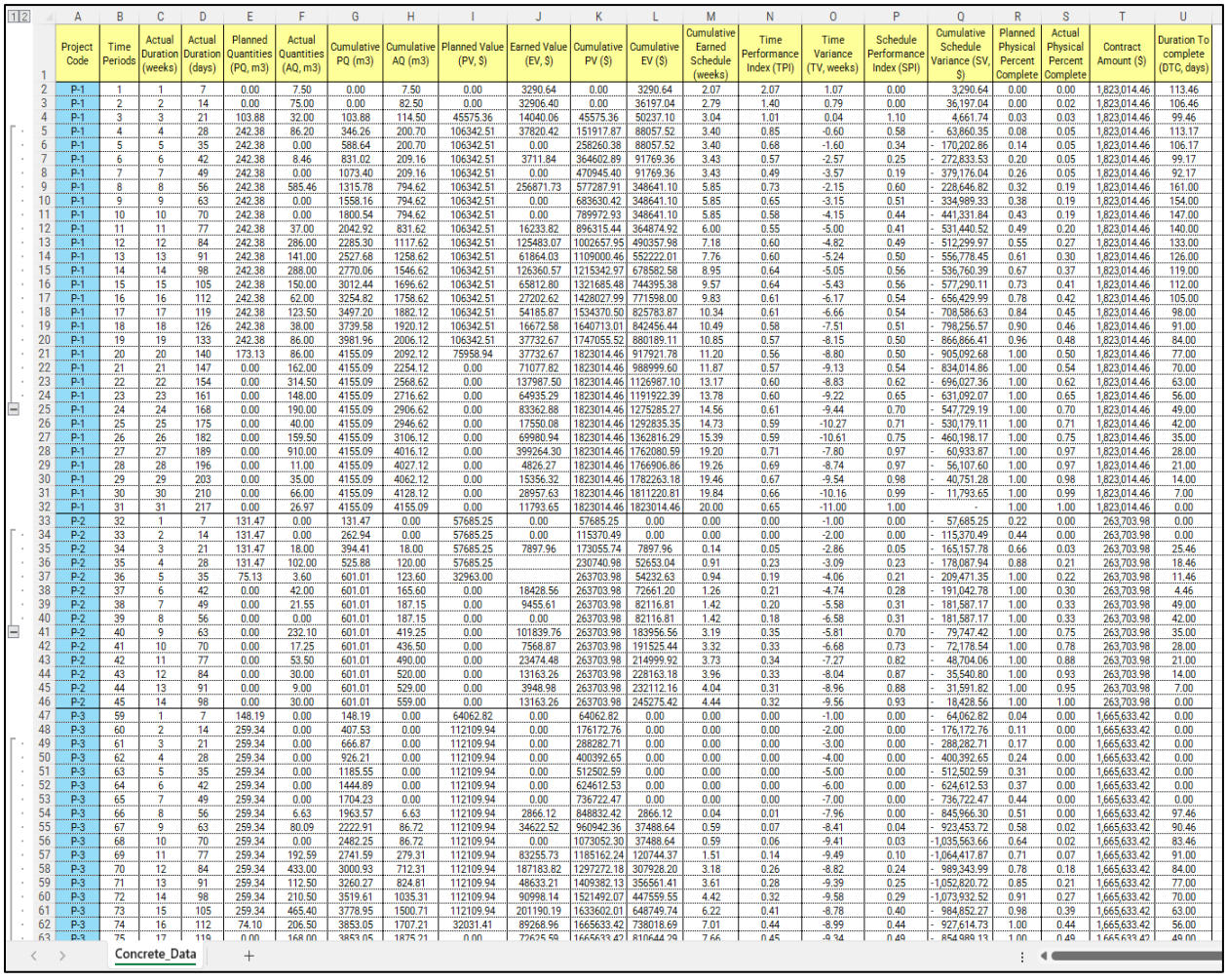


Figure 4.2 Three Time Series Data Setups to Improve Machine Learning Performance

4.3 Data Preprocessing

While data preprocessing is a broader step initiated in Chapter 3 by handling the raw data, additional refinements are essential for enhancing the forecasting model performance, especially when handling time series data for a regression approach. In this sense, the feature selection arises to fit the best inputs regarding forecasting performance accuracy. Then, a meticulous treatment of new missing values through strategies like time-based imputation or interpolation to ensure consistency with the temporal nature of the data. After that, normalizing or scaling the data is critical, with techniques like Min-Max scaling or Z-score normalization, to maintain uniformity in variable scales. Furthermore, creating lag features and utilizing rolling window statistics are crucial to capturing time series data's inherent temporal dynamics and dependencies. Preserving

sequential relationships between consecutive data points is paramount, necessitating careful consideration when structuring the dataset in a tabular format. Moreover, data transformation, such as differencing or any mathematical operation, might stabilize trends and seasonality in the multivariate time series dataset. Once Chapter 3 enabled statistical inspection of the collected data, they were assessed through the data preprocessing process. Figure 4.3 shows an example of the Concrete work package dataset after such a process. This Figure depicts the dataset in a structured way, with potential predictors and target variables placed as headers and the 'Concrete' work package from multiple projects placed as rows sequentially downwards.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Project Code	Time Periods	Actual Duration (weeks)	Actual Duration (days)	Planned Quantities (PQ, m3)	Actual Quantities (AQ, m3)	Cumulative PQ (m3)	Cumulative AQ (m3)	Planned Value (PV, $)	Earned Value (EV, $)	Cumulative PV ($)	Cumulative EV ($)	Cumulative Earned Schedule (weeks)	Time Performance Index (TPI)	Time Variance (TV, weeks)	Schedule Performance Index (SPI)	Cumulative Schedule Variance (SV, $)	Planned Physical Percent Complete	Actual Physical Percent Complete	Contract Amount ($)	Duration To complete (DTC, days)
2	P-1	1	1	7	0.00	7.50	0.00	7.50	0.00	3290.64	0.00	3290.64	2.07	2.07	1.07	0.00	3,290.64	0.00	0.00	1,823,014.46	113.46
3	P-1	2	2	14	0.00	75.00	0.00	82.50	0.00	32906.40	0.00	36197.04	2.79	1.40	0.79	0.00	36,197.04	0.00	0.02	1,823,014.46	106.46
4	P-1	3	3	21	103.88	32.00	103.88	114.50	45575.36	14040.06	45575.36	50237.10	3.04	1.01	0.04	1.10	4,661.74	0.03	0.03	1,823,014.46	99.46
5	P-1	4	4	28	242.38	86.20	346.26	200.70	106342.51	37820.42	151917.87	88057.52	3.40	0.85	-0.60	0.58	- 63,860.35	0.08	0.05	1,823,014.46	113.17
6	P-1	5	5	35	242.38	0.00	588.64	200.70	106342.51	0.00	258260.38	88057.52	3.40	0.68	-1.60	0.34	- 170,202.86	0.14	0.05	1,823,014.46	106.17
7	P-1	6	6	42	242.38	8.46	831.02	209.16	106342.51	3711.84	364602.89	91769.36	3.43	0.57	-2.57	0.25	- 272,833.53	0.20	0.05	1,823,014.46	99.17
8	P-1	7	7	49	242.38	0.00	1073.40	209.16	106342.51	0.00	470945.40	91769.36	3.43	0.49	-3.57	0.19	- 379,176.04	0.26	0.05	1,823,014.46	92.17
9	P-1	8	8	56	242.38	585.46	1315.78	794.62	106342.51	256871.73	577287.91	348641.10	5.85	0.73	-2.15	0.60	- 228,646.82	0.32	0.19	1,823,014.46	161.00
10	P-1	9	9	63	242.38	0.00	1558.16	794.62	106342.51	0.00	683630.42	348641.10	5.85	0.65	-3.15	0.51	- 334,989.33	0.38	0.19	1,823,014.46	154.00
11	P-1	10	10	70	242.38	0.00	1800.54	794.62	106342.51	0.00	789972.93	348641.10	5.85	0.58	-4.15	0.44	- 441,331.84	0.43	0.19	1,823,014.46	147.00
12	P-1	11	11	77	242.38	37.00	2042.92	831.62	106342.51	16233.82	896315.44	364874.92	6.00	0.55	-5.00	0.41	- 531,440.52	0.49	0.20	1,823,014.46	140.00
13	P-1	12	12	84	242.38	286.00	2285.30	1117.62	106342.51	125483.07	1002657.95	490357.98	7.18	0.60	-4.82	0.49	- 512,299.97	0.55	0.27	1,823,014.46	133.00
14	P-1	13	13	91	242.38	141.00	2527.68	1258.62	106342.51	61864.03	1109000.46	552222.01	7.76	0.60	-5.24	0.50	- 556,778.45	0.61	0.30	1,823,014.46	126.00
15	P-1	14	14	98	242.38	288.00	2770.06	1546.62	106342.51	126360.57	1215342.97	678582.58	8.95	0.64	-5.05	0.56	- 536,760.39	0.67	0.37	1,823,014.46	119.00
16	P-1	15	15	105	242.38	150.00	3012.44	1696.62	106342.51	65812.80	1321685.48	744395.38	9.57	0.64	-5.43	0.56	- 577,290.11	0.73	0.41	1,823,014.46	112.00
17	P-1	16	16	112	242.38	62.00	3254.82	1758.62	106342.51	27202.62	1428027.99	771598.00	9.83	0.61	-6.17	0.54	- 656,429.99	0.78	0.42	1,823,014.46	105.00
18	P-1	17	17	119	242.38	123.50	3497.20	1882.12	106342.51	54185.87	1534370.50	825783.87	10.34	0.61	-6.66	0.54	- 708,586.63	0.84	0.45	1,823,014.46	98.00
19	P-1	18	18	126	242.38	38.00	3739.58	1920.12	106342.51	16672.58	1640713.01	842456.44	10.49	0.58	-7.51	0.51	- 798,256.57	0.90	0.46	1,823,014.46	91.00
20	P-1	19	19	133	242.38	86.00	3981.96	2006.12	106342.51	37732.67	1747055.52	880189.11	10.85	0.57	-8.15	0.50	- 866,866.41	0.96	0.48	1,823,014.46	84.00
21	P-1	20	20	140	173.13	86.00	4155.09	2092.12	75958.94	37732.67	1823014.46	917921.78	11.20	0.56	-8.80	0.50	- 905,092.68	1.00	0.50	1,823,014.46	77.00
22	P-1	21	21	147	0.00	162.00	4155.09	2254.12	0.00	71077.82	1823014.46	988999.60	11.87	0.57	-9.13	0.54	- 834,014.86	1.00	0.54	1,823,014.46	70.00
23	P-1	22	22	154	0.00	314.50	4155.09	2568.62	0.00	137987.50	1823014.46	1126987.10	13.17	0.60	-8.83	0.62	- 696,027.36	1.00	0.62	1,823,014.46	63.00
24	P-1	23	23	161	0.00	148.00	4155.09	2716.62	0.00	64935.29	1823014.46	1191922.39	13.78	0.60	-9.22	0.65	- 631,092.07	1.00	0.65	1,823,014.46	56.00
25	P-1	24	24	168	0.00	190.00	4155.09	2906.62	0.00	83362.88	1823014.46	1275285.27	14.56	0.61	-9.44	0.70	- 547,729.19	1.00	0.70	1,823,014.46	49.00
26	P-1	25	25	175	0.00	40.00	4155.09	2946.62	0.00	17550.08	1823014.46	1292835.35	14.73	0.59	-10.27	0.71	- 530,179.11	1.00	0.71	1,823,014.46	42.00
27	P-1	26	26	182	0.00	159.50	4155.09	3106.12	0.00	69980.94	1823014.46	1362816.29	15.39	0.59	-10.61	0.75	- 460,198.17	1.00	0.75	1,823,014.46	35.00
28	P-1	27	27	189	0.00	910.00	4155.09	4016.12	0.00	399264.30	1823014.46	1762080.59	19.20	0.71	-7.80	0.97	- 60,933.87	1.00	0.97	1,823,014.46	28.00
29	P-1	28	28	196	0.00	11.00	4155.09	4027.12	0.00	4826.27	1823014.46	1766906.86	19.26	0.69	-8.74	0.97	- 56,107.60	1.00	0.97	1,823,014.46	21.00
30	P-1	29	29	203	0.00	35.00	4155.09	4062.12	0.00	15356.32	1823014.46	1782263.18	19.46	0.67	-9.54	0.98	- 40,751.28	1.00	0.98	1,823,014.46	14.00
31	P-1	30	30	210	0.00	66.00	4155.09	4128.12	0.00	28957.63	1823014.46	1811220.81	19.84	0.66	-10.16	0.99	- 11,793.65	1.00	0.99	1,823,014.46	7.00
32	P-1	31	31	217	0.00	26.97	4155.09	4155.09	0.00	11793.65	1823014.46	1823014.46	20.00	0.65	-11.00	1.00	-	1.00	1.00	1,823,014.46	0.00
33	P-2	32	1	7	131.47	0.00	131.47	0.00	57685.25	0.00	57685.25	0.00	0.00	0.00	-1.00	0.00	- 57,685.25	0.22	0.00	263,703.98	0.00
34	P-2	33	2	14	131.47	0.00	262.94	0.00	57685.25	0.00	115370.49	0.00	0.00	0.00	-2.00	0.00	- 115,370.49	0.44	0.00	263,703.98	0.00
35	P-2	34	3	21	131.47	18.00	394.41	18.00	57685.25	7897.96	173055.74	7897.96	0.14	0.05	-2.86	0.05	- 165,157.78	0.66	0.03	263,703.98	25.46
36	P-2	35	4	28	131.47	102.00	525.88	120.00	57685.25		230740.98	52653.04	0.91	0.23	-3.09	0.23	- 178,087.94	0.88	0.21	263,703.98	18.46
37	P-2	36	5	35	75.13	3.60	601.01	123.60	32963.00		263703.98	54232.63	0.94	0.19	-4.06	0.21	- 209,471.35	1.00	0.22	263,703.98	11.46
38	P-2	37	6	42	0.00	42.00	601.01	165.60	0.00	18428.56	263703.98	72661.20	1.26	0.21	-4.74	0.28	- 191,042.78	1.00	0.30	263,703.98	4.46
39	P-2	38	7	49	0.00	21.55	601.01	187.15	0.00	9455.61	263703.98	82116.81	1.42	0.20	-5.58	0.31	- 181,587.17	1.00	0.33	263,703.98	49.00
40	P-2	39	8	56	0.00	0.00	601.01	187.15	0.00	0.00	263703.98	82116.81	1.42	0.18	-6.58	0.31	- 181,587.17	1.00	0.33	263,703.98	42.00
41	P-2	40	9	63	0.00	232.10	601.01	419.25	0.00	101839.76	263703.98	183956.56	3.19	0.35	-5.81	0.70	- 79,747.42	1.00	0.75	263,703.98	35.00
42	P-2	41	10	70	0.00	17.25	601.01	436.50	0.00	7568.87	263703.98	191525.44	3.32	0.33	-6.68	0.73	- 72,178.54	1.00	0.78	263,703.98	28.00
43	P-2	42	11	77	0.00	53.50	601.01	490.00	0.00	23474.48	263703.98	214999.92	3.73	0.34	-7.27	0.82	- 48,704.06	1.00	0.88	263,703.98	21.00
44	P-2	43	12	84	0.00	30.00	601.01	520.00	0.00	13163.26	263703.98	228163.18	3.96	0.33	-8.04	0.87	- 35,540.80	1.00	0.93	263,703.98	14.00
45	P-2	44	13	91	0.00	9.00	601.01	529.00	0.00	3948.98	263703.98	232112.16	4.04	0.31	-8.96	0.88	- 31,591.82	1.00	0.95	263,703.98	7.00
46	P-2	45	14	98	0.00	30.00	601.01	559.00	0.00	13163.26	263703.98	245275.42	4.44	0.32	-9.56	0.93	- 18,428.56	1.00	1.00	263,703.98	0.00
47	P-3	59	1	7	148.19	0.00	148.19	0.00	64062.82	0.00	64062.82	0.00	0.00	0.00	-1.00	0.00	- 64,062.82	0.04	0.00	1,665,633.42	0.00
48	P-3	60	2	14	259.34	0.00	407.53	0.00	112109.94	0.00	176172.76	0.00	0.00	0.00	-2.00	0.00	- 176,172.76	0.11	0.00	1,665,633.42	0.00
49	P-3	61	3	21	259.34	0.00	666.87	0.00	112109.94	0.00	288282.71	0.00	0.00	0.00	-3.00	0.00	- 288,282.71	0.17	0.00	1,665,633.42	0.00
50	P-3	62	4	28	259.34	0.00	926.21	0.00	112109.94	0.00	400392.65	0.00	0.00	0.00	-4.00	0.00	- 400,392.65	0.24	0.00	1,665,633.42	0.00
51	P-3	63	5	35	259.34	0.00	1185.55	0.00	112109.94	0.00	512502.59	0.00	0.00	0.00	-5.00	0.00	- 512,502.59	0.31	0.00	1,665,633.42	0.00
52	P-3	64	6	42	259.34	0.00	1444.89	0.00	112109.94	0.00	624612.53	0.00	0.00	0.00	-6.00	0.00	- 624,612.53	0.37	0.00	1,665,633.42	0.00
53	P-3	65	7	49	259.34	0.00	1704.23	0.00	112109.94	0.00	736722.47	0.00	0.00	0.00	-7.00	0.00	- 736,722.47	0.44	0.00	1,665,633.42	0.00
54	P-3	66	8	56	259.34	6.63	1963.57	6.63	112109.94	2866.12	848832.42	2866.12	0.04	0.01	-7.96	0.00	- 845,966.30	0.51	0.00	1,665,633.42	97.46
55	P-3	67	9	63	259.34	80.09	2222.91	86.72	112109.94	34622.52	960942.36	37488.64	0.59	0.07	-8.41	0.04	- 923,453.72	0.58	0.02	1,665,633.42	90.46
56	P-3	68	10	70	259.34	0.00	2482.25	86.72	112109.94	0.00	1073052.30	37488.64	0.59	0.06	-9.41	0.03	-1,035,563.66	0.64	0.02	1,665,633.42	83.46
57	P-3	69	11	77	259.34	192.59	2741.59	279.31	112109.94	83255.73	1185162.24	120744.37	1.51	0.14	-9.49	0.10	-1,064,417.87	0.71	0.07	1,665,633.42	91.00
58	P-3	70	12	84	259.34	433.00	3000.93	712.31	112109.94	187183.82	1297272.18	307928.20	3.18	0.26	-8.82	0.24	- 989,343.99	0.78	0.18	1,665,633.42	84.00
59	P-3	71	13	91	259.34	112.50	3260.27	824.81	112109.94	48633.21	1409382.13	356561.41	3.61	0.28	-9.39	0.25	-1,052,820.72	0.85	0.21	1,665,633.42	77.00
60	P-3	72	14	98	259.34	210.50	3519.61	1035.31	112109.94	90998.14	1521492.07	447559.55	4.42	0.32	-9.58	0.29	-1,073,932.52	0.91	0.27	1,665,633.42	70.00
61	P-3	73	15	105	259.34	465.40	3778.95	1500.71	112109.94	201190.19	1633602.01	648749.74	6.22	0.41	-8.78	0.40	- 984,852.27	0.98	0.39	1,665,633.42	63.00
62	P-3	74	16	112	74.10	206.50	3853.05	1707.21	32031.41	89268.96	1665633.42	738018.69	7.01	0.44	-8.99	0.44	- 927,614.73	1.00	0.44	1,665,633.42	56.00
63	P-3	75	17	119	0.00	168.00	3853.05	1875.21	0.00	72625.59	1665633.42	810644.28	7.66	0.45	-9.34	0.49	- 854,989.13	1.00	0.49	1,665,633.42	49.00

Figure 4.3 Example of work package dataset after the data preprocessing Feature Selection

4.3.1 Feature Selection

identifies and selects a subset of relevant features (predictors) for the forecasting model construction (Pirbazari et al., 2019). It aims to improve the model's performance by eliminating unnecessary, redundant, or noisy data, leading to more straightforward, faster, and more efficient models (Pabuccu & Barbu, 2023). Moreover, it is relevant to include a visual inspection of independent and dependent variables (Figure 4.4) as a must-do initial action. Notably, the selection of features for this study involved two aspects considering the multivariate time series dataset to be handled. First, analyze the relationship between predictor variables only and second, between each predictor and target variable. The former is known as *multicollinearity*, which was addressed by applying a Variance Inflation Factor (VIF) technique, while the latter was solved by using the Spearman Correlation matrix.

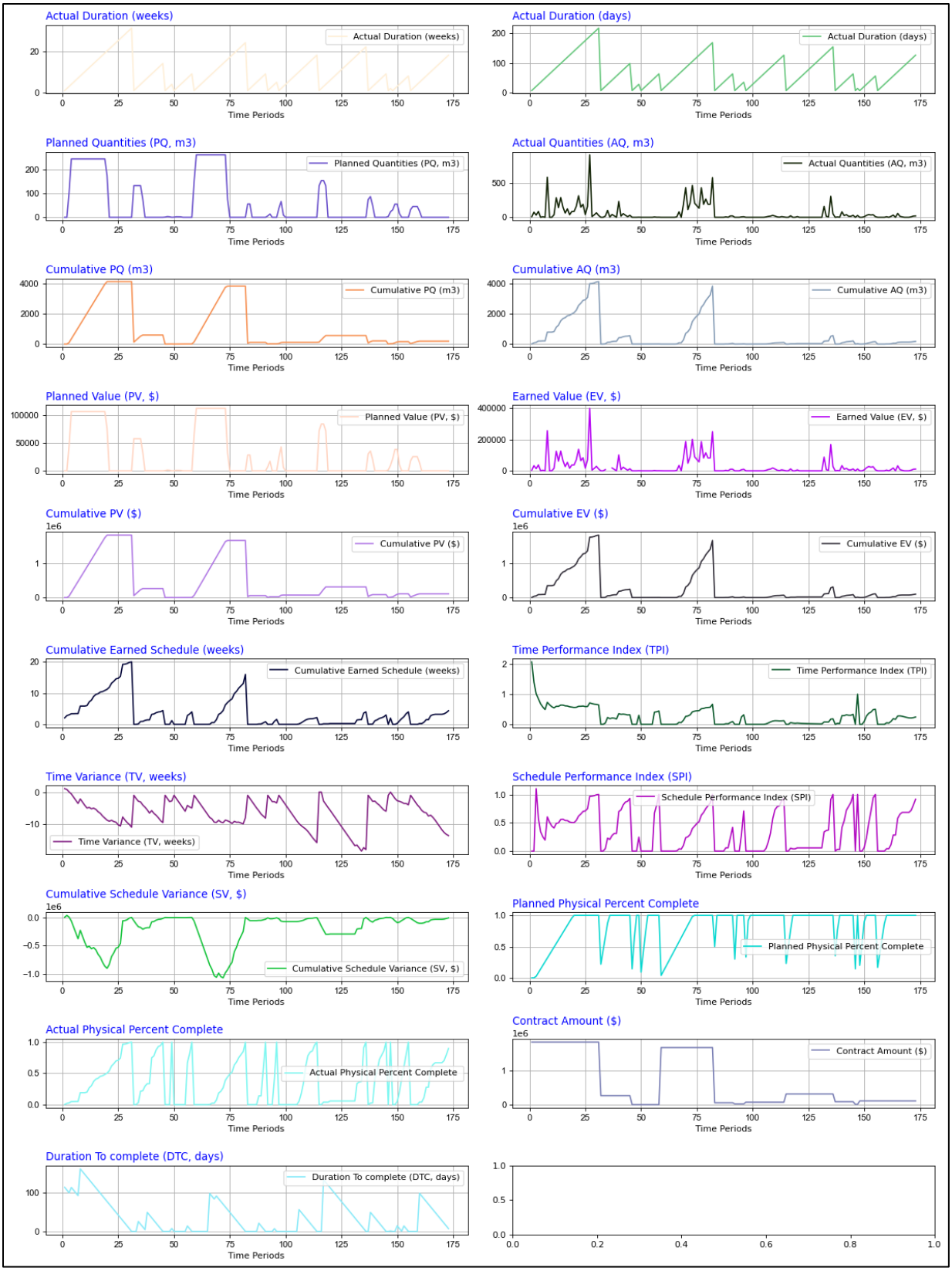


Figure 4.4 Visual Representation of Independent and Dependent Variables

Multicollinearity Analysis by Variance Inflation Factor

Multicollinearity is a situation in regression analysis where two or more predictor variables (independent variables) in a model are highly correlated (Shrestha, 2020). This is a linear relationship that can influence outcome reliability (J. H. Kim, 2019). In the context of a regression model, multicollinearity can be problematic because (Daoud, 2017; Obite et al., 2020):

- It is difficult to isolate each predictor's effects on the target variable.
- It can lead to incorrectly estimated coefficients, which may fluctuate wildly in response to small changes in the model or data.
- It can artificially inflate the standard errors of the coefficients, resulting in a loss of statistical significance for the affected predictors.

Addressing multicollinearity may involve removing some correlated predictors, combining them into a single predictor through dimensionality reduction techniques (like Principal Component Analysis), or using regularization techniques that can handle correlated predictors in the model. On this matter, this study uses the Variance Inflation Factor (VIF) which is focus on the variation of the regression coefficients when predictors are correlated. If no factors are correlated, the VIFs will all be equal to 1 (Daoud, 2017; Senaviratna & A. Cooray, 2019). Here's how VIF is calculated for each predictor variable:

- Run a linear regression using the predictor of interest as the dependent variable and all other predictors as independent variables.
- Calculate the R-squared value from this regression.
- The VIF for that predictor is calculated as $VIF = \frac{1}{1-R^2}$

A VIF equals to 1 indicates no correlation between the k-th predictor and the remaining predictor variables and, hence, a deficient level of multicollinearity. Values of VIF exceeding 5 or 10 suggest

high multicollinearity, depending on the sources and the context of the analysis, and may require further investigation or adjustment of the model. When applying to the dataset, Planned and Actual Quantities and Planned and Earned Values are highly correlated, which implies managing them as explained above. Conversely, the rest of the predictors are going on within tolerable limits. The following figure shows the VIF results for predictors.

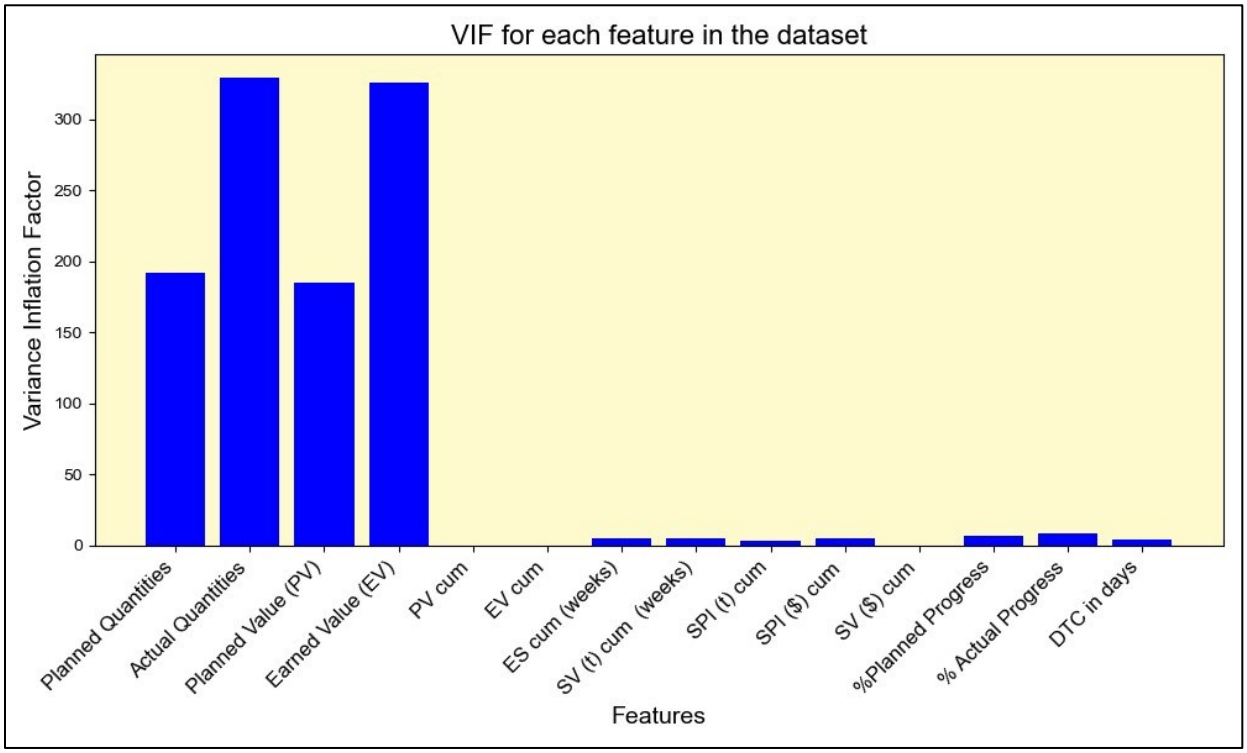


Figure 4.5 Variance Influence Factor (VIF) Performance per Predictor

Spearman Correlation matrix

The correlation matrix shows correlation coefficients between variables (predictors and targets) (Schober et al., 2018). In the following chart, each cell shows the correlation between two variables. The value is in the range of -1 to 1. If two variables have a high correlation, it means the movement of one variable is highly predictive of the movement of the other variable. There are three main types to calculate the correlation factor. The Pearson correlation coefficient is typically

used in correlation matrices, but different types, like Spearman's rank correlation or Kendall's tau, can also be used depending on the data and requirement. For time series data, which often involves trends and does not necessarily follow a normal distribution, Spearman or Kendall might be more appropriate (Croux & Dehon, 2010). Spearman is generally preferred for its balance between sensitivity and robustness, but Kendall's Tau could be a better choice when a dataset is small or particularly noisy. Accordingly, in the present research work, the dataset was evaluated under the Spearman correlation type, whose results are shown in Figure 4.6 – Correlation Matrix.

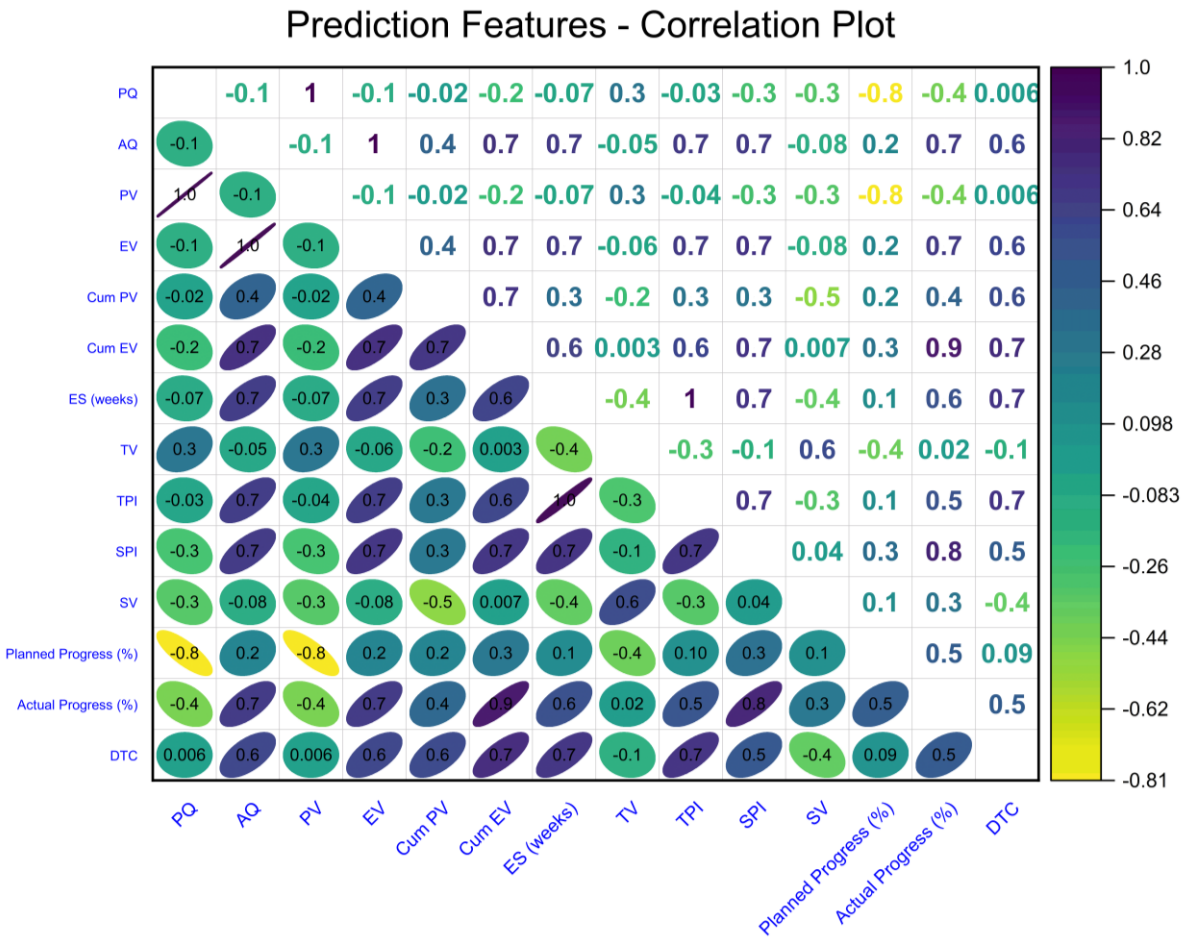


Figure 4.6 Spearman Correlation Matrix for Variables

4.3.2 Data Splitting

Data Splitting is dividing the dataset into three sets, including training, validation, and test sets (Xu & Goodacre, 2018). The training dataset serves to train and learning the model, the validation dataset to adjust the model parameters and prevent overfitting, and the test dataset to assess the model's performance on unseen data (Vabalas et al., 2019). This split is crucial for assessing the model's generalization ability. The ratios for splitting data can vary based on the size and nature of the dataset. Accordingly, the present research evaluated several split ratios considering the characteristics of this specific time series datasets before getting the best outcomes. In time series datasets, it is important to maintain trends/patterns on split datasets and a suggested practice is analysing visually the dataset behaviour curve.

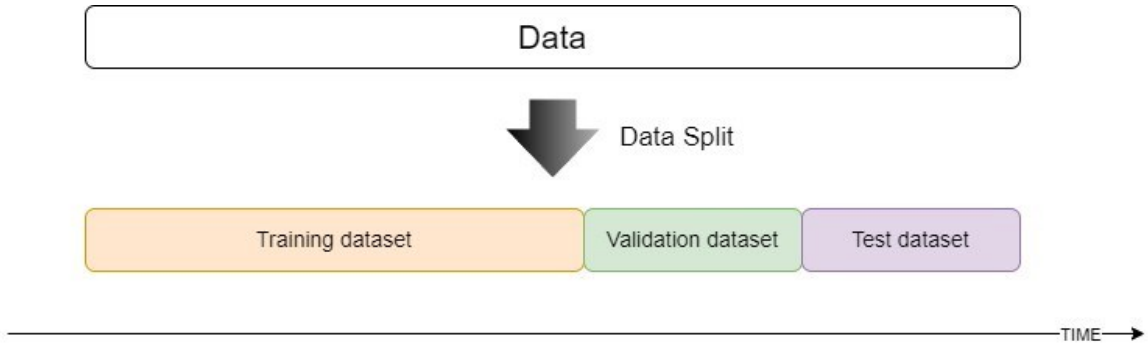


Figure 4.7 Schematic of Data Splitting

4.3.3 Data Normalization

Data normalization is vital in preparing data for time series forecasting, especially when using deep learning models (Bhanja & Das, 2018; Nayak et al., 2014). Normalization adjusts the scale of data to a standard range, typically between 0 and 1 or -1 and 1. This ensures that each input feature contributes equally to model training and predictions (outcomes), facilitating a more

effective learning process. The importance of normalization lies in its ability to prevent features with larger scales from disproportionately influencing the learning dynamics of the model.

Three normalization techniques are commonly employed: min-max normalization, z-score normalization (also known as Standardization), and scaling to unit length. These methods bring different scale features onto a level playing field, crucial in deep learning models that often handle complex, high-dimensional data (Bhanja & Das, 2018). Additionally, normalization aids in avoiding problems like gradient vanishing or exploding, thereby enhancing the model's training efficiency and overall performance. Finally, the process becomes particularly noteworthy when considering the sequence of Data Splitting followed by Data Normalization. First, it is strongly advised to execute the data splitting, as this ensures that the resulting datasets are not influenced by each other when any normalization technique is applied.

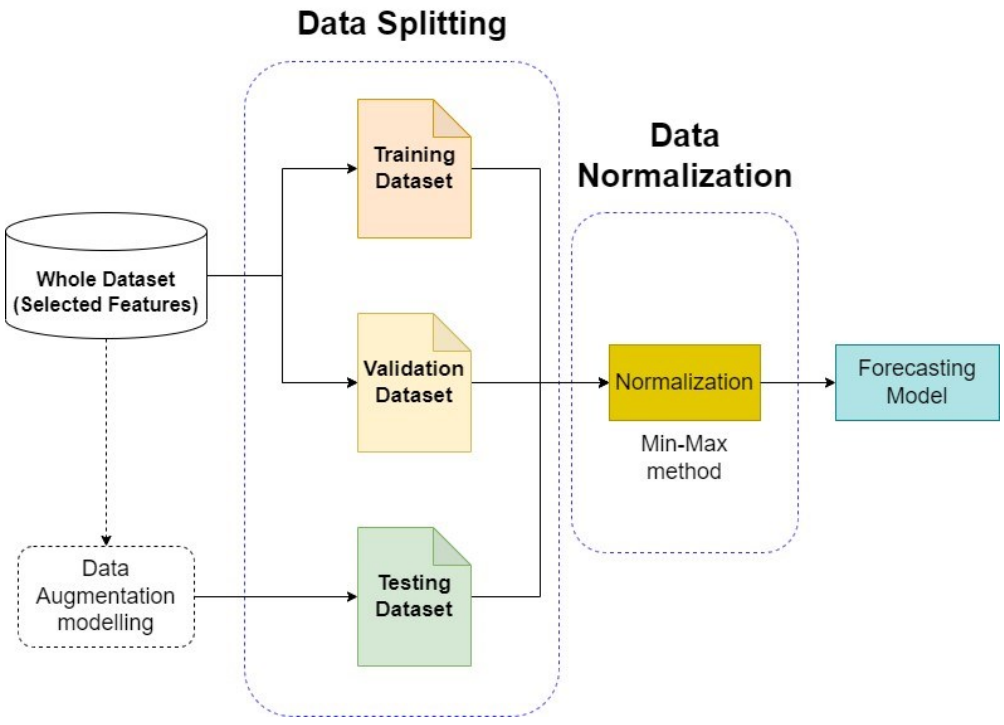


Figure 4.8 Schematic of Data Normalization

4.4 Forecasting Model Development

The first step is transforming the split datasets into more feeding data for the deep learning algorithms. It implies setting the number of pasts that will be considered to predict the Duration to Complete (DTC) (which is the target variable in the present study) and arranging predictors and targets to maintain the autoregressive, a crucial time series dataset property (Ullrich, 2021). This process is also known as the Rolling-Window analysis (Inoue et al., 2017), which is typically used on time series datasets before being processed by any machine learning algorithm. The term "windows" refers to the number of pasts considered for the prediction; the feature input and target variable are represented as "X" values and "y" values, respectively (Zivot et al., 2003).

To manage the autoregressive property, specifically for the multivariate Time Series Dataset, the "X" values are the combination of the predictor variables (i.e. Cumulative Earned Value, Cumulative Earned Schedule, Time Performance index, Project Budget) and the lagged target variables (i.e. the previous values of the target variable). The lagged target valuables are included because of the autoregressive property on the multivariate time series datasets, which can be explained as the existent correlation between the target variable to be predicted in the current period and the past predicted target variables. For example, in time series data will exist correlation among three past predicted values of the Duration to Complete (DTC) (from periods "t-3", "t-2" and "t-1") and the current Duration to Complete (DTC) variable to be predicted (in the period "t"). It can be represented as:

X (for a single sample):

Period t-3: [Cumulative EV, Cumulative ES, TPI, Project Budget, DTC (predicted)]

Period t-2: [Cumulative EV, Cumulative ES, TPI, Project Budget, DTC (predicted)]

Period t-1: [Cumulative EV, Cumulative ES, TPI, Project Budget, DTC (predicted)]

Y (for the same sample):

Period t: [DTC (to be predicted)]

In this sense, the Rolling-Window are arranged as shown in Figure 4.9, which indicates just three rolling-windows groups with three “pasts” and one target each as an example:

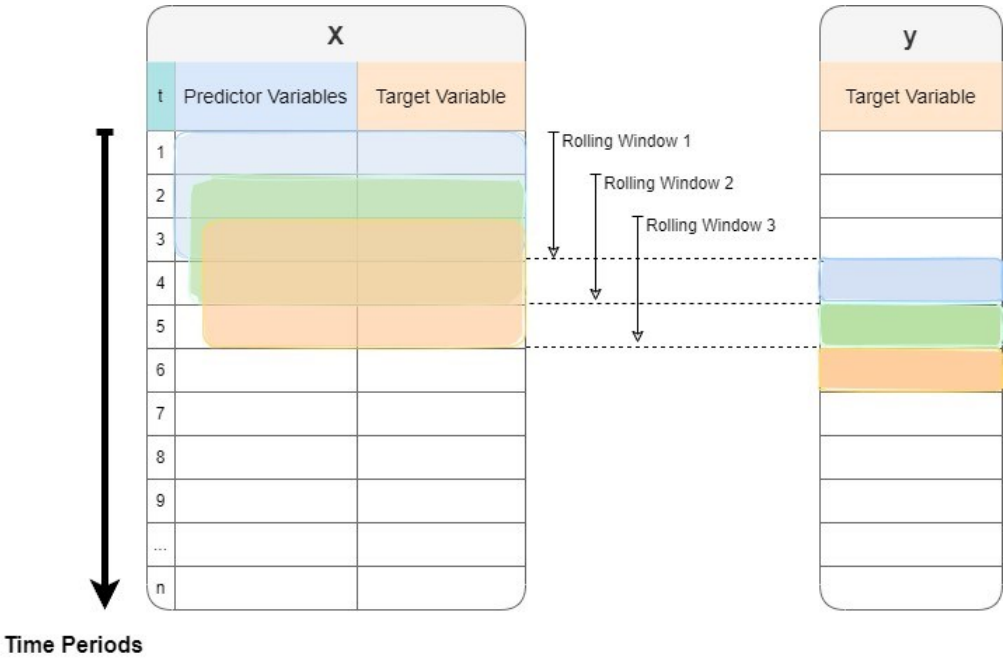


Figure 4.9 Rolling-Window Representation for the Forecasting Model

In practice, each rolling-window is also referred as a sample. It should be noticed that the second sample starts in the second record of the dataset, the third one begins on the third record, and so on. A group of samples is known as a batch. Deep learning models work with a certain quantity of batches, the quantity of samples that process per propagation through the model architecture. The number of batches is a hyperparameter related to the prediction accuracy set by experimentation (Kandel & Castelli, 2020). A graphical representation of a batch formation since samples are shown in Figure 4.10. Moreover, an epoch refers when all batches have been processed through

the entire dataset. During an epoch, the model sees every sample in the dataset once, allowing it to learn from the data over multiple iterations (epochs) to improve its predictions.

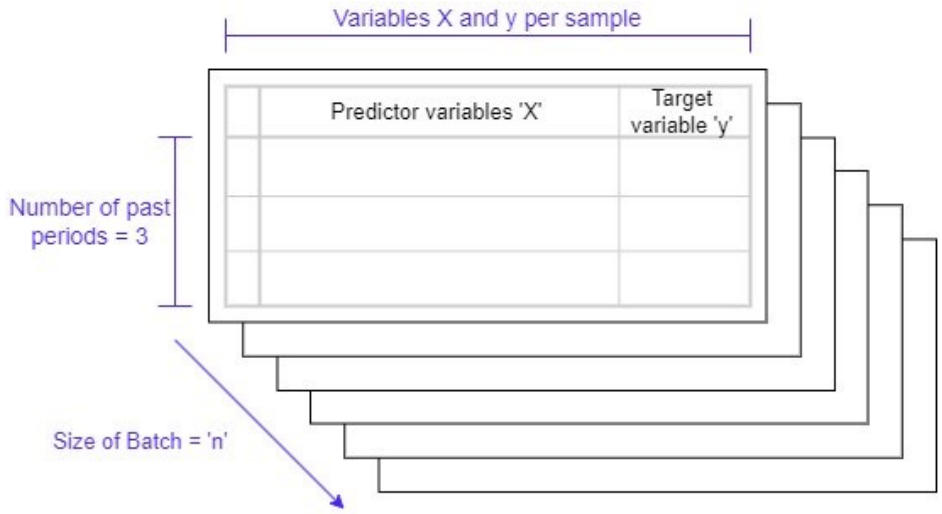


Figure 4.10 Representation of Relationship Between Samples and Batches

4.4.1 Forecasting model with LSTM algorithm

The LSTM algorithm was considered due to previous research's most recent successful results, demonstrating better fitting by handling time series datasets. It is because it contains specific memory cells aiming to remember information for long periods, a remarkable characteristic that differs from other models. This model was computerized using Python's Keras library. Its architecture construction involves setting the number of layers, learning rate, number of epochs, batch sizes and the recurrent dropout factor as the most relevant elements. These items are grouped under the name of hyperparameters within Machine Learning terms. The hyperparameters were tunned by experimentation to find the most optimal set for the LSTM model. They must be set before the learning process and are not associated with the data.

Additionally, selecting the proper Optimizer, an algorithm used to enhance the neural network parameters to reduce the losses, is relevant for the project duration prediction accuracy. The ADAM (Adaptive Moment Estimation) optimizer, an extension of stochastic gradient descent, is designed to be more efficient in this time series forecasting model. ADAM maintains a learning rate for each network weight (parameter). It adapts these rates individually over time, combining the advantages of two other extensions of stochastic gradient descent: Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). In an LSTM layer, each LSTM unit does not divide the samples among themselves. Instead, all units process all samples, contributing to the layer's overall output. The quantity of units represents the dimensionality of the output feature space, not the number of samples each unit processes (Arsov et al., 2021).

4.4.2 Forecasting model with CONV-1D algorithm

Initially developed for image processing, CNNs have proven effective in time series analysis like project progress datasets, thanks to their ability to identify patterns in sequential data. It uses 1D convolutional layers to extract temporal features from sequences, such as features created during project execution. The architecture of a CNN includes convolutional layers, pooling layers, and fully connected layers as hyperparameters. This model was elaborated on using Python language programming and the Keras library. In CNNs, convolutional layers apply filters (kernels) to the input data. Each filter is slid across the input data to produce a feature map that highlights certain features in the input. The same set of filters is applied to all samples in the batch, extracting features from each sample independently.

After convolutional layers, pooling layers may reduce the dimensionality of the feature maps. Eventually, the data may pass through fully connected layers (similar to those in MLPs) before producing the final output. Each layer, whether convolutional, pooling, or fully connected,

processes each sample in the batch using the same weights and operations. Each filter in a convolutional layer produces a feature map for each sample in the batch. The process is parallel across samples but is shared regarding the filters applied. The network learns filters that are useful across all samples in the training dataset, which are applied equally to each sample in the batch.

4.4.3 Forecasting model with MLP algorithm

Multi-Layer Perceptron is a feedforward artificial neural network developed to tackle many problems ranging from simple binary classification to complex regression tasks. The strength of MLPs lies in their ability to model complex, non-linear relationships through multiple layers of computation. It allows for extracting high-level features and relationships across sequential data points. Whether it handles forecasting future trends, identifying cyclic patterns, or detecting anomalies in time series data, MLPs can learn from historical values to make informed predictions about future or unseen events. An MLP comprises an input layer, multiple hidden layers, and an output layer. Each neuron in a layer of an MLP is fully connected to all neurons in the preceding layer. Regardless of the number of neurons in a layer, each neuron processes every sample in the batch. The number of neurons in a layer determines the dimensionality of the output for that layer, but all neurons participate in processing all samples. Each sample in the batch is processed in parallel through the network's layers. The network output for each sample is determined by the collective computation of neurons across the layers, according to the network's weights and biases and the activation functions applied.

4.4.4 Model Performance Measurement.

During the model forecasting design, the loss curve plays an important role. It depicts the model's inaccuracy by plotting the losses between training and validation datasets. Ideally, both curves should decrease over time, indicating the model is learning from the dataset. Interpretation of this

chart spans to identify three possible scenarios: overfitting, underfitting and good fit. While the overfitting indicates that the model learns too well, including noise, which leads to poor generalization on unseen data, the underfitting means that the model cannot capture the underlying data pattern. Ideally, in a good fit, both training and validation losses should decrease to a point of stability with a minimal gap between the two curves. An example of a loss curve is shown in the following figure.

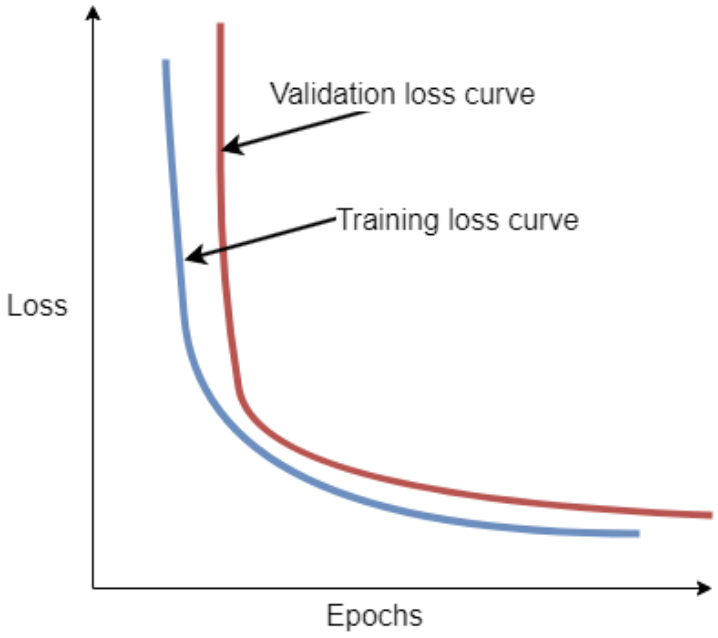


Figure 4.11 Schematic of the Loss Curve

4.5 Data Augmentation

As discussed earlier, deep learning algorithms for time series forecasting require extensive data. By gathering more data, they improve their performance significantly (Javeri et al., 2021). This dataset should be accomplished in terms of quantity and quality (Wen et al., 2020). In addition, actual project data is often incomplete, disorganized, or simply unavailable, especially in a time series manner (Adekunle et al., 2022). To face with this issue, data augmentation arises as an

alternative offering diversity and coherence (X. Zhang et al., 2023), creating copies of the original dataset available (Bandara et al., 2021).

Data augmentation techniques are used to increase the diversity of the dataset without collecting new data. This technique uses available data as a benchmark to create a 'new' dataset. For instance, if actual project datasets are available, where projects have similar conditions among them such as location (country, region, continent, etc.), remoteness (like resource availability, distance to nearby cities, etc.), contract type, adopted PDM, construction type among more relevant aspects, this group of projects can become a benchmark to create new datasets. Like Monte Carlo simulations, when making multiple random scenarios starting from a preset scenario as a benchmark, data augmentation uses random data in a likely range between optimistic and pessimistic values to deliver possible outcomes.

The primary goal of data augmentation is to improve a model's capacity to manage diverse real-world conditions, thus boosting its robustness and ability to generalize (Y. Yue et al., 2023). Data augmentation is not a rigid process that follows a standardized approach; instead, it is often customized to the features and needs of the dataset in question, which can be influenced by domain knowledge. (S. Y. Li, 2020). For augmented (synthetic) time series datasets related to project duration management, domain knowledge plays a pivotal role in guiding appropriate data augmentation outcomes, like coherence, as pointed out by Zhang et al. (2023). This ‘new’ data should reflect realistic scenarios and variations that could occur during the project, which involves understanding the typical patterns, ranges, and behaviours within the existent data. A typical data augmentation pipeline is shown in Figure 4.12.

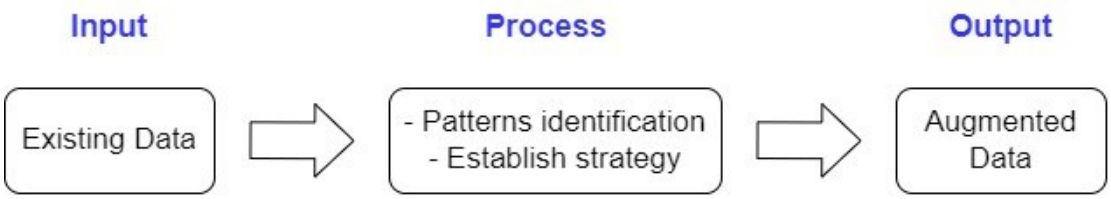


Figure 4.12 Pipeline for Data Augmentation

4.6 Performance Metrics for Time Series Dataset models

This section is focused on the metrics used to assess the accuracy of the machine learning models when applied to time series training/validation/test datasets for forecasting. The present time series data, characterized by its sequential nature and temporal dependencies, requires specialized metrics to capture the model's predictive power in such contexts accurately (Makridakis et al., 2023). Consequently, the performance metrics such as Mean Absolute Scaled Error (MASE) proposed by Hyndman (2006) and Symmetric Mean Absolute Percentage Error (sMAPE) by Makridakis (1993), whose application is focused on Time Series forecasting, were added to the conventional Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).

Table 4.1 Performance Metrics for Time Series Datasets

Metric	**Description**	**Formulas** *(Where y_i is the actual and $\hat{y}_i$ is the predicted and $\overline{y_i}$ is the naïve value)*	**Interpretation**
MAE	Mean Absolute Error calculates the average number of errors in a group of predictions without considering whether they are positive or negative. It is given in the same units as the target variable.	$MAE = \frac{1}{n}\sum_{i=1}^{n} \lvert y_i - \hat{y}_i \rvert$	A lower MAE indicates better model performance.

RMSE	Root Mean Squared Error represents the square root of the average squared differences between prediction and actual observation. It is also given in the same units as the target variable.	$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$	A lower RMSE indicates better performance.
MASE	Mean Absolute Scaled Error measures the accuracy of forecasts relative to a simple benchmark, scaling errors based on the in-sample MAE from a naive forecast.	$MASE = \frac{\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert}{\left(\frac{1}{n-1}\right)\sum\lvert y_i - \bar{\bar{y}}_i\rvert}$	If the MASE value is less than 1, it indicates that the model performs better than a naïve model. Additionally, the lower the MASE value, the better the model performs compared to the naïve model
sMAPE	Like MAPE, Symmetric Mean Absolute Percentage Error adjusts the formula to handle zero and near-zero denominators, providing a more balanced error percentage.	$sMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{\lvert y_i - \hat{y}_i\rvert}{(\lvert y_i\rvert - \lvert\hat{y}_i\rvert)/2}$	A lower sMAPE indicates better accuracy, with values closer to zero indicating more accurate forecasts

While evaluating the performance metrics on the training dataset provides an understanding of the initial learning and model behaviour, the performance metrics on the validation dataset give a chance to tune and select the optimal model. On the other hand, assessing the test dataset performance lets us know how the model will generalize by addressing new, unseen data.

4.7 Calculation of the Overall Project Duration

Once work package durations were accurately predicted through the Deep Learning model, they were integrated to calculate the Overall Project Duration. For this purpose, the proposed methodology is based on the Precedence Diagramming Method (PDM) and the Critical Path

Method (CPM) (Lu, 2020). PDM is a technique used to create a project schedule network diagram that depicts the logic sequence in which tasks must be performed. At the same time, CPM is a technique in project management that helps identify the most extended sequence of tasks in a project (critical path), determining the shortest possible project duration.

Most construction schedules are set following the PDM method. So, the methodology simplifies the PDM network, which contains multiple precedence relationships and lags, into an AON (Activity-On-Node) network, which contains finish-start relationships and *without* lags. It uses dummy nodes to represent lags and durations equivalent to the lag value. After that, CPM is applied to find the overall project duration. The following figure shows the methodology proposed sequence:

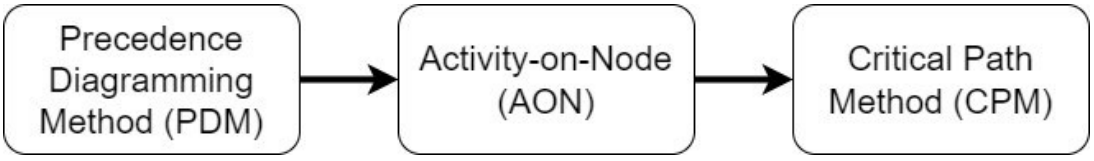


Figure 4.13 Pipeline for Overall Project Duration Calculation

Each is considered an activity when applying this framework to the work packages. Moreover, it is remarkable to analyze the current project schedule at the work package level, which provides planned/actual durations, start dates and finish dates. This methodology was modelled in Python and is explained as follows:

1. As the Primavera P6 software presents limitations when managing schedule information at the work package level, this modelling imports activities' information from Primavera P6, such as durations, start and finish dates (planned and actuals), overlaps and precedence relationships. Then, it is processed in a Python environment to calculate the work package

features like durations start and finish dates. The work package start date is calculated as the finding of the earliest activity start date within such work package; similarly, the work package finish date is the latest activity finish date. Next, work package durations are calculated by subtracting the start and finish dates.

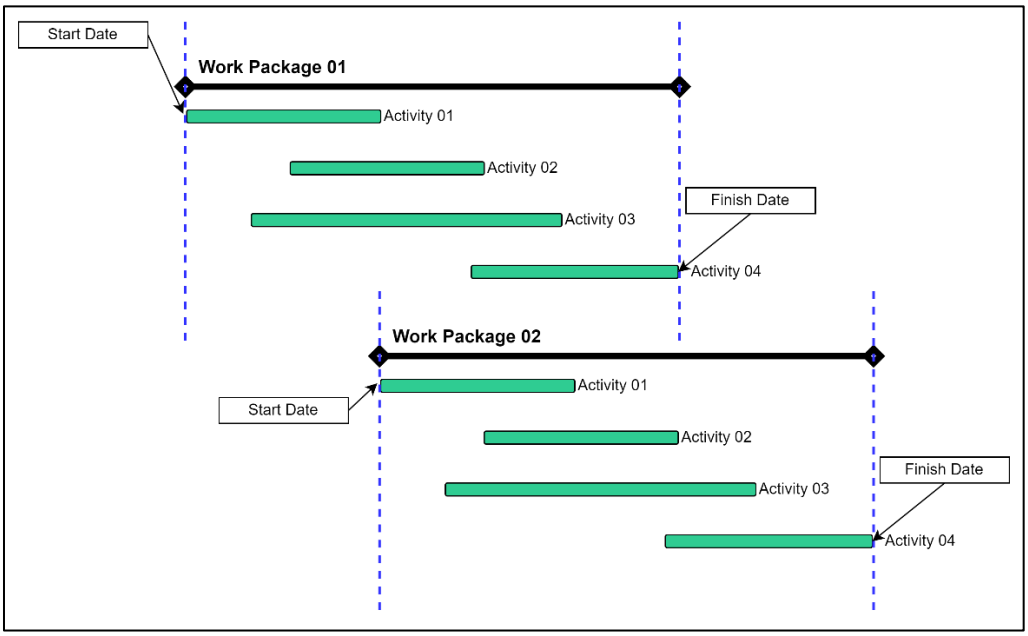


Figure 4.14 Duration, Start and Finish Dates Handling when Extracting from Primavera P6

2. After that, the work packages were set up sequentially to recall their lifecycle occurrences (Boskers & AbouRizk, 2005; Siu et al., 2014). To do so, the model analyzes every start date from the current project schedule. Accordingly, lags among them were computed by subtracting the start dates of sequential work packages (see Figure 4.15). Also, the type of relationship is attributed to a start–to–start relationship between consecutive work packages, which will serve to proceed to the transform schemes later.

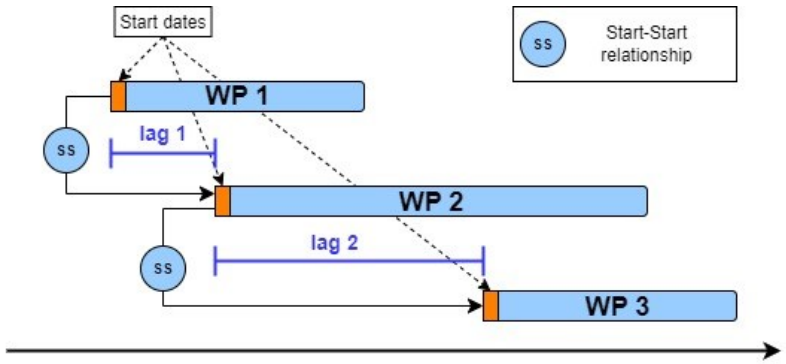


Figure 4.15 Setup of Work Packages

3. After that, the PDM network will be obtained. To do so, the durations of work packages are inputted according to their status: *Completed*, *In Progress* or *Non-Started*. The actual durations of *completed* work packages are considered. In the case of *in-progress* work packages, their durations are calculated using deep learning modelling. Similarly, as *non-started* work packages do not create records, forecasting their final duration should not be possible; hence, their planned durations are inputted.
4. Then, it is continued with the transformation on the AON network that includes precedence relationship as type Finish-Start solely. It creates dummy work packages between them by splitting the earliest work package and then imputing the lag time between them as the duration of the dummy work package (See Figure 4.16).

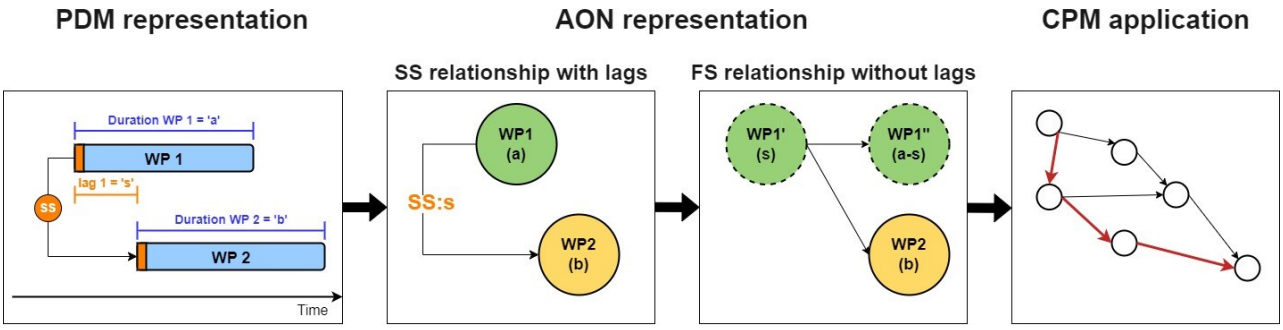


Figure 4.16 Schematic of methodology flow describing transformation from PDM to CPM

5. After transforming the Work Package PDM representation to the AON network without lags where there are Finish–Start relationships only, the CPM method is applied to calculate the overall project duration.

Integrating Deep Learning predictions into CPM-PDM allows for a more data-driven and potentially accurate assessment of project timelines. Moreover, it can help project managers:

- **Improve Scheduling Accuracy:** The overall project schedule can be estimated more precisely by providing more accurate predictions of work package durations.
- **Identify Risk Areas:** Understand which parts of the project might be at risk due to potential delays in critical work packages.
- **Enhance Decision Making:** Make informed decisions about project planning, scheduling, and management based on data-driven insights.

This approach modernizes traditional project management methods with the predictive power of machine learning, potentially leading to more efficient, reliable, and successful project outcomes.

4.8 Graphical User Interface (GUI) for Project Duration Forecasting

A poor handling of deep learning algorithms can lead to misuse or misunderstood due to their complexity. This section describes the design and implementation of a friendly Graphical User Interface (GUI) to manage deep learning algorithms. It was programmed using Python's Tkinter library and designed to be used and applied by non-expert users during the project execution monitoring stage (also known as project tracking).

It is remarkable that this type of GUI based on purely deep learning for project management software, particularly for duration forecasting, is an emerging field that combines advanced predictive analytics capabilities with traditional project management methodologies. Most

software solutions use a combination of techniques, including conventional statistical methods and some machine learning components, to enhance forecasting accuracy. However, these might be integrated into broader project management tools or specialized analytics platforms. Applying pure deep learning models for forecasting project duration, such as Long Short-Term Memory networks, is still relatively innovative.

4.8.1 Software Design and Reporting

The software design consists of two main areas. The “Main Menu” is located on the left, and the “Displaying area” is on the right. The Main Menu contains three sections: “Projects Setup”, “Deep Learning Forecasting Data”, and “Deep Learning Forecasting”. Moreover, the displaying area dynamically show windows according to the selected button in the Main Menu. Below, each section is explained in detail.

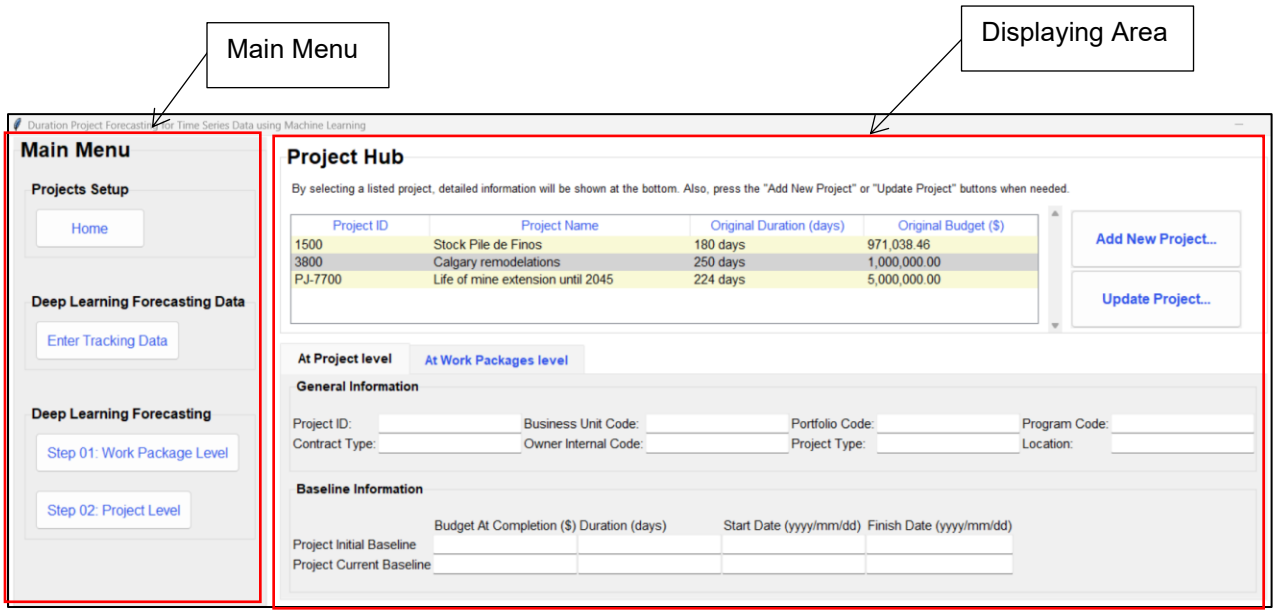


Figure 4.17 Location of Main Menu and Displaying Area

- **Main Menu - Projects Setup Section:**

Home: Clicking on the “Home” button will display the “Project Hub” window, which contains detailed information on the projects that have been entered. Each project is listed in the top left region within the project hub, describing its Project ID, Name, Original Duration, and Original Budget. Similarly, the “Add New Project” and “Update Project” buttons are distinguished on the top right area, which will open new windows separately to enter new project information or update an existing one. The information requested is aligned with the ERD described in Chapter 3. Finally, in the Project Hub’s bottom region, a notebook is shown with two tabs: "At Project Level" and "At Work Package Level." These notebook tabs will interactively populate when you select any project from the top area. Below are the windows used when clicking “Add New Project” and “Update Project.”

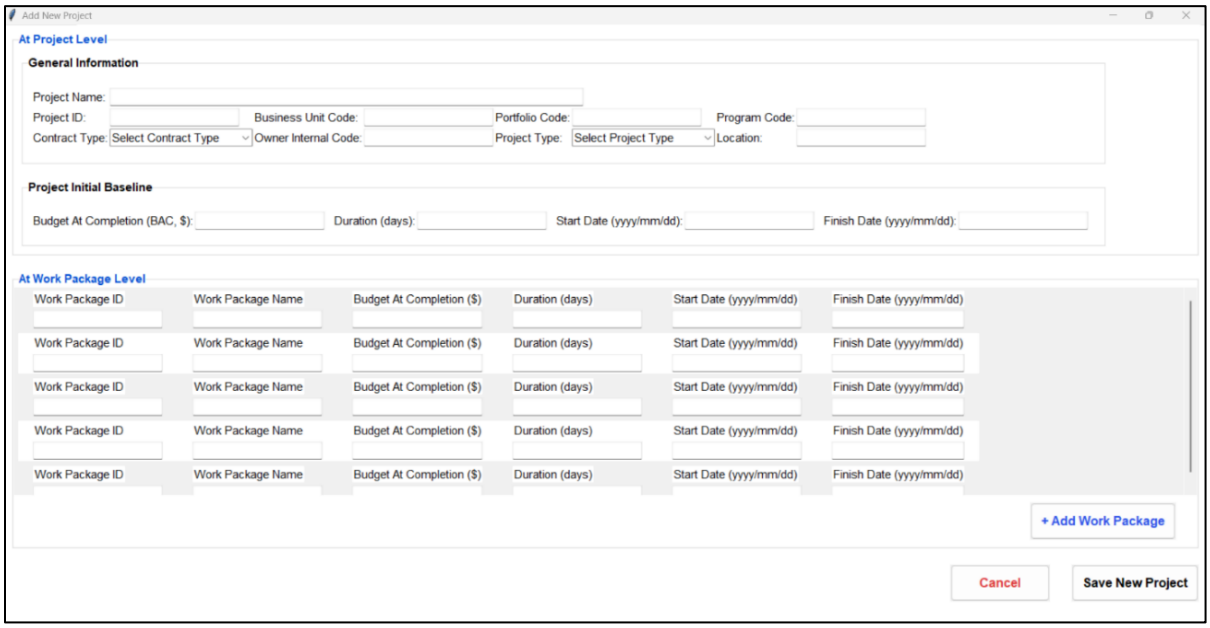


Figure 4.18 The “Add New Project” Window.

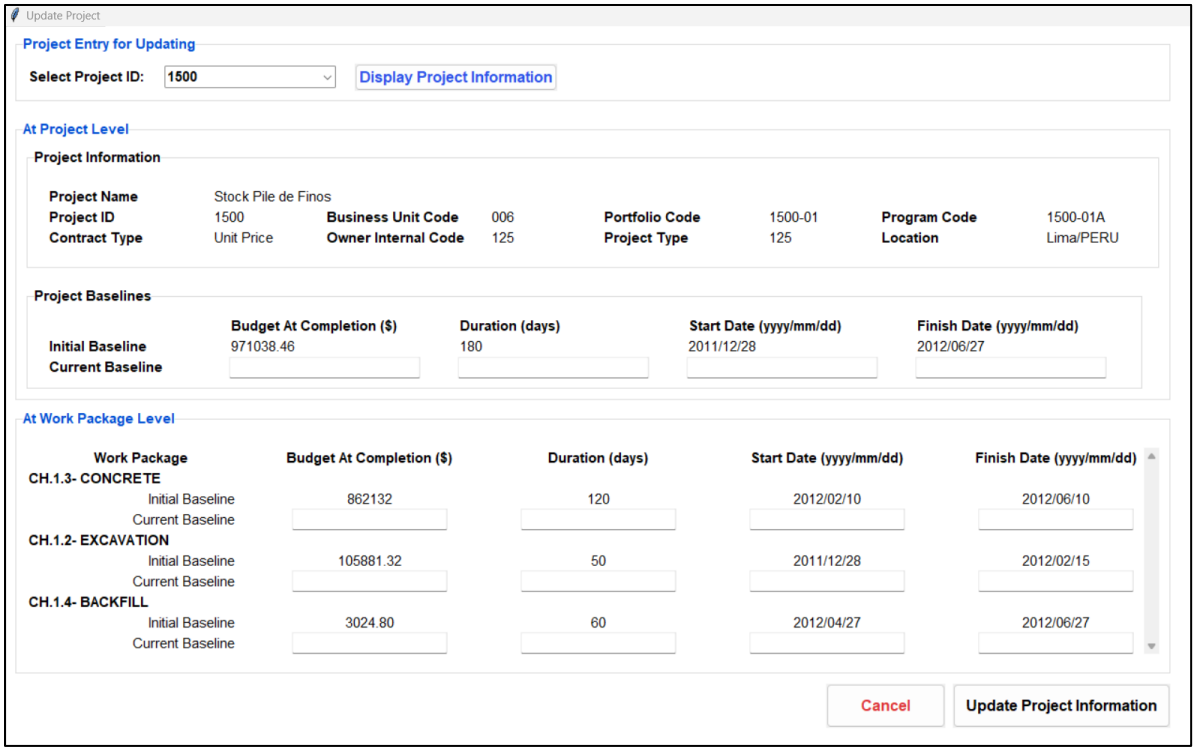


Figure 4.19 The “Update Project” window.

The “Add Project window” requires filling in general information on the top entries and adding work package information on the bottom. On the latter, clicking the “+ Add Work Package” button will add rows as many as quantity of work packages needed without limits. Finally, the “Save New Project” button should be pressed.

- **Main Menu – Deep Learning Forecasting Data section:**

This section contains the “Enter Tracking Data” button, which enables users to enter tracking data per period per work package. First, the project name and reporting date will be entered. After clicking “Enter Data,” new entry fields will appear per work package, as shown in Figure 4.20 below. It contains the Project Work Packages arranged as rows and the features for forecasting as columns. After completing the information, it should be clicked on “Save.”

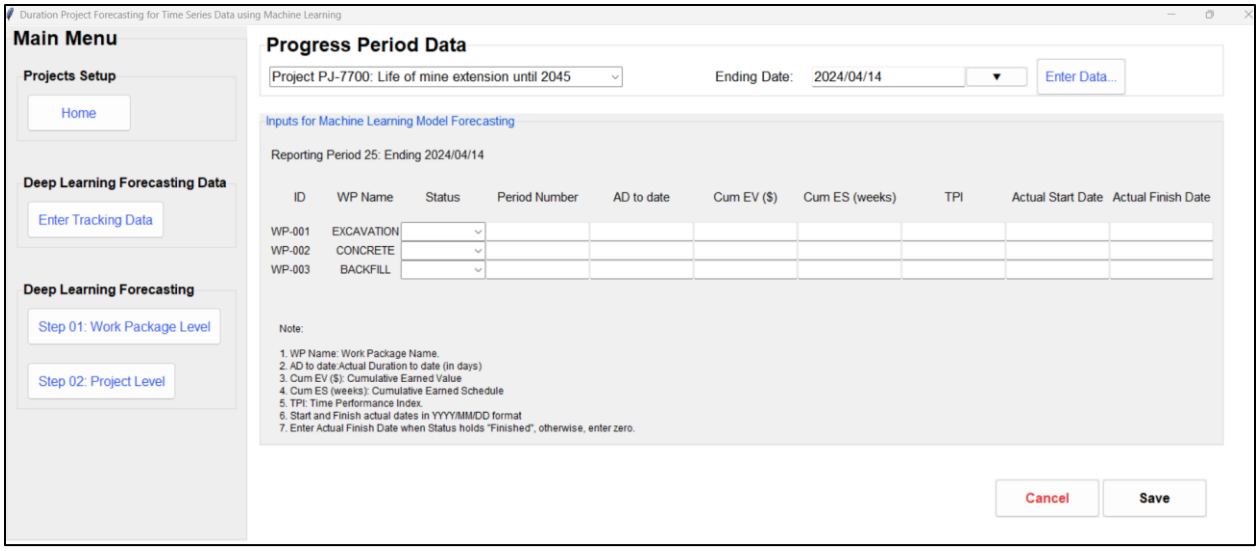


Figure 4.20 The "Project Tracking" window.

The "Status" of the work package is quite significant when entering tracking data. In the GUI, the "Status" entry will show "non-started," "in progress," or "finished" options from a dropdown list. When the work packages have a "non-started" status, the rest of its entry inputs should be filled with null (zero) values. Also, when the work package is "in progress," complete the requested progress data; otherwise, when it is "Finished," complete the actual data, considering the actual last period data. Additionally, when a work package is "in progress," the following entry input, called Project Period, is solely related to the work package, and can differ from the reporting period number. Figure 4.21 shows an example where the timeline for the entire project and each work package differs for Work Packages 02 and 03.

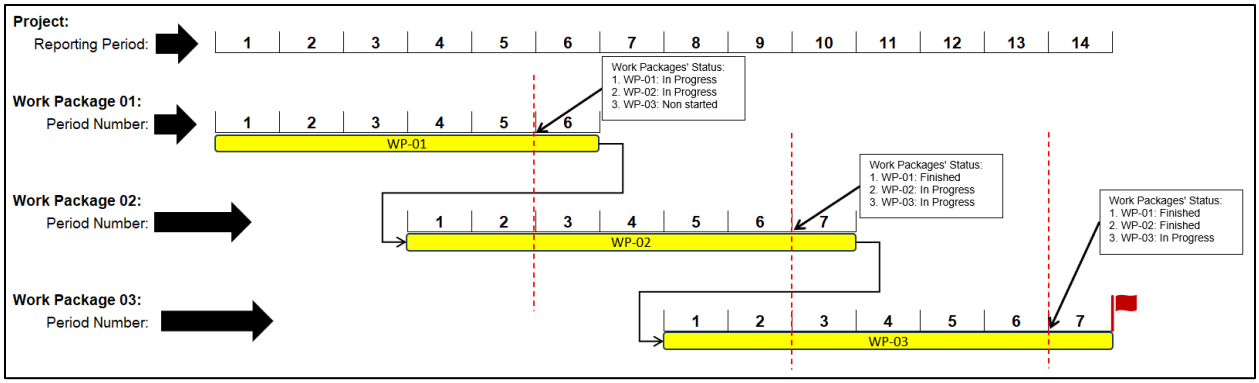


Figure 4.21 Project Reporting Period against Work Package Period Number

- **Main Menu – Deep Learning Forecasting:**

This section includes "Step 01: Work Package Level" and "Step 02: Project Level". **Step 1** will require the Project ID and the Reporting Period. By doing so, it will run the deep learning algorithm and will display the results per work package. On the top is horizontally listed the project work packages, and at the bottom, a Gantt chart showing the planned, actual, and forecasted durations and a Schedule Deviation per period chart depicting the variation of finish dates planned and forecasted. These charts will change dynamically when selecting a work package and another reporting perio*d*.

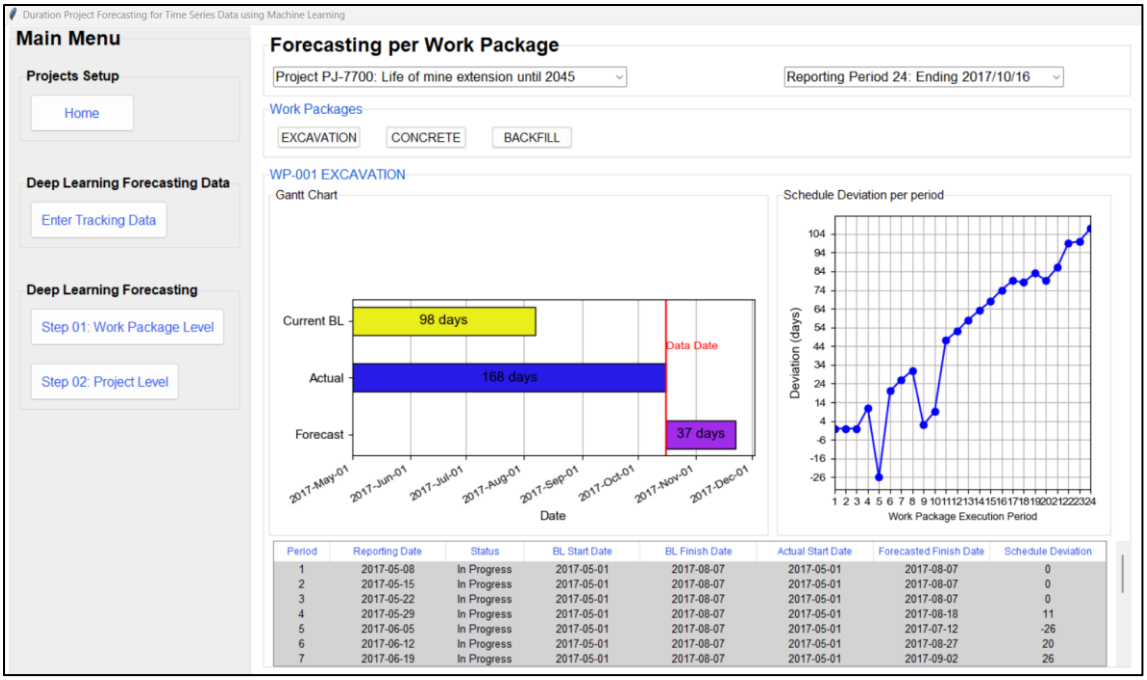


Figure 4.22 “Forecasting per Work Package” window.

On the other hand, **Step 02** will require the Project ID, the Reporting Period. Then, the relationships among work packages will be shown in a table when uploading the Primavera P6 Schedule at the work package level in .xml format, as shown in Figure 4.23. After that, the button “Display Duration at Completion Forecasting” will show the Gantt Chart of the entire project. Also, a button will be displayed to show the critical path for PDM-CPM calculation.

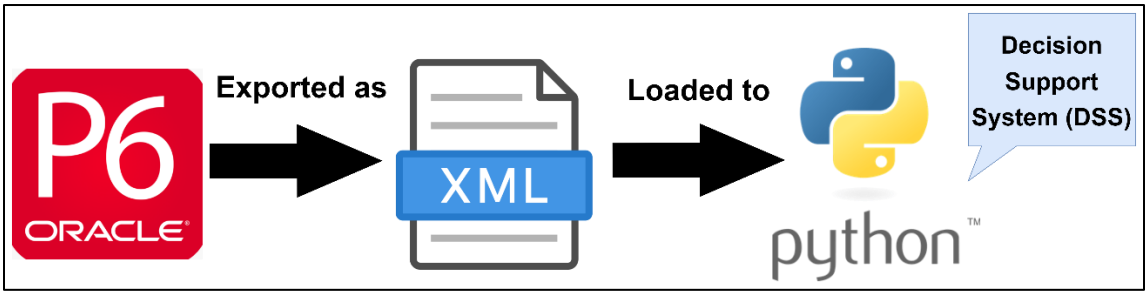


Figure 4.23 “Forecasting per Work Package” window.

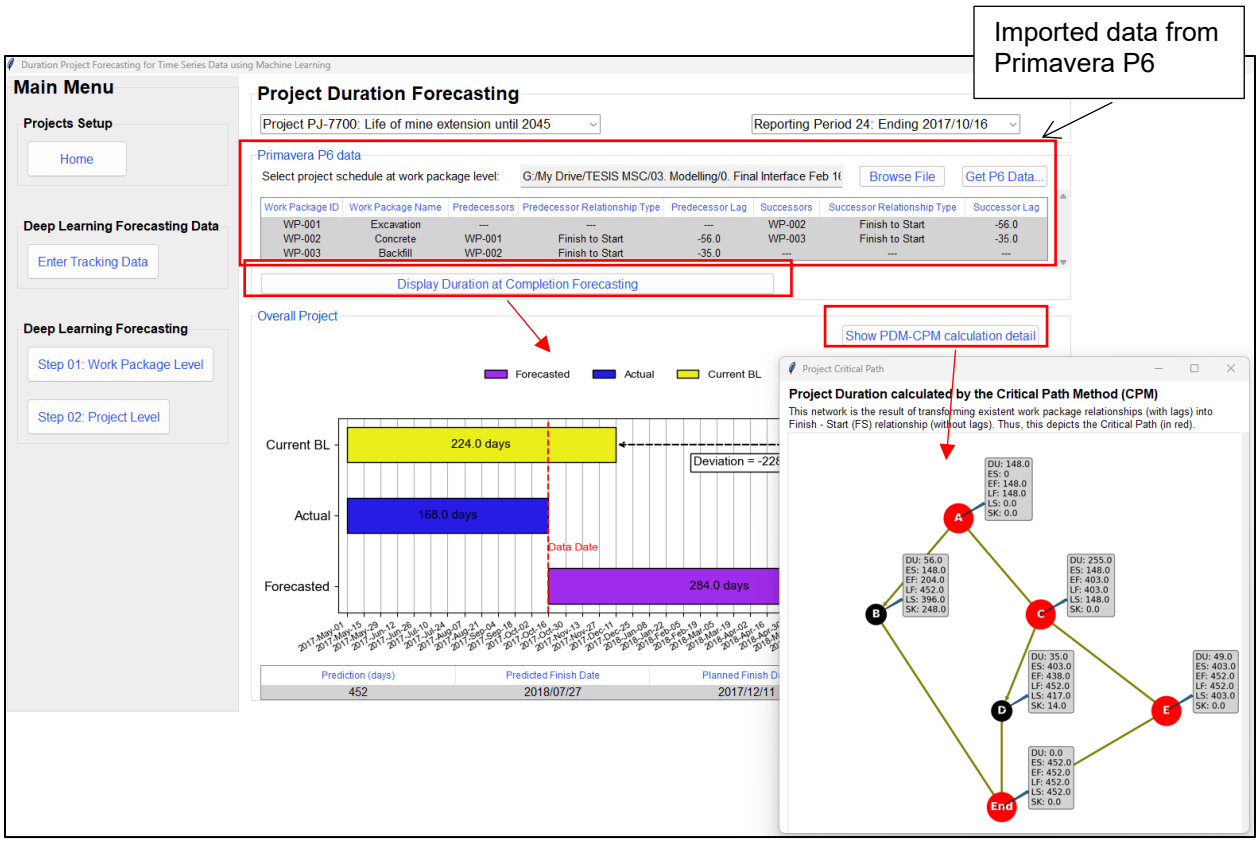


Figure 4.24 “Project Duration Forecasting” window.

Chapter 5 Project Duration Forecasting and Graphical User Interface Deployment

5.1 Introduction

This chapter outlines *applying* deep learning predictive modelling alongside the Graphical User Interface (GUI). It discusses using training, validation, and test datasets to evaluate and select the optimal forecasting model from the Long Short-Term Memory (LSTM), Convolutional Neural network 1-D (CONV-1D), and Multi-Layer Perceptron (MLP) algorithms. The best-performing algorithm is then implemented in the GUI. To do so, each forecasting model was evaluated using actual data collected. Then, data preprocessing, splitting, and normalization, as well as the assessment of three algorithms to determine the highest accuracy performer for integration into the GUI, was performed.

The GUI functionality is demonstrated step-by-step, considering project data across the three work packages evaluated previously (i.e. Concrete, Excavation and Backfill). This GUI application also addresses the challenges encountered and the outcomes, highlighting the synergy between advanced analytics and user-centred design in the resulting predictive analysis tool. A visualization of the process using the GUI is presented in Figure 5.1.

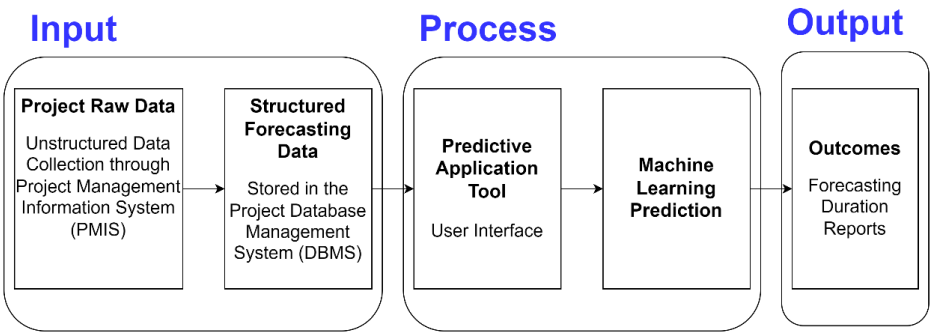


Figure 5.1 Pipeline of GUI application for Project Duration Forecasting

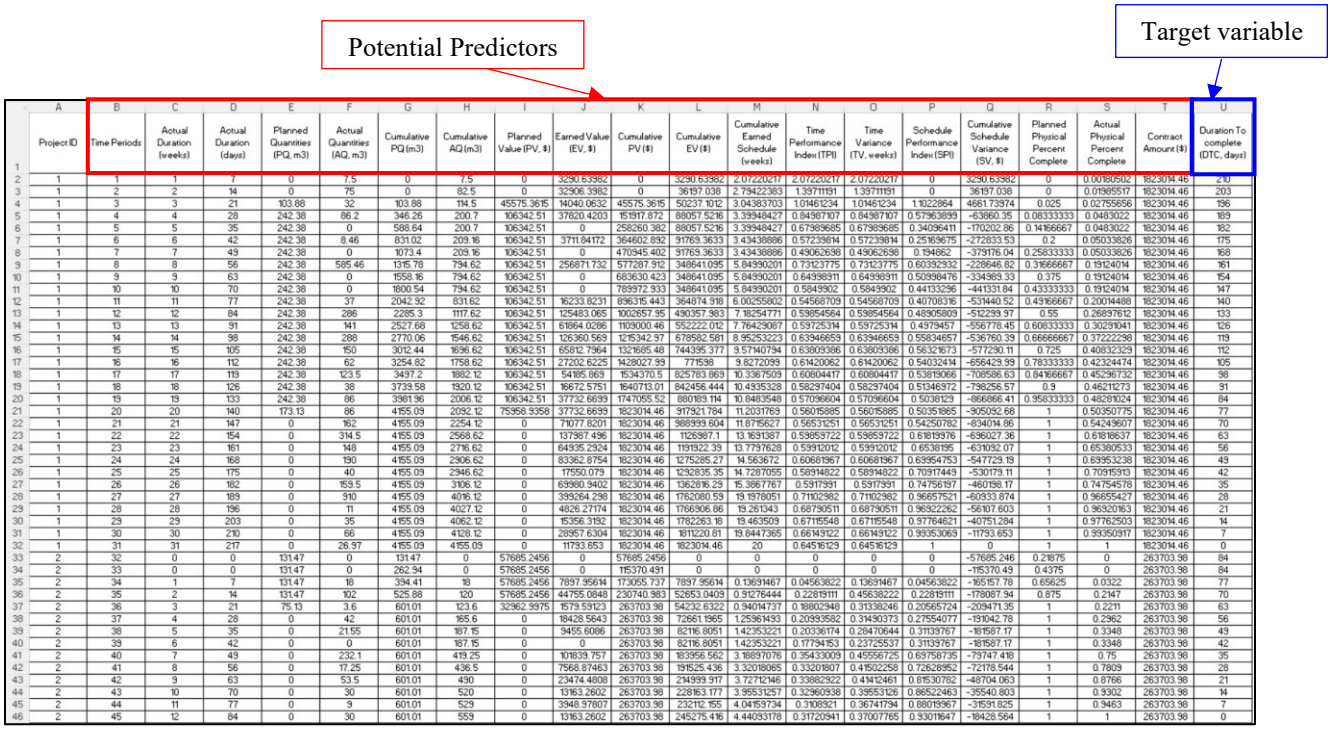
5.2 Duration Forecasting model application – Case Study

This section describes the application of three deep learning algorithms: LSTM, CONV-1D and MLP; which are evaluated with actual project data. The goal is to put into practice the steps described in the previous chapter to create the forecasting model. After that, each models' prediction performance is calculated to select the optimal predictive deep-learning algorithm. Finally, this is included in the Graphical User Interface (GUI) to perform the overall project duration at completion.

5.2.1 Data Preprocessing and Feature Selection

In Section 3.4, the project's actual data was presented. Originally, this data pertains to a mining civil project portfolio contained five work packages; however, after conducting the initial data cleaning, three work packages, namely Concrete, Excavation, and Backfill, were retained. The remaining two work packages, Demolition and Ground Mesh, were discarded due to insufficient data. Similarly, some records with missing values were removed from the three work package datasets to ensure consistency, resulting 173, 65, and 85 records for the Concrete, Excavation, and Backfill work packages.

Figure 5.2 is an example of the preprocessed data, where only five out of thirteen projects considered in the analysis of the Excavation work package are presented. On this dataset, multicollinearity processes and Spearman correlation were performed, identifying the input variables for the model. These variables are Actual Duration, Cumulative EV, Cumulative Earned Schedule, Time Performance Index, and Contract Amount. It is important to note that the Duration To Complete (DTC) is the target variable for the analysis.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Project ID	Time Periods	Actual Duration (weeks)	Actual Duration (days)	Planned Quantities (PQ, m3)	Actual Quantities (AQ, m3)	Cumulative PQ (m3)	Cumulative AQ (m3)	Planned Value (PV, $)	Earned Value (EV, $)	Cumulative PV ($)	Cumulative EV ($)	Cumulative Earned Schedule (weeks)	Time Performance Index (TPI)	Time Variance (TV, weeks)	Schedule Performance Index (SPI)	Cumulative Schedule Variance (SV, $)	Planned Physical Percent Complete	Actual Physical Percent Complete	Contract Amount ($)	Duration To complete (DTC, days)
2	1	1	1	7	0	7.5	0	7.5	0	3290.63982	0	3290.63982	2.07220217	2.07220217	2.07220217	0	3290.63982	0	0.00180502	1823014.46	210
3	1	2	2	14	0	75	0	82.5	0	32906.3982	0	36197.038	2.79422383	1.39711191	1.39711191	0	36197.038	0	0.01985517	1823014.46	203
4	1	3	3	21	103.88	32	103.88	114.5	45575.3615	14040.0632	45575.3615	50237.1012	3.04383703	1.01461234	1.01461234	1.1022864	4661.73974	0.025	0.02755656	1823014.46	196
5	1	4	4	28	242.38	86.2	346.26	200.7	106342.51	37820.4203	151917.872	88057.5216	3.39348427	0.84967107	0.84967107	0.57963899	-63860.35	0.08333333	0.0483022	1823014.46	189
6	1	5	5	35	242.38	0	588.64	200.7	106342.51	0	258260.382	88057.5216	3.39348427	0.67389685	0.67389685	0.34096411	-170202.86	0.14166667	0.0483022	1823014.46	182
7	1	6	6	42	242.38	8.46	831.02	209.16	106342.51	3711.84172	364602.892	91769.3633	3.43438886	0.57239814	0.57239814	0.25163675	-272833.53	0.2	0.05033826	1823014.46	175
8	1	7	7	49	242.38	0	1073.4	209.16	106342.51	0	470945.402	91769.3633	3.43438886	0.49062698	0.194862	0.194862	-379176.04	0.25833333	0.05033826	1823014.46	168
9	1	8	8	56	242.38	585.46	1315.78	794.62	106342.51	256871.732	577287.912	348641.095	5.84390201	0.73123775	0.73123775	0.60392332	-228646.82	0.31666667	0.19124014	1823014.46	161
10	1	9	9	63	242.38	0	1558.16	794.62	106342.51	0	683630.423	348641.095	5.84390201	0.64938911	0.64938911	0.50938476	-334983.33	0.375	0.19124014	1823014.46	154
11	1	10	10	70	242.38	0	1800.54	794.62	106342.51	0	789972.933	348641.095	5.84390201	0.5849902	0.5849902	0.44132296	-441331.84	0.43333333	0.19124014	1823014.46	147
12	1	11	11	77	242.38	37	2042.92	831.62	106342.51	16233.8231	896315.443	364874.918	6.00255802	0.54568709	0.54568709	0.40708316	-531440.52	0.45166667	0.20014488	1823014.46	140
13	1	12	12	84	242.38	286	2285.3	1117.62	106342.51	125483.065	1002657.95	490357.983	7.18254771	0.59854564	0.59854564	0.48905809	-512299.97	0.55	0.26897612	1823014.46	133
14	1	13	13	91	242.38	141	2527.68	1258.62	106342.51	61864.0286	1109000.46	552222.012	7.76429087	0.59725314	0.59725314	0.4979457	-556778.45	0.60833333	0.30291041	1823014.46	126
15	1	14	14	98	242.38	288	2770.06	1546.62	106342.51	126360.569	1215342.57	678582.581	8.95253223	0.63946859	0.63946859	0.55834657	-536760.39	0.76666667	0.37222298	1823014.46	119
16	1	15	15	105	242.38	150	3012.44	1696.62	106342.51	65612.7964	1321685.48	744395.377	9.51740794	0.63805386	0.63805386	0.56321673	-577230.11	0.725	0.40832329	1823014.46	112
17	1	16	16	112	242.38	62	3254.82	1758.62	106342.51	27202.6225	1428027.99	771598	9.8272099	0.61420062	0.61420062	0.54032414	-656429.99	0.78333333	0.42324474	1823014.46	105
18	1	17	17	119	242.38	123.5	3497.2	1882.12	106342.51	54185.869	1534370.5	825783.869	10.3367509	0.60804417	0.60804417	0.53819066	-708586.63	0.84166667	0.45296732	1823014.46	98
19	1	18	18	126	242.38	38	3739.58	1920.12	106342.51	16672.5751	1640713.01	842456.444	10.4935328	0.58297404	0.58297404	0.51346972	-738256.57	0.9	0.46212723	1823014.46	91
20	1	19	19	133	242.38	86	3981.96	2006.12	106342.51	37732.6693	1747055.52	880189.114	10.8483548	0.57096604	0.57096604	0.5038129	-866866.41	0.95833333	0.48281024	1823014.46	84
21	1	20	20	140	173.13	86	4155.09	2092.12	75958.9358	37732.6693	1823014.46	917921.784	11.2031769	0.56015885	0.56015885	0.50351865	-905092.68	1	0.50350775	1823014.46	77
22	1	21	21	147	0	162	4155.09	2254.12	0	71077.8201	1823014.46	988993.604	11.8775627	0.56531251	0.56531251	0.54250782	-834014.86	1	0.54249607	1823014.46	70
23	1	22	22	154	0	314.5	4155.09	2568.62	0	137987.496	1823014.46	1126987.1	13.1691387	0.59859722	0.59859722	0.61819976	-696027.36	1	0.61818637	1823014.46	63
24	1	23	23	161	0	148	4155.09	2716.62	0	64935.2924	1823014.46	1191922.39	13.7797628	0.53912012	0.53912012	0.6538195	-631092.07	1	0.65380533	1823014.46	56
25	1	24	24	168	0	190	4155.09	2906.62	0	83362.8754	1823014.46	1275285.27	14.563672	0.60681967	0.60681967	0.69954753	-547723.19	1	0.69353238	1823014.46	49
26	1	25	25	175	0	40	4155.09	2946.62	0	17550.079	1823014.46	1292835.35	14.7287055	0.58914822	0.58914822	0.70917449	-530179.11	1	0.70915913	1823014.46	42
27	1	26	26	182	0	159.5	4155.09	3106.12	0	69360.9402	1823014.46	1362816.29	15.3867767	0.5917391	0.5917391	0.74756197	-460198.17	1	0.74754578	1823014.46	35
28	1	27	27	189	0	910	4155.09	4016.12	0	399264.298	1823014.46	1762080.59	19.1978051	0.71102382	0.71102382	0.96657521	-60933.874	1	0.96655427	1823014.46	28
29	1	28	28	196	0	11	4155.09	4027.12	0	4826.27174	1823014.46	1766906.86	19.261043	0.68790511	0.68790511	0.96322262	-56107.603	1	0.96320163	1823014.46	21
30	1	29	29	203	0	35	4155.09	4062.12	0	15356.3192	1823014.46	1782263.18	19.463509	0.67115548	0.67115548	0.97764621	-40751.284	1	0.97762503	1823014.46	14
31	1	30	30	210	0	66	4155.09	4128.12	0	28957.6304	1823014.46	1811220.81	19.8447365	0.66749122	0.66749122	0.99353069	-11793.653	1	0.99350917	1823014.46	7
32	1	31	31	217	0	26.97	4155.09	4155.09	0	11793.653	1823014.46	1823014.46	20	0.64516129	0.64516129	1	0	1	1	1823014.46	0
33	2	32	0	0	131.47	0	131.47	0	57685.2456	0	57685.2456	0	0	0	0	0	-57685.246	0.21875	0	263703.98	84
34	2	33	0	0	131.47	0	262.94	0	57685.2456	0	115370.491	0	0	0	0	0	-115370.49	0.4375	0	263703.98	84
35	2	34	1	7	131.47	18	394.41	18	57685.2456	7897.95614	173055.737	7897.95614	0.13691467	0.04563822	0.13691467	0.04563822	-165157.78	0.65625	0.0322	263703.98	77
36	2	35	2	14	131.47	102	525.88	120	57685.2456	44755.0848	230740.983	52653.0409	0.91276444	0.22819111	0.45638222	0.22819111	-178087.94	0.875	0.2147	263703.98	70
37	2	36	3	21	75.13	3.6	601.01	123.6	32962.9975	1579.59123	263703.98	54232.6322	0.94014737	0.18802948	0.31338246	0.20565724	-209471.35	1	0.2211	263703.98	63
38	2	37	4	28	0	42	601.01	165.6	0	18428.5643	263703.98	72661.1965	1.25961493	0.20933582	0.31480373	0.27554077	-191042.78	1	0.2962	263703.98	56
39	2	38	5	35	0	21.55	601.01	187.15	0	3455.6086	263703.98	82116.8051	1.42353221	0.20336174	0.28470644	0.31139767	-181587.17	1	0.3348	263703.98	49
40	2	39	6	42	0	0	601.01	187.15	0	0	263703.98	82116.8051	1.42353221	0.17794153	0.23725537	0.31139767	-181587.17	1	0.3348	263703.98	42
41	2	40	7	49	0	232.1	601.01	419.25	0	101839.757	263703.98	183956.562	3.18897076	0.35433009	0.45556725	0.69758735	-73747.418	1	0.75	263703.98	35
42	2	41	8	56	0	17.25	601.01	436.5	0	7568.87463	263703.98	191525.436	3.32018065	0.33201807	0.41502258	0.72628952	-72178.544	1	0.7809	263703.98	28
43	2	42	9	63	0	53.5	601.01	490	0	23474.4808	263703.98	214999.917	3.72712146	0.33882922	0.41412461	0.81530782	-48704.063	1	0.8766	263703.98	21
44	2	43	10	70	0	30	601.01	520	0	13163.2602	263703.98	228163.177	3.95531257	0.32960938	0.39553126	0.86522463	-35540.803	1	0.9302	263703.98	14
45	2	44	11	77	0	9	601.01	529	0	3948.97807	263703.98	232112.155	4.04159734	0.3108921	0.36741794	0.88019367	-31591.825	1	0.9463	263703.98	7
46	2	45	12	84	0	30	601.01	559	0	13163.2602	263703.98	245275.416	4.44093178	0.31720941	0.37007765	0.93011647	-18428.564	1	1	263703.98	0

Figure 5.2 Example of preprocessed data spreadsheet – Excavation Work Package.

5.2.2 Data Splitting

The collected dataset was divided into two sets: the training and validation datasets. After several experiments, the splitting values were obtained based on the optimal accuracy of the deep-learning model. Then, to ensure that the deep learning model could perform well on unseen data, the data augmentation technique was applied to obtain the test dataset.

Data augmentation for test dataset:

As the time series dataset from available projects was stored per Work Package (i.e. Concrete, Excavation and Backfill), it enables users to find specific behaviour patterns from each. Typically, progress control parameters form an S curve (Cristóbal, 2017; Mubarak, 2019). In this sense, the logistic function was chosen to characterize the Planned Value (PV). It represents growth that starts exponentially but eventually slows down because it approaches a maximum limit due to resource limitations, productivity variations, or other project factors. Also, the logistic function parameters needed, such as the maximum value that the function can take (L), growth rate (k) and

the x-value of the inflection point (X0), were grasped from existing data by studying the behaviour of every Planned Value's S curve through an automated model in Python.

The Earned Value (EV) curves were based on previous research closely related to the characteristics of the available projects. Thus, the Earned Values' S curves were created using polynomial functions and neural networks (Chao & Chien, 2009). Typical occurrences in real-life projects, like delays or early starts concerning the planned start date, were also considered. It was achieved by assigning a random quantity of periods (positive or negative) considered as slip periods. Subsequently, additional indicators, such as the Earned Schedule (ES) and the Time Performance Index (TPI), were calculated based on the estimated value (EV) of the project and the planned value (PV) and the ES and the actual time that has elapsed, respectively.

Furthermore, the budget was generated randomly within the range of the original project budgets. After the new data was collected, it was thoroughly reviewed and validated. Similarly, the overall data augmentation process was computerized using the *curve_fit* function from Python's library SCIPY. To create synthetic time series data in the implemented Python program, practitioners must enter the number of new projects to generate. The code will return a CSV file holding the specified number of projects, each containing a random number of timestamps (the project progress periods) as rows and the predictors as columns. Figure 5.3 is a visual representation of the data augmentation process for this study.

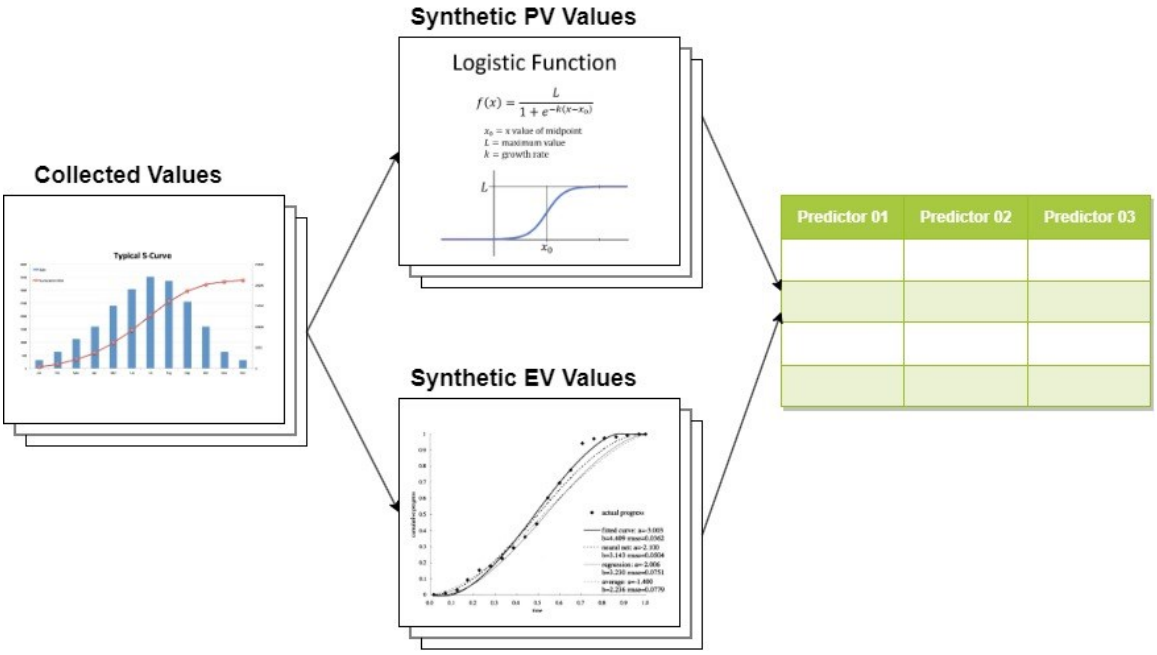


Figure 5.3 Schematic of Process for Data Augmentation for the Present Study

A new project with Concrete, Excavation, and Backfill work packages was created containing 33, 27, and 21 records, respectively. The table below shows the optimal split percentages and the number of records used in the test datasets.

Table 5.1 Training, Validation and Test Datasets per Work Package and LSTM, CONV-1D and MLP Algorithms

Work Package	**Datasets**	**Deep Learning Algorithms**		
		LSTM	**CONV-1D**	**MLP**
Concrete	Training (%, # records)	70% (121)	70% (121)	75% (129)
	Validation (%, # records)	30% (52)	30% (52)	25% (44)
	Test (# records)	33	33	33
Excavation	Training (%, # records)	60% (39)	60% (39)	60% (39)
	Validation (%, # records)	40% (25)	40% (25)	40% (25)
	Test (# records)	27	27	27
Backfill	Training (%, # records)	70% (60)	70% (60)	70% (60)
	Validation (%, # records)	30% (25)	30% (25)	30% (25)
	Test (# records)	21	21	21

5.2.3 Data normalization

The Min-Max normalization was chosen for this case study because of the optimal dataset arrangement to leverage the machine learning features as explained in section 4.2. Each work package dataset contains multiple projects and is structured in a tabular way, where rows represent the timestamps and columns represent the predictor and target variables (see Figure 5.2 as reference). Therefore, to keep each project's characteristics, the min-max technique follows this equation: $Scaled\ values = \frac{x - x_{min}}{x_{max} - x_{min}}$, which reduces the scale of original values without changing the graphical distribution. The other alternative was the z-score normalization, which fed the mean and standard deviation. However, given the data arrangement (multiple projects in series), it would alter the dataset and its graphical distribution. The z-score equation is expressed by $Z = \frac{x - \mu}{\sigma}$ (where μ is the mean and σ represents the standard deviation).

5.2.4 Forecasting model

The algorithms architecture used in the forecasting model was built using Keras, an open-source neural network library for Python. Keras is renowned for enabling rapid experimentation with deep neural networks, designed to be user-friendly, modular, and easily extendable. The library's core data structures are models and layers, and it provides a Sequential model for linear stacking of layers and a functional API (more flexibility compared to Sequential API) for building complex model architectures. Nine forecasting models were created because of the three evaluating algorithms (LSTM, CONV-1D, and MLP) and for the three work packages assessed (Concrete, Excavation, and Backfill). It is crucial to look into the number of past timestamps to predict. Initially, 4, 5 and 6 past timestamps were considered to form the batches; however, after experimentation, when inputting three timestamps demonstrated the best performance. The selection of the optimal value depends on the dataset characteristics and the amount of available

data. Similarly, the hyperparameters were selected using a trial-and-error process. The resulting accuracy demonstrated being highly sensitive to the hyperparameters modification during the learning stage. For instance, the learning rate selection depends on the optimizer utilized, which for this study was ADAM; therefore, the learning rate varies from 0.001 to 0.01. If the SGD (Stochastic Gradient Descent) is used as an optimizer, values will vary between 0.01 and 0.1. Finally, the tables below show values for each algorithm per work package.

Table 5.2 LSTM Hyperparameters used in the forecasting model per Work Package

LSTM Hyperparameters	**Description**	**Work Packages**		
		Concrete	**Excavation**	**Backfill**
Number of Layers	Determines the depth of the network, affecting its complexity and capacity to learn patterns.	3	3	3
Number of neurons	Describe the total quantity of units that process sequence data	64	64	64
Learning Rate	Controls the step size during optimization, influencing the convergence speed and accuracy.	0.001	0.01	0.005
Number of Epochs	It is the total number of passes through the entire training dataset.	200	200	200
Batch Size	It is the number of samples processed before the model updates its weights.	8	4	8
Recurrent Dropout	The percentage of dropped recurrent connections was randomly selected to prevent overfitting.	0.30	0.05	0.30

Table 5.3 CONV-1D Hyperparameters used in the forecasting model per Work Package

CONV-1D Hyperparameters	**Description**	**Work Packages**		
		Concrete	**Excavation**	**Backfill**
Number of Convolutional Layers	Determines the depth of feature extraction.	1	1	1
Number of filters	Specifies the number of filters (or kernels) in the convolutional layer	32	32	32

Kernel Size	Size of the filters used in convolutional layers.	3	3	3
Number of dropout layers	Serve as a regularization technique to prevent overfitting	1	1	1
Dropout	Percentage of ignored neurons to avoid overfitting	0.35	0.05	0.25
Number of dense layers	Used to increase the model's complexity and learning capacity.	2	2	2
Number of neurons on dense layers	Used to learn different aspects of the input it receives from the previous layer	50 (first layer) and 1 (second layer)	50 (first layer) and 1 (second layer)	60 (first layer) and 1 (second layer)
Learning Rate	It affects how quickly the network updates its parameters.	0.001	0.0001	0.001
Number of Epochs	The total number of training cycles.	150	120	200
Batch Size	Number of samples processed before the model is updated.	32	8	8

Table 5.4 MLP Hyperparameters used in the forecasting model per Work Package

MLP Hyperparameters	**Description**	**Work Packages**		
		Concrete	**Excavation**	**Backfill**
Number of Hidden Layers	Influences the model's ability to capture complex relationships.	2	2	2
Number of Neurons per Layer	Determines the width of the network.	10 (First layer) and 100 (second layer)	10 (First layer) and 40 (second layer)	10 (First layer) and 30 (second layer)
Activation Function	Such as ReLU, Sigmoid, or Tanh, used in neurons.	ReLU	ReLU	ReLU
Learning Rate	Impacts the convergence speed during training.	0.001	0.001	0.0005
Number of Epochs	The total round of training the network undergoes.	200	250	150
Batch Size	The quantity of data samples used in one iteration.	8	8	32

5.2.5 Forecasting Models performance assessments.

After setting up the model architecture, their performance is evaluated during the learning process, using the training and validation datasets, and analysing the results on the *Loss Curve*. Then, it is calculated the performance metrics (MAE, MASE, and sMAPE) for all the datasets, including testing dataset to assess the model on unseen data. After that, R-squared curves to analyse the relationship between observed and predicted values are elaborated for the unseen dataset. The Figures 5.4, 5.5 and 5.6 shows the "Loss curves" per algorithm per work package.

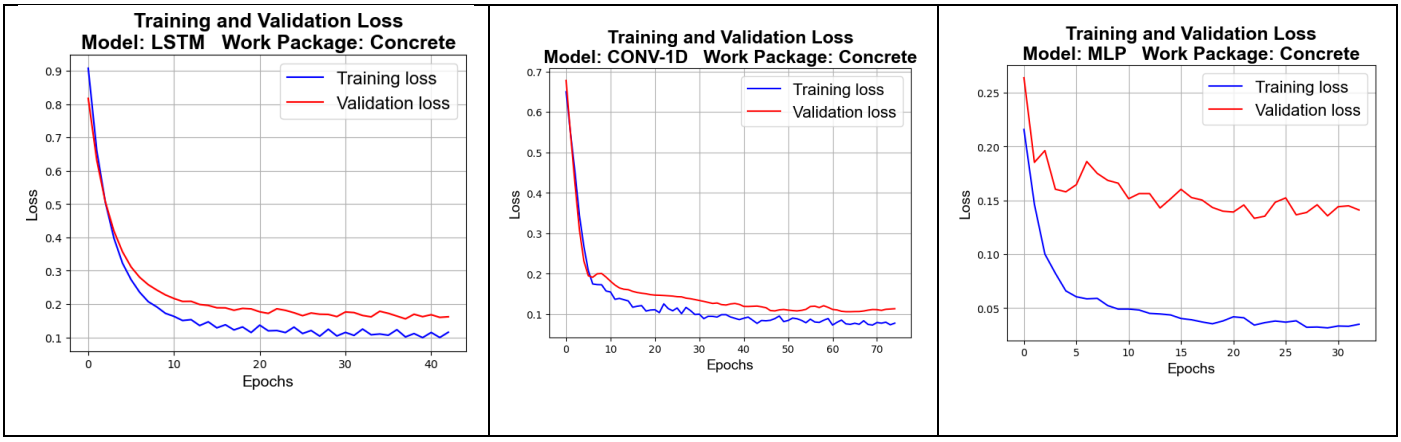


Figure 5.4 Work Package Concrete loss curves per deep learning algorithm.

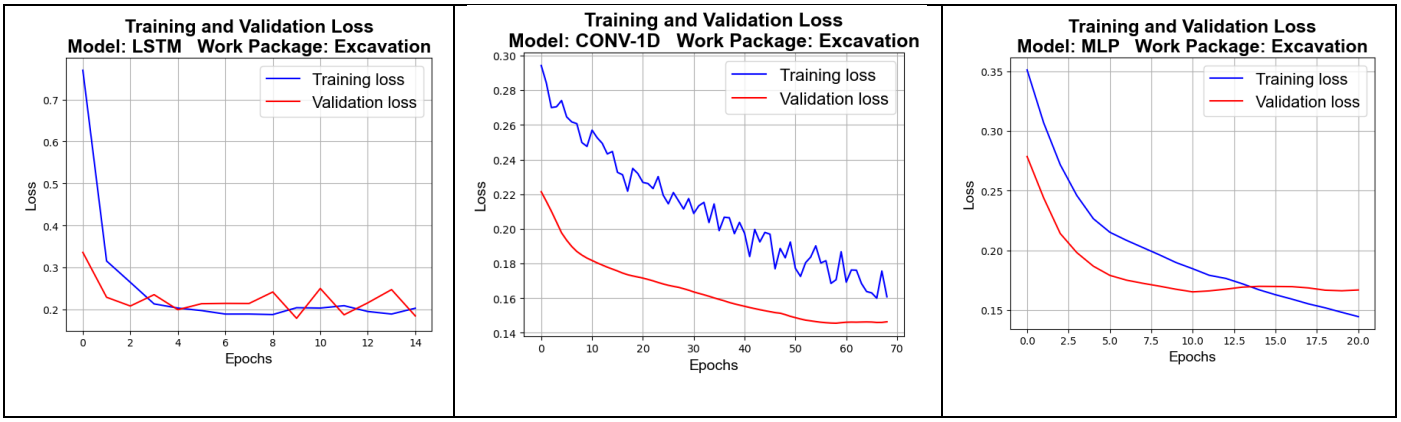


Figure 5.5 Work Package Excavation loss curves per deep learning algorithm.

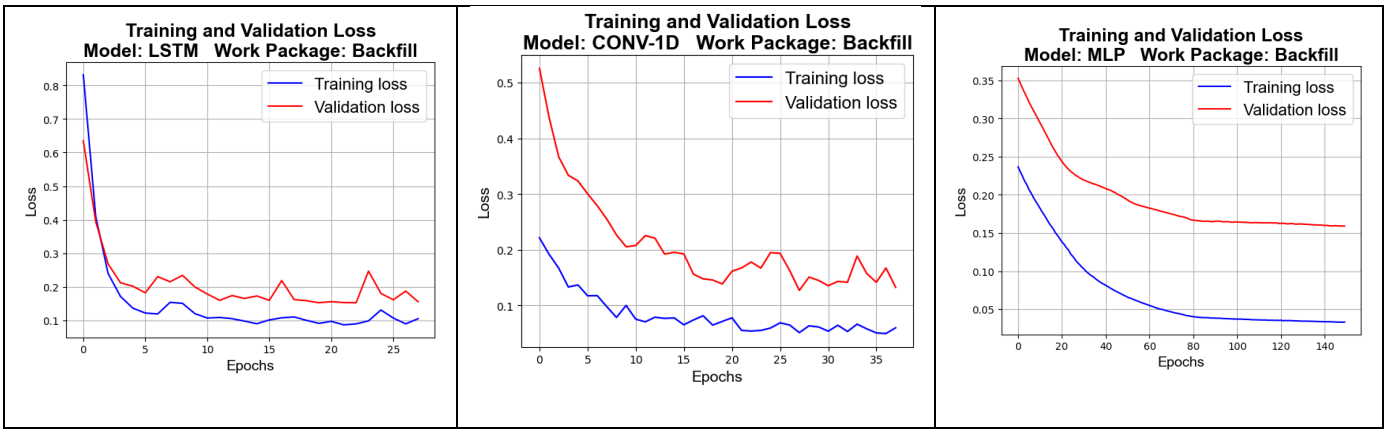


Figure 5.6 Work Package Backfill loss curves per deep learning algorithm.

For the Concrete work package, the LSTM model displays a training loss that sharply decreases and then plateaus, while the validation loss is slightly higher but also reaches a plateau. This interaction between the training and validation loss curves could indicate good generalization after a certain number of epochs. In contrast, the CONV-1D and MLP models show more fluctuations in validation loss, which might suggest less stability. Therefore, the LSTM's smoother convergence is considered the best performer for the Concrete work package.

In the Excavation work package, the LSTM model quickly reduces loss and shows less overfitting as the epochs increase. Also, training and validation losses converge closely, which differs from the CONV-1D and MLP models, where the validation loss tends to diverge as epochs increase. Thus, the LSTM model performs better for the Excavation work package.

In the Backfill work package, the LSTM model again shows rapid initial learning and consistent validation loss, indicative of learning stability and good generalization. On the other hand, the different models, particularly the MLP, appear to overfit, as indicated by increasing validation loss after a certain point. Therefore, the LSTM would be considered the best for the Backfill work

package. The LSTM model consistently shows the best performance across all three work packages, with the least overfitting and the lowest validation loss.

Similarly, the performance metrics per algorithm per work package are depicted in the subsequent tables. The metrics considered were MAE, MASE, and sMAPE, as they are more fit when handling Time Series datasets for forecasting, as explained in Chapter 4. RMSE was disregarded because the target variable (Duration to Complete) will become zero at any time, affecting the consistency of the RMSE result. Although the following tables also show training and validation performance metrics, it is essential to analyze the testing ones to evaluate the model performance over unseen datasets.

Table 5.5 Performance Metrics for the Concrete Work Package per algorithm.

Metric	**LSTM**			**CONV-1D**			**MLP**		
	Training	Validation	Testing	Training	Validation	Testing	Training	Validation	Testing
MAE	12.85	12.39	**16.42**	12.76	10.36	**18.67**	8.49	13.07	**15.61**
MASE	0.21	0.44	**0.27**	0.21	0.37	**0.31**	0.15	0.49	**0.27**
sMAPE	40.14%	52.45%	**42.3 %**	36.05%	46.38%	**33.26%**	31.58%	56.99%	**38.35%**

Table 5.6 Performance Metrics for the Excavation Work Package per algorithm.

Metric	**LSTM**			**CONV-1D**			**MLP**		
	Training	Validation	Testing	Training	Validation	Testing	Training	Validation	Testing
MAE	14.71	9.82	**17.14**	14.31	9.17	**41.45**	15.18	10.41	**48.87**
MASE	0.65	0.58	**0.29**	0.63	0.54	**0.71**	0.67	0.62	**0.84**
sMAPE	81.49%	76.75%	**29.73%**	76.45%	75.38%	**54.81%**	77.45 %	78.38 %	**64.57%**

Table 5.7 Performance Metrics for the Backfill Work Package per algorithm.

Metric	**LSTM**			**CONV-1D**			**MLP**		
	Training	Validation	Testing	Training	Validation	Testing	Training	Validation	Testing
MAE	12.08	7.15	**18.27**	5.47	7.12	**23.20**	8.91	4.87	**20.66**
MASE	0.33	0.52	**0.54**	0.16	0.56	**0.73**	0.70	0.15	**0.65**
sMAPE	51.42%	46.7%	**48.63%**	58.43%	40.38%	**51.27%**	61.94%	59.72%	**47.37%**

Given these results, the model performance on unseen data using the LSTM algorithm depicts the best results on MAE, MASE and sMAPE metrics. Consequently, the LSTM will perform better when handling upcoming project datasets. The adjusted R-squared can also provide some insights into model performance despite not typically being the primary metric for evaluating deep learning models in time series regression problems.

The charts below are displayed per Concrete, Excavation and Backfill work packages, respectively.

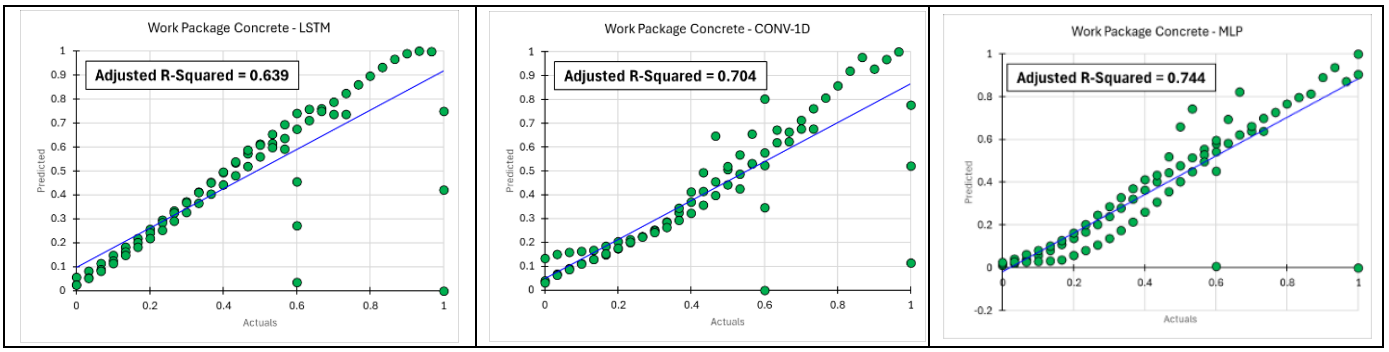


Figure 5.7 Work Package Concrete adjusted R-squared per LSTM, CONV-1D and MLP.

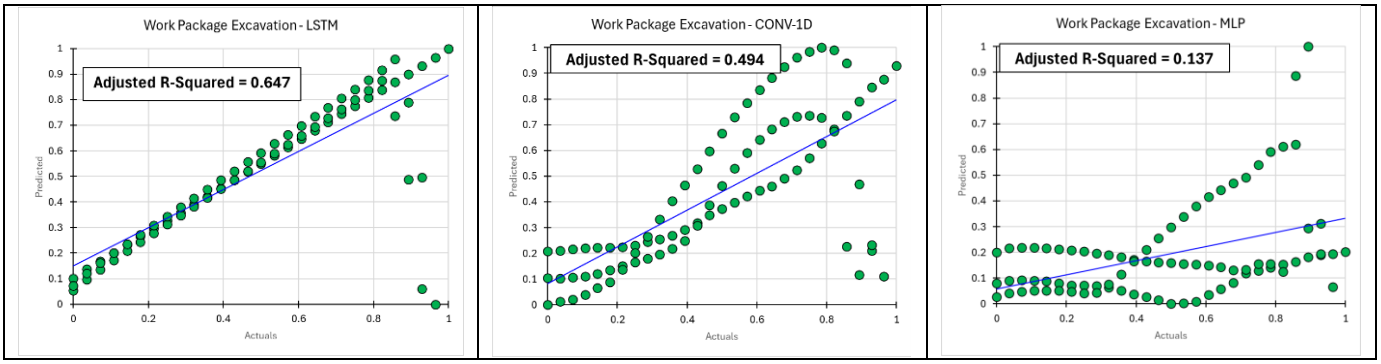


Figure 5.8 Work Package Excavation adjusted R-squared per LSTM, CONV-1D and MLP.

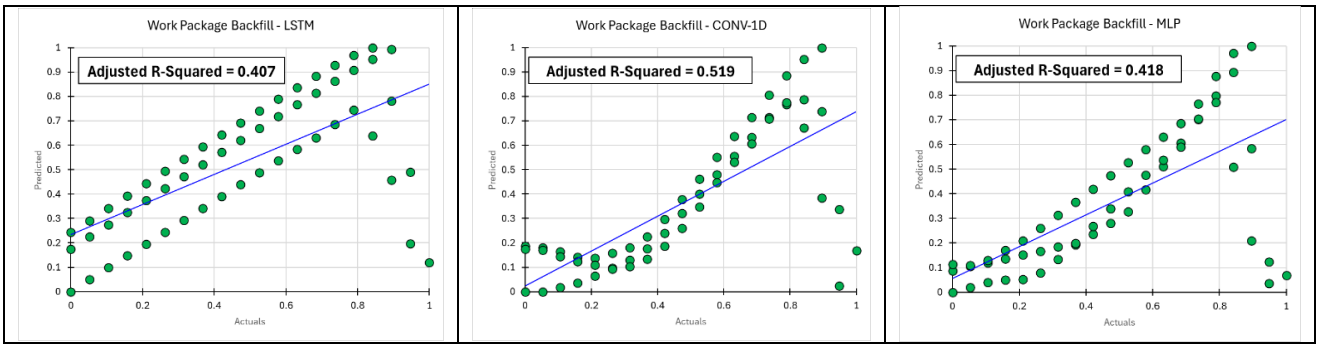


Figure 5.9 Work Package Backfill adjusted R-squared per LSTM, CONV-1D and MLP.

The charts below compare the yielded predictions per period and per algorithm against the actual time completion per work package of the test dataset. Because of the model predicts the Duration to Complete (DTC), it is then added to the Actual Time (AT) elapsed to obtain the Duration at Completion (DAC), as indicated below.

$$DAC = AT + DTC$$

Where DAC=Duration at Completion, AT=Actual Time, and DTC=Duration to Complete.

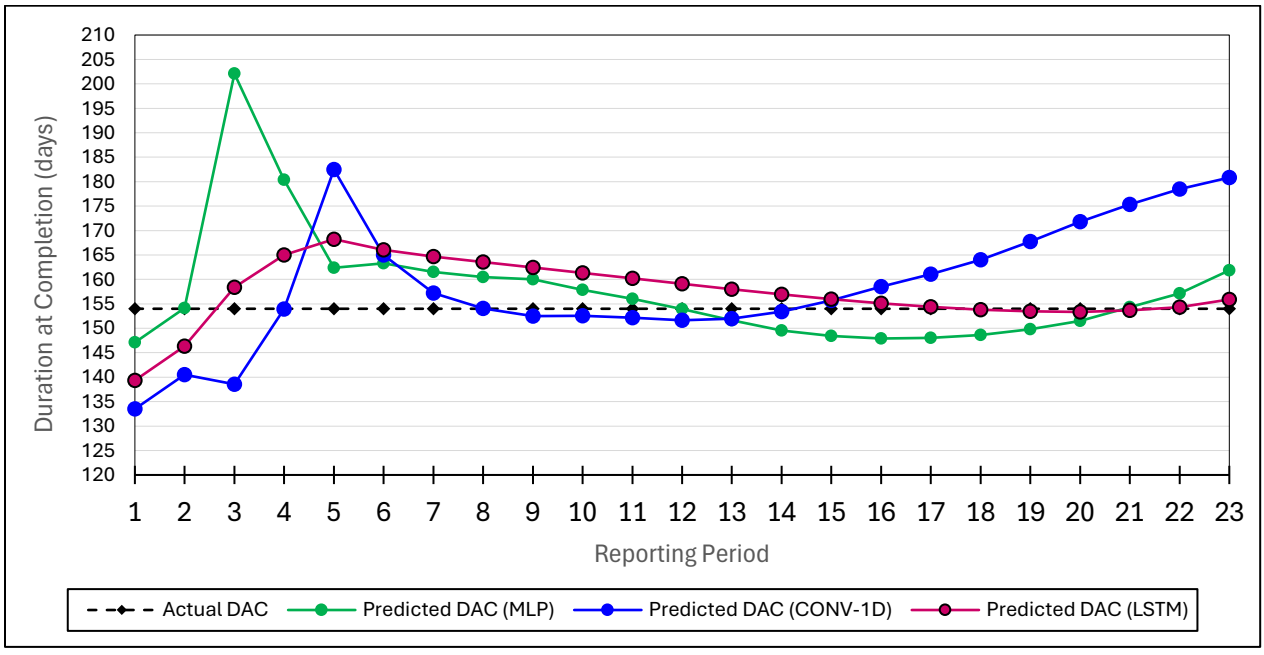


Figure 5.10 Forecasted DAC for Work Package "Concrete" per Period and per Algorithm against Actual Duration.

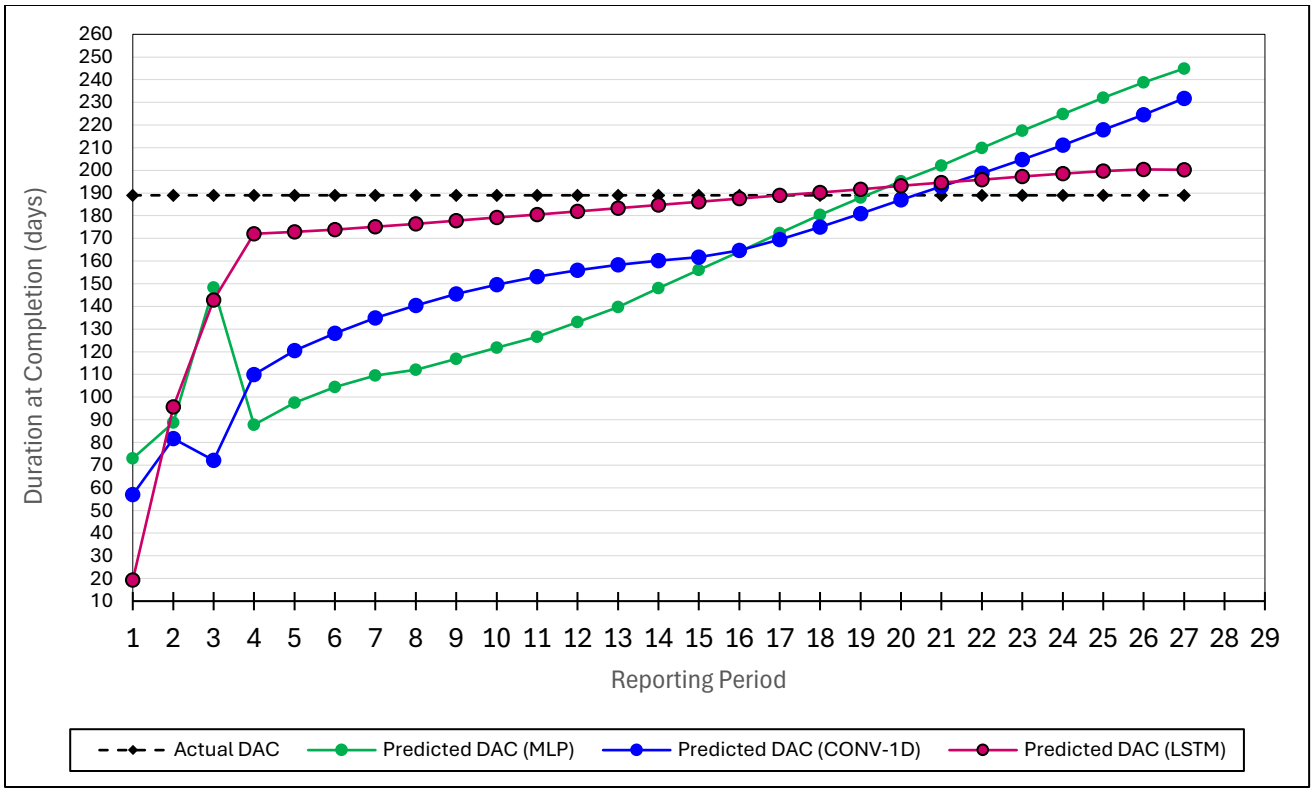


Figure 5.11 Forecasted DAC for Work Package "Excavation" per Period and per Algorithm against Actual Duration.

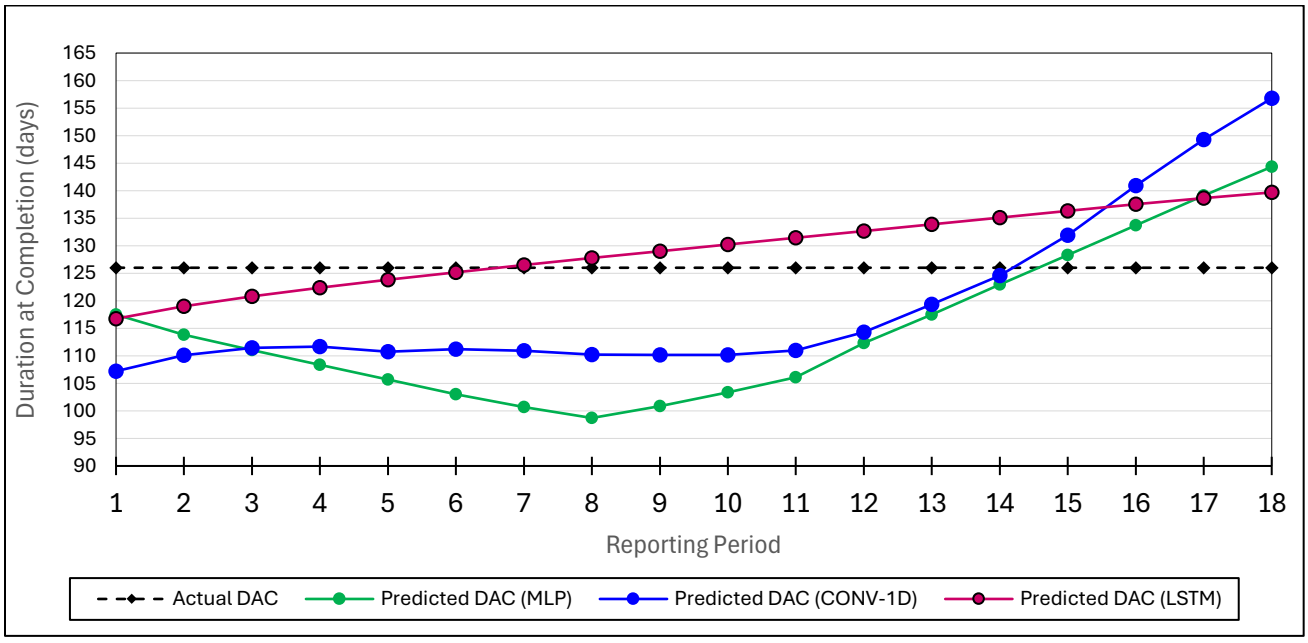


Figure 5.12 Forecasted DAC for Work Package "Backfill" per Period and per Algorithm against Actual Duration.

The LSTMs own inherent advantages that might explain why outperformed CONV-1D and MLP algorithms:

a. Long-Term Dependencies: As discussed earlier, LSTMs remember long-term dependencies through memory cells and gate mechanisms, crucial for time series prediction.
b. Causality: LSTMs capture causal relationships due to their memory past states, essential in time series where past events influence future outcomes.

Also, when comparing only the LSTM models obtained from the different work packages, the Concrete work package had the lowest MASE value (0.27), showing the best performance. Such work package also had more data than the Excavation and Backfill, suggesting that more data leads to more accurate results. However, it's important to consider other factors, such as the construction characteristics of each work package. This includes on-site conditions, such as interferences or lack of preventive equipment maintenance, which can have a greater impact on excavation-related activities than on Concrete work, making the prediction process more challenging.

5.3 Graphical User Interface (GUI) for Project Duration Forecasting

The application of the GUI follows up a high-level pipeline presented in Figure 5.13. It begins by collecting progress data from the work packages, which is used to generate work package predictions using the LSTM algorithm. This algorithm has been evaluated previous section, showing the best performance. Similarly, the GUI collects the project schedule to calculate the Overall Project Forecasting using PDM and CPM methods. As outputs, this GUI delivers two graphical reports at each level of analysis, i.e., work packages and the overall project.

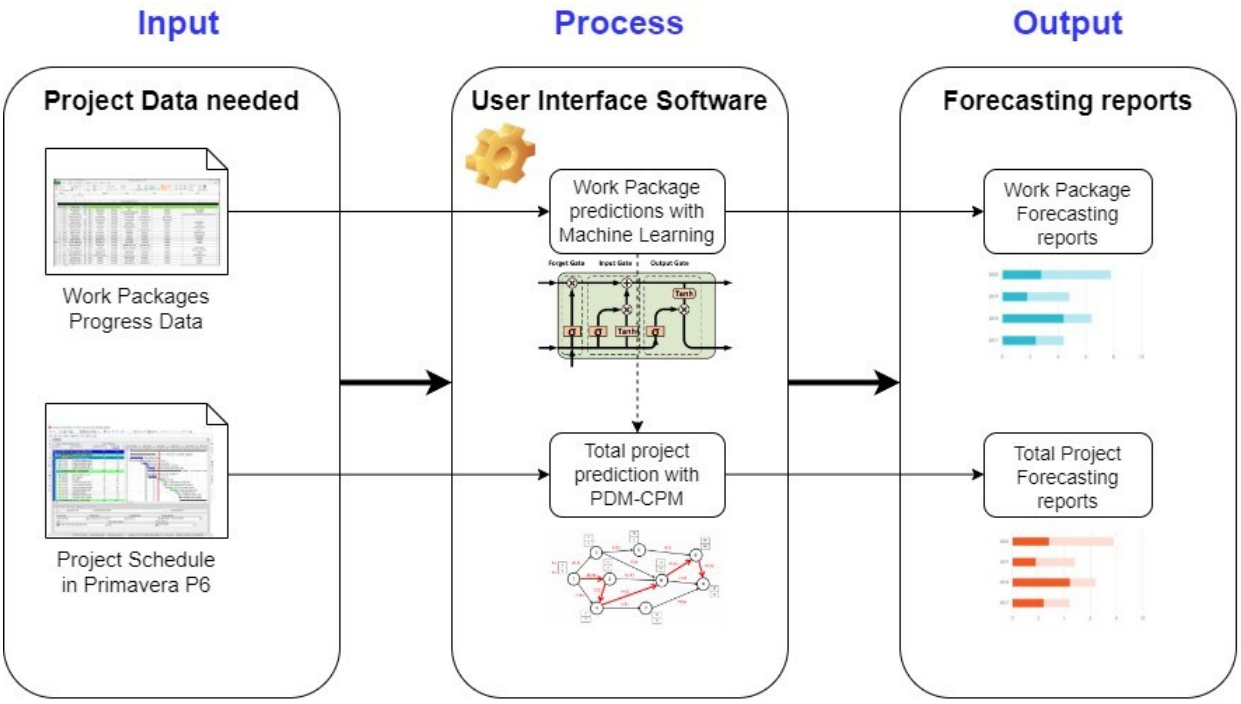


Figure 5.13 Schematic of GUI Functioning

The GUI application uses unseen data obtained from the data augmentation process. This data is from a project containing 03 work packages: Excavation, Concrete and Backfill. The following table summarizes the project information:

Table 5.8 Project information for the Graphical User Interface

Name	Code	Planned Duration (days)	Budget ($)
Excavation	WP-0001	154	1,250,000.00
Concrete	WP-0002	168	1,100,000.00
Backfill	WP-0003	105	1,900,000.00
Total Project	**PJ-9000**	**427**	**4,250,000.00**

Whereas each period is entered (assuming a project tracking during the execution), the predictions per work packages and the overall project duration are calculated. Concerning the project schedule at the work package level, the current GUI version works with Primavera P6, which in subsequent versions may incorporate new features such as linking with MS Project or Asta Powerproject due to the flexibility of Python. Figure 5.14 shows a screenshot of the schedule utilized in this demonstration, considering the three work packages, including start date, finish date, and precedence relationship information. In addition, Figure 5.15 shows its version in XML format.

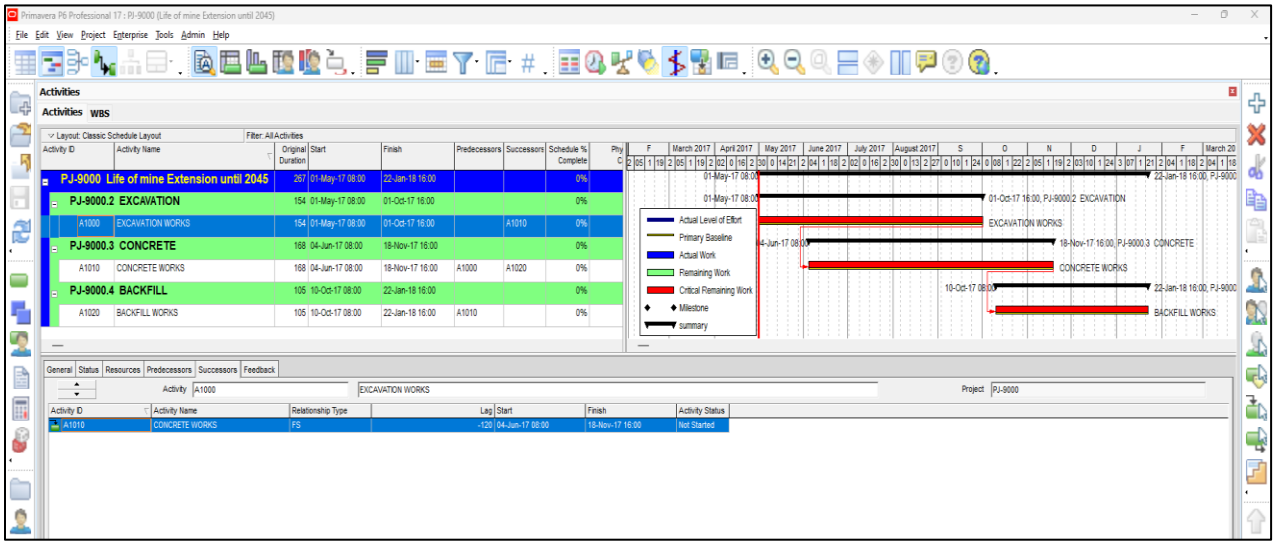


Figure 5.14 Project Schedule at Work Package Level in Primavera P6

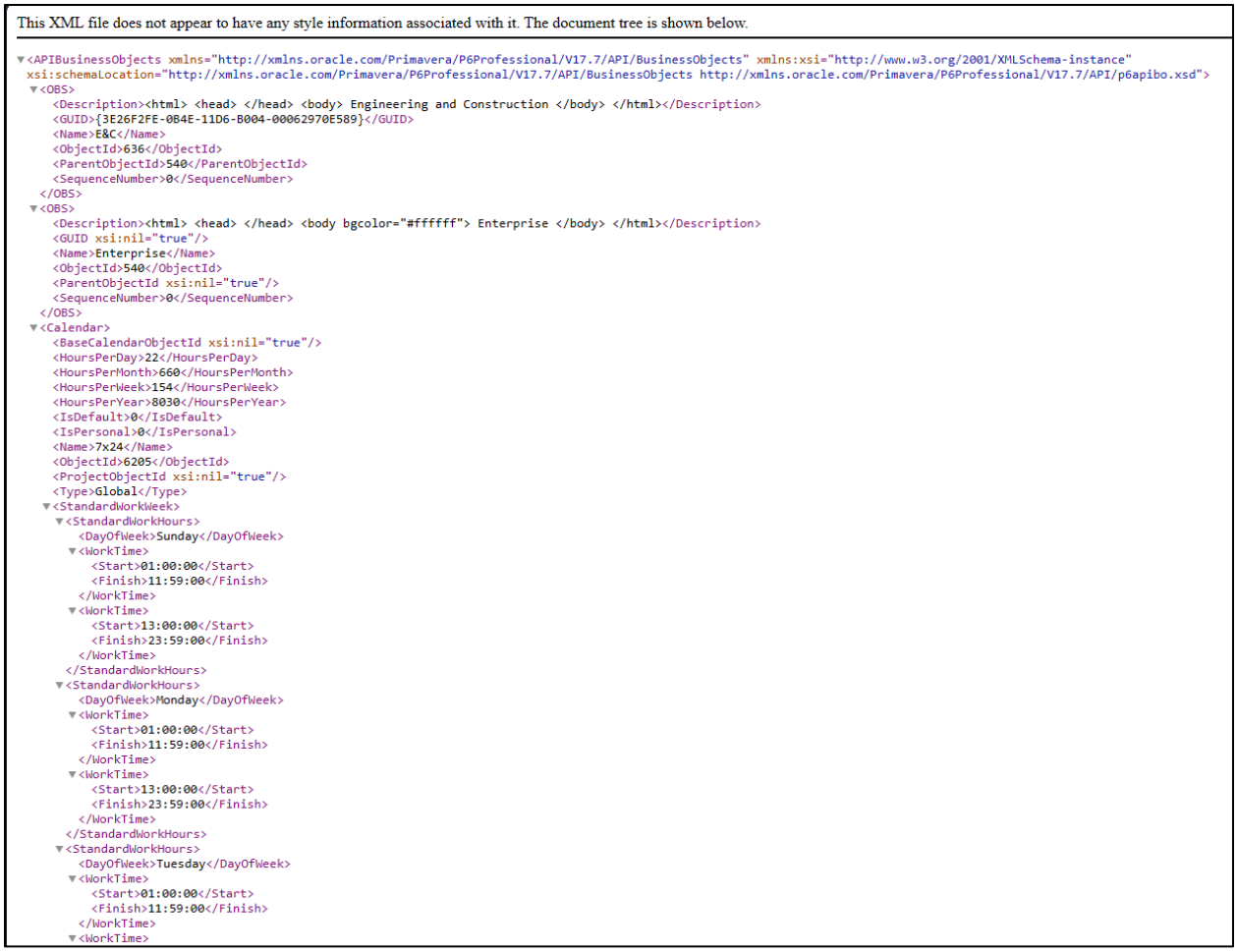


Figure 5.15 Schedule at the Work Package Level in XML format (first lines)

The following sections describe the Graphical User Interface (GUI) application:

1. The user should ensure that the project information is filled in. This task should be done before entering any reporting period. To do so, it should look for it on the hub listed and review its corresponding details on the notebook. If the project does not exist, click the “Add New Project…” button and filled the project information requested in the emerging window.

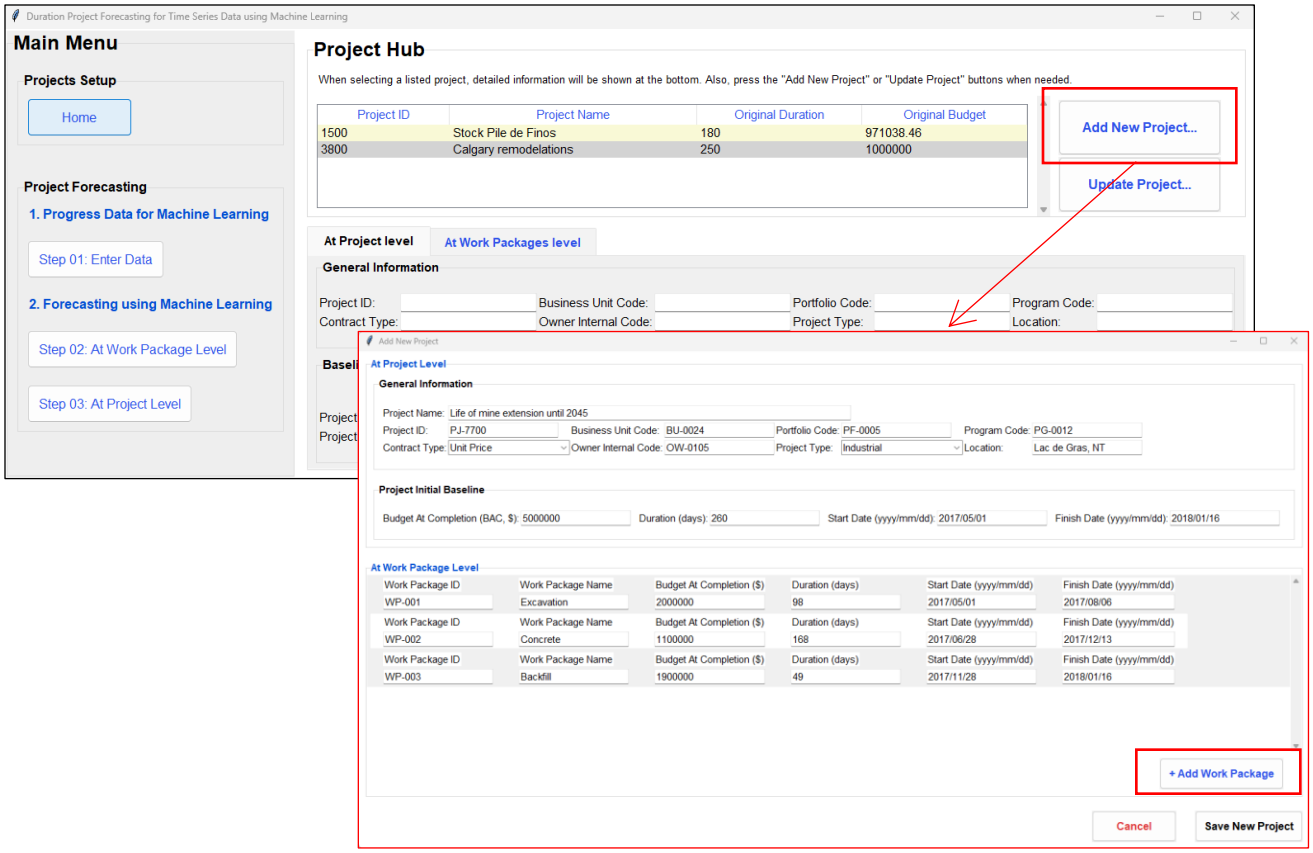


Figure 5.16 Setup of the Project

2. Verifying that the added project is listed on the Project Hub. To do this, click on the "Home" button to refresh the page. After that, the project hub window should look like the image provided below, showing the project PJ-9000.

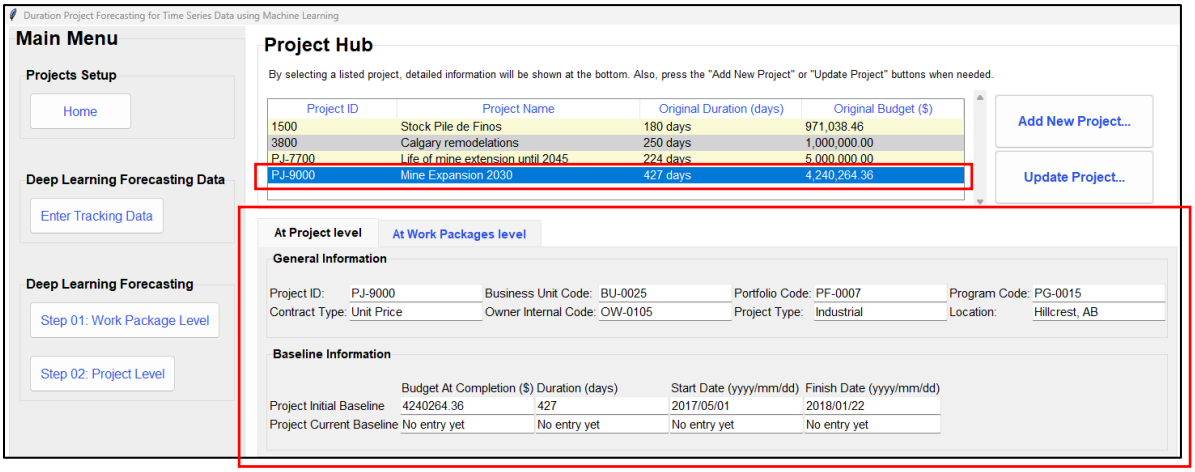


Figure 5.17 The Project Information of “PJ-9000” on the Project Hub

3. Once set up, the user can proceed with the forecasting. On the main menu, click the "Enter Tracking Data" button to enter the work packages' information for the analyzed period. Then, Select the project (PJ-9000) from the dropdown list, entering the ending date, and press "Enter Data…". Once entering the tracking data, press on "Save." This process is repeated per progress period. It was entered until the Reporting Period number 09 (date = 2017-07-03).

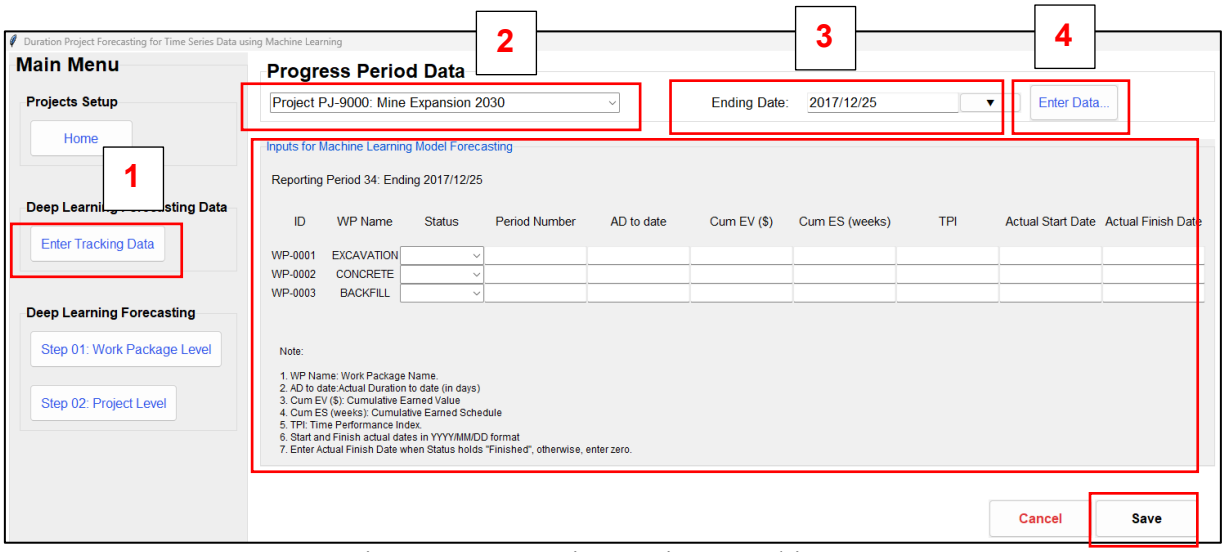


Figure 5.18 Entering Project Tracking Data

4. Then, click "Step 01. Work Package level", selecting the project PJ-9000 and the Reporting Period 09. The Excavation work package's forecasted duration of 63+114=177 days is reported at this data date. Similarly, navigating by the Work Packages buttons will allow users to observe other predictions.

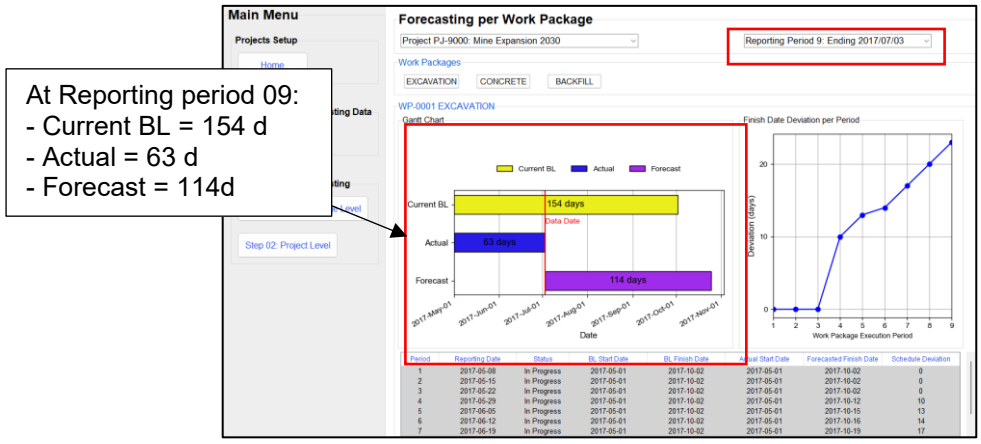


Figure 5.19 Forecasting Report for Excavation Work Package at Period 09

5. Next, click “Step 2: Project Level” to show the overall project prediction. The XML file is loaded and then executed to read the primavera P6 information. It is relevant to note that the information in the GUI matches the information in the source P6 schedule. Next, click Display Duration at the Completion Forecasting button to show the Gantt Chart for the overall project. Finally, it is processed with PDM-CPM to show the overall project duration.

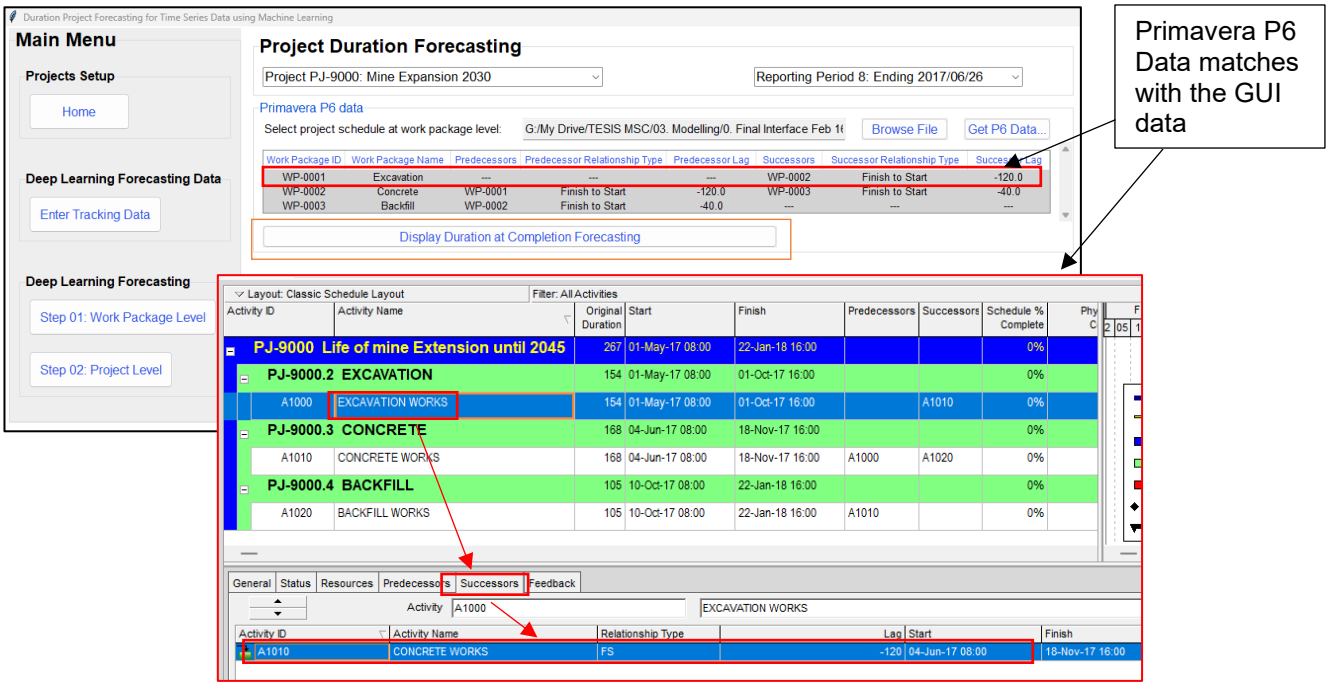


Figure 5.20 Loading Primavera P6 data to the GUI

The charts below show results for periods 08 and 09. It was included the period 08 (Figure 5.21) to present how differs the prediction period by period. Also, it is included the CPM network depicting the critical path per each period.

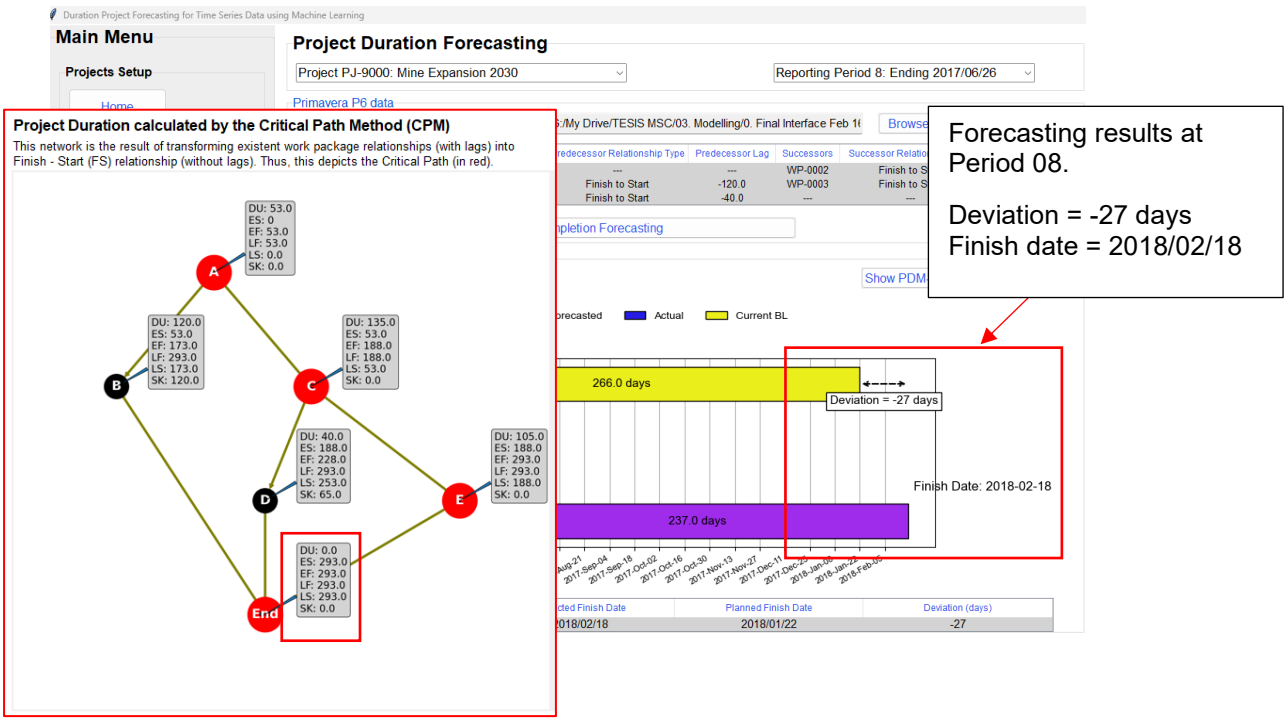


Figure 5.21 Forecasting Report for Overall Project at Period 08

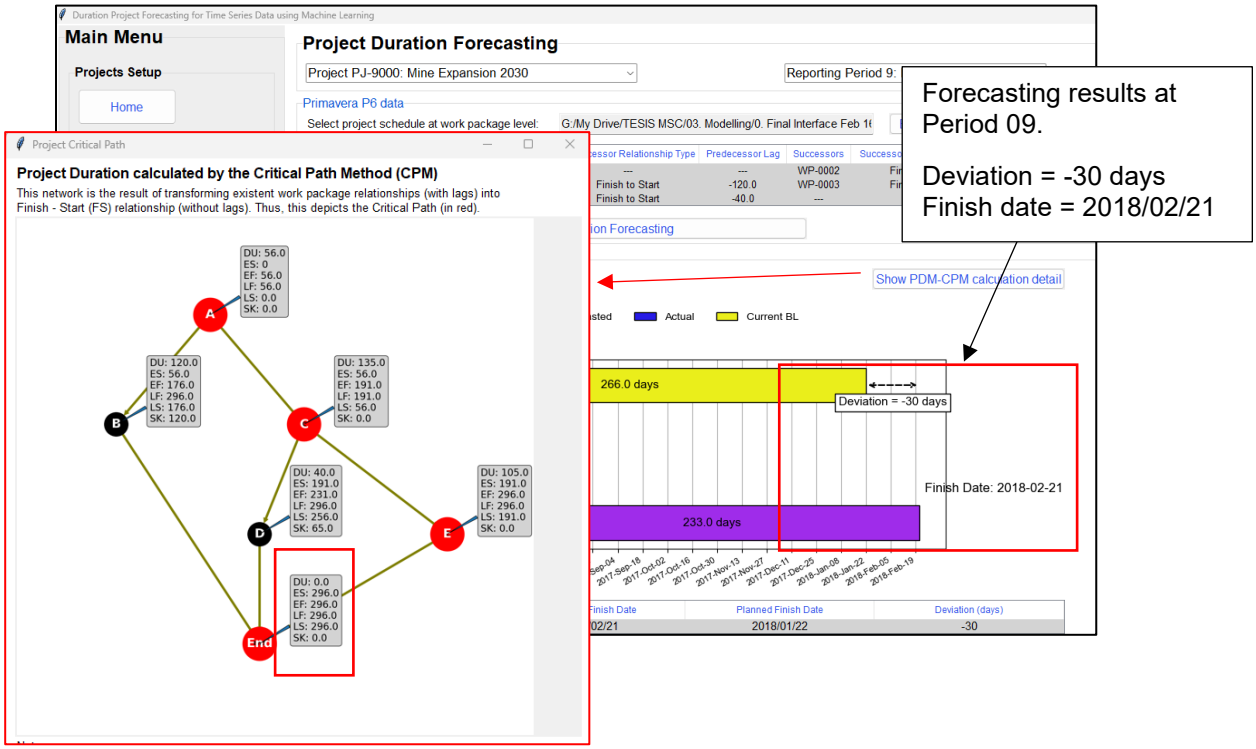


Figure 5.22 Forecasting Report for Overall Project at Period 09

5.4 Comparison with traditional methods for duration prediction

Project managers often rely on the Earned Value methodology (EVM) to forecast project duration. Similarly, another helpful method gaining popularity in project time management is the Earned Schedule Management (ESM) because it is solely calculated using time units-based parameters. These techniques are frequently used in the construction industry and are known for leveraging current performances to achieve forecast outcomes. As a result, they are suitable for comparison with the proposed Deep Learning-based method.

The Duration at completion (DAC) under EVM is calculated following the formula below:

$$\boldsymbol{DAC_{EVM} = \frac{PD}{SPI}}$$

Where DAC_{EVM} = Duration at Completion through EVM; PD = Planned Duration; and SPI = Schedule Performance Index. On the other hand, the DAC obtained using ESM uses formulas such as :

$$\boldsymbol{DAC_{ESM} = AT + \left(\frac{PD - ES}{SPI_t}\right)}$$

Where DAC_{ESM} = Duration at Completion through ESM; AT=Actual Time; PD = Planned Duration; ES = Cumulated Earned Schedule (until analyzed period); SPI_t = Schedule Performance Index in time units (calculated as ES/AT), also called Time Performance index in this research.

5.4.1 Comparison of Deep Learning, EVM and ESM models per work package

Table 5.9 shows the comparison of DAC over the concrete work package dataset (PD=98 days), which also includes the error in percentage per prediction at each period, calculated as:

$$\boldsymbol{Error}\ (\%) = \left(\frac{\boldsymbol{DAC_{Actual} - DAC_{predicted}}}{\boldsymbol{DAC_{Actual}}}\right)\boldsymbol{x100}$$

Where DAC $_{\text{Actual}}$ stands for the actual duration at completion, and DAC $_{\text{predicted}}$ represents the predicted value at each period per model.

Table 5.9 Comparison of Deep Learning model, EVM and ESM for Concrete Work Package

Reporting Period	Duration at Completion (DAC, in days)			Error (in percentage)		
	Deep-Learning	EVM	ESM	Deep-Learning	EVM	ESM
4	168	36	81	-9%	77%	47%
5	166	75	94	-8%	52%	39%
6	165	144	107	-7%	7%	30%
7	164	220	118	-6%	-43%	23%
8	162	251	129	-5%	-63%	16%
9	161	239	140	-5%	-55%	9%
10	160	213	150	-4%	-38%	3%
11	159	188	159	-3%	-22%	-3%
12	158	167	167	-3%	-8%	-9%
13	157	150	174	-2%	2%	-13%
14	156	137	181	-1%	11%	-18%
15	155	127	188	-1%	18%	-22%
16	154	118	194	0%	23%	-26%
17	154	112	196	0%	28%	-27%
18	153	106	199	0%	31%	-29%
19	153	103	197	0%	33%	-28%
20	154	100	195	0%	35%	-26%
21	154	98	179	0%	36%	-16%
22	156	98	154	-1%	36%	0%

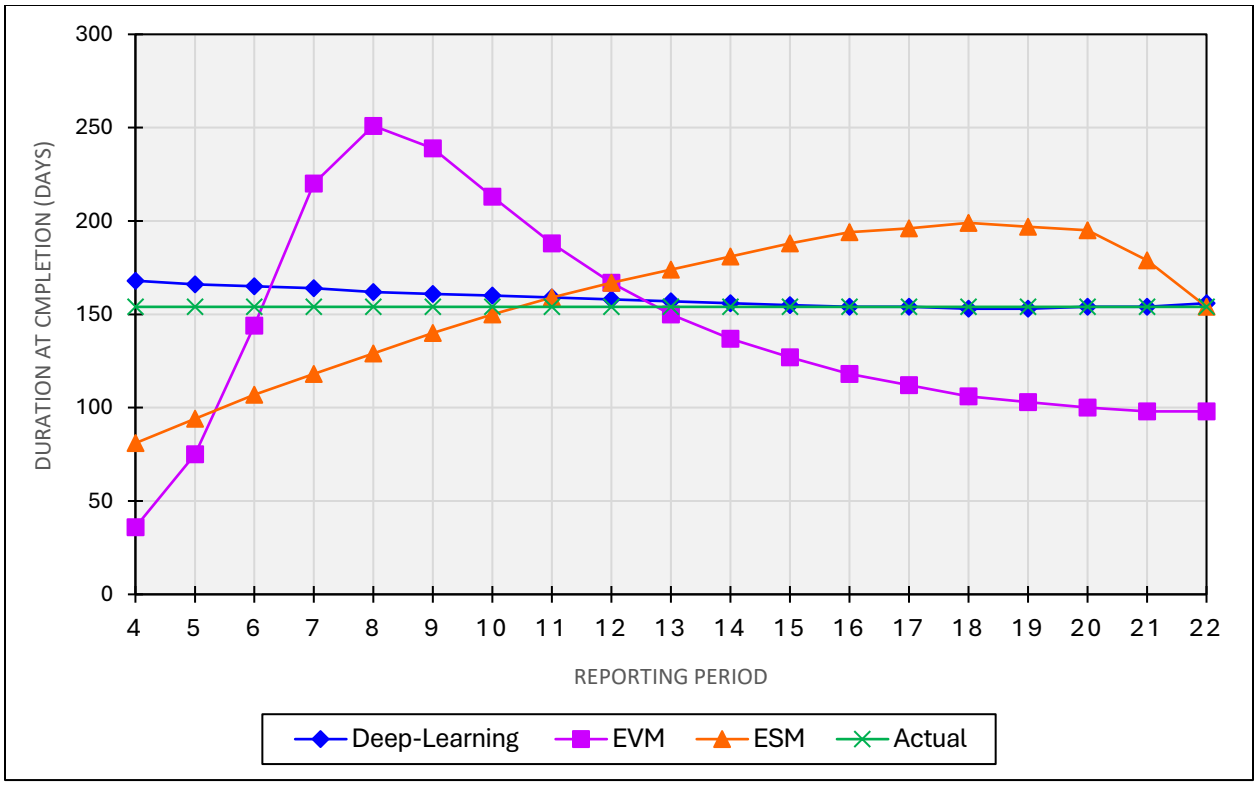


Figure 5.23 Comparison of Deep Learning model, EVM and ESM for Concrete Work Package

Similarly, the comparison analysis for the Excavation work packages (PD=98 days) is shown as follows.

Table 5.10 Comparison of Deep Learning model, EVM and ESM for Excavation Work Package

Reporting Period	Duration at Completion (DAC, in days)			Error (in percentage)		
	Deep-Learning	EVM	ESM	Deep-Learning	EVM	ESM
4	172	54	89	9%	71%	53%
5	173	110	101	9%	42%	46%
6	174	209	114	8%	-11%	40%
7	175	317	128	7%	-68%	32%
8	176	360	139	7%	-90%	26%
9	178	339	149	6%	-79%	21%
10	179	298	160	5%	-58%	15%
11	181	260	171	4%	-37%	10%
12	182	228	181	4%	-20%	4%

13	183	202	190	3%	-7%	-1%
14	185	181	199	2%	4%	-5%
15	186	165	207	2%	13%	-10%
16	188	151	214	1%	20%	-13%
17	189	140	221	0%	26%	-17%
18	190	130	228	-1%	31%	-21%
19	192	123	235	-1%	35%	-24%
20	193	116	238	-2%	38%	-26%
21	195	111	240	-3%	41%	-27%
22	196	107	244	-4%	43%	-29%
23	197	104	243	-4%	45%	-29%
24	199	101	240	-5%	47%	-27%
25	200	99	232	-6%	47%	-23%
26	200	98	214	-6%	48%	-13%
27	200	98	189	-6%	48%	0%

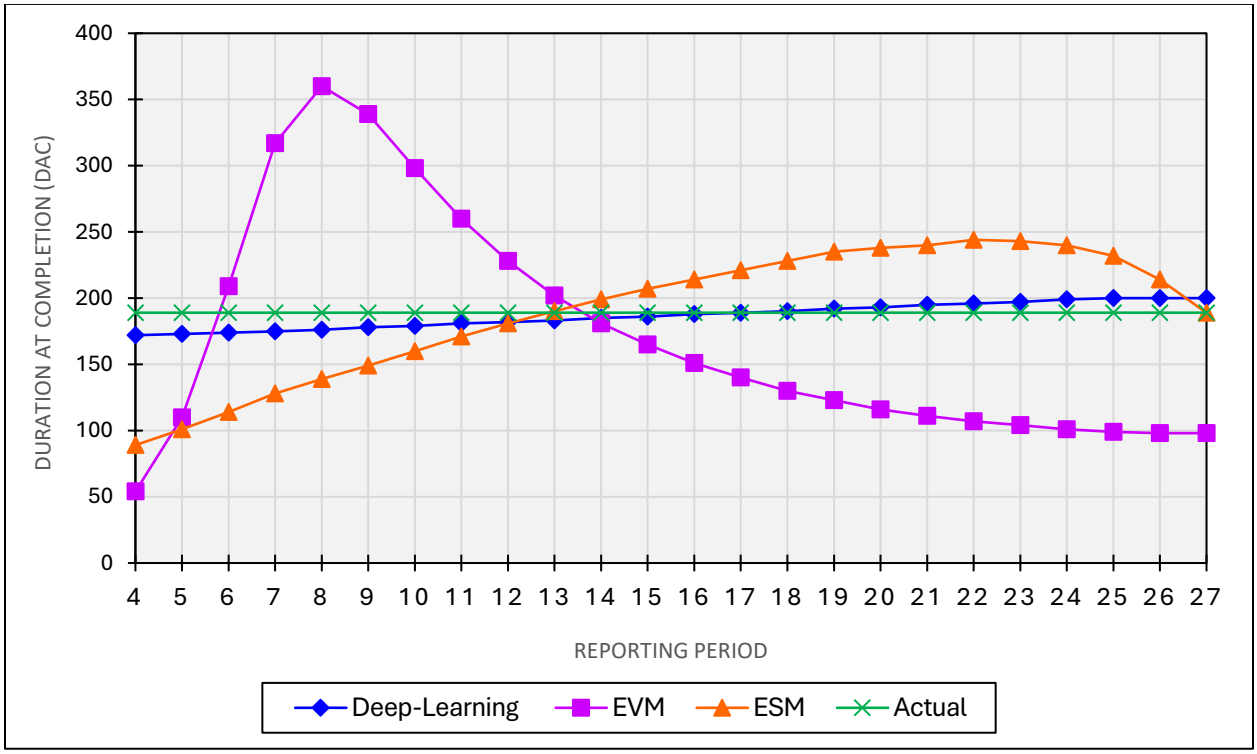


Figure 5.24 Comparison of Deep Learning model, EVM and ESM for Excavation Work Package

Note that predictions have only been available since period 04 because the deep-learning model requires at least three historical records to make predictions. It is also worth mentioning that the EVM method is not the most suitable approach to monitor project time, despite its widespread use. This is because calculating time estimations with money units can be highly variable from one period to the next, leading to inconsistencies in the estimations. As example, the Figure 5.24 shows a lower prediction value by EVM. Below it is presented the comparison analysis for the Backfill work package (PD=84 days).

Table 5.11 Comparison of Deep Learning model, EVM and ESM for Backfill Work Package

Period of forecast	DAC (in days)			Error (in percentage)		
	Deep Learning model	EVM	ESM	Deep Learning model	EVM	ESM
4	138	56	79	6%	62%	46%
5	140	124	93	5%	16%	37%
6	142	213	103	4%	-45%	30%
7	143	249	114	2%	-69%	22%
8	145	231	126	1%	-57%	14%
9	146	200	136	1%	-36%	8%
10	148	173	145	0%	-17%	1%
11	149	151	154	-1%	-3%	-4%
12	150	134	161	-2%	9%	-9%
13	151	121	168	-3%	17%	-14%
14	152	111	175	-4%	24%	-19%
15	154	103	181	-5%	30%	-23%
16	155	97	183	-5%	34%	-25%
17	156	92	185	-6%	37%	-26%
18	157	88	188	-7%	40%	-28%
19	159	86	182	-8%	42%	-24%
20	160	84	170	-9%	43%	-16%
21	161	84	147	-9%	43%	0%

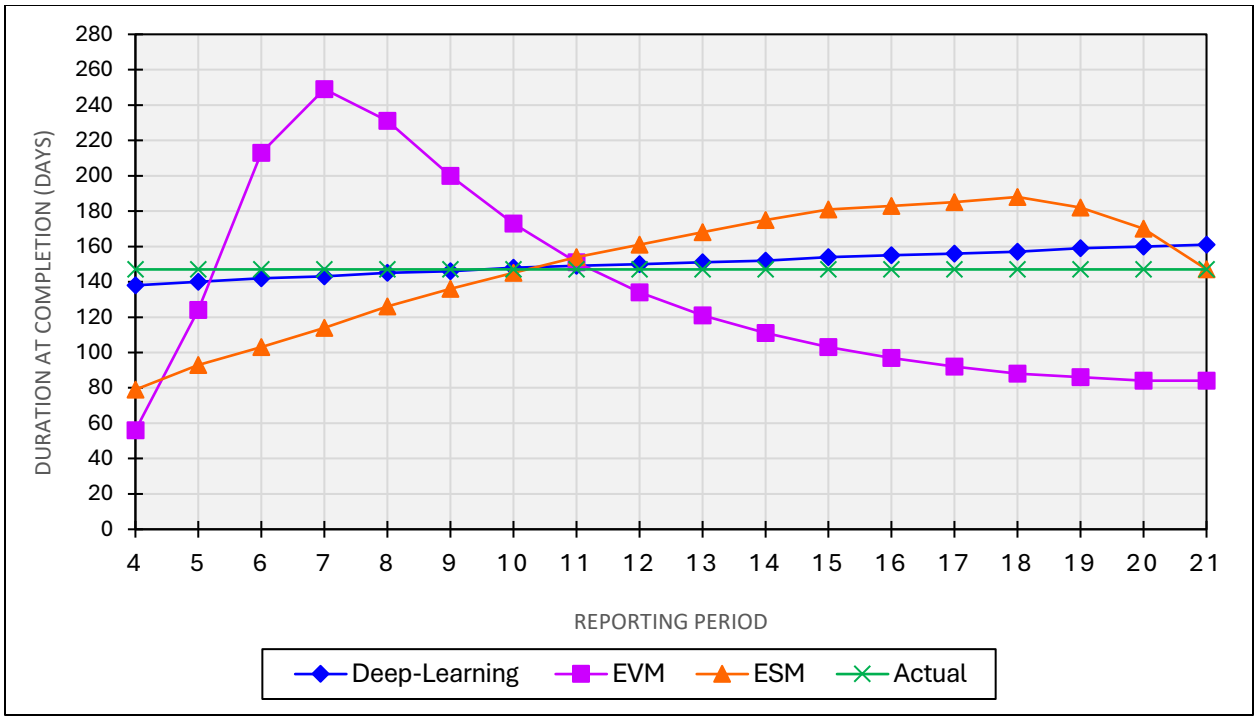


Figure 5.25 Comparison of Deep Learning model, EVM and ESM for Backfill Work Package

Such results where also evaluated using MAE and MAPE, underlining the Deep Learning model is superior to traditional methodologies evaluated. These are showing in the Table below:

Table 5.12 Comparison of Deep Learning model, EVM and ESM for Backfill Work Package

Work Package	Model	MAE	RMSE	Ranking
Concrete	Deep Learning	2.97%	4.14%	1
	ESM	20.30%	23.69%	2
	EVM	32.54%	37.76%	3
Excavation	Deep Learning	4.34%	5.03%	1
	ESM	21.34%	25.13%	2
	EVM	40.50%	46.04%	3
Backfill	Deep Learning	4.38%	5.16%	1
	ESM	19.27%	22.64%	2
	EVM	34.73%	38.90%	3

5.5 Sensitivity Analysis applying Montecarlo Simulation

A sensitivity analysis was conducted to assess how variations in independent variables impact the dependent variable (outcome). Specifically, the analysis focused on project progress data, which encompasses multiple variables per progress period. Consistent values were inputted, considering two key considerations. Firstly, the interdependency between time-sequential records (between timestamps), and secondly, the intrinsic relationship between project progress parameters within each timestamp. This includes relationships between PV (Planned Value) with EV (Earned Value), actual time and progress, actual time, and PV, among others. In time series datasets, both interactions are crucial to consider. Accordingly, the following steps were undertaken for the present study:

1. A random progress period (timestamp) was selected for each work package, incorporating the original predictors.
2. Each independent variable (predictor) underwent adjustment within predefined ranges to generate multiple consistent replicates of this specific progress period, utilizing Monte Carlo Simulation. This process exploited the intrinsic relationships between predictors, as illustrated in Figure 5.26.
3. Monte Carlo simulation involves creating random values for the predictors based on specified distributions or criteria. Subsequently, the resulting outcomes were analyzed to assess sensitivity and uncertainty.
4. Once the reproductions of progress period data were generated, each was seamlessly integrated as record data into the original project tracking dataset, preserving its chronological position.

5. Following integration, the deep learning algorithm was executed multiple times, corresponding to the number of independent variable chains created. This iterative process yielded predictions for Duration to Complete (DTC).
6. The inputs and respective outcomes obtained from the deep learning algorithm iterations were thoroughly analyzed to discern patterns and behavior.

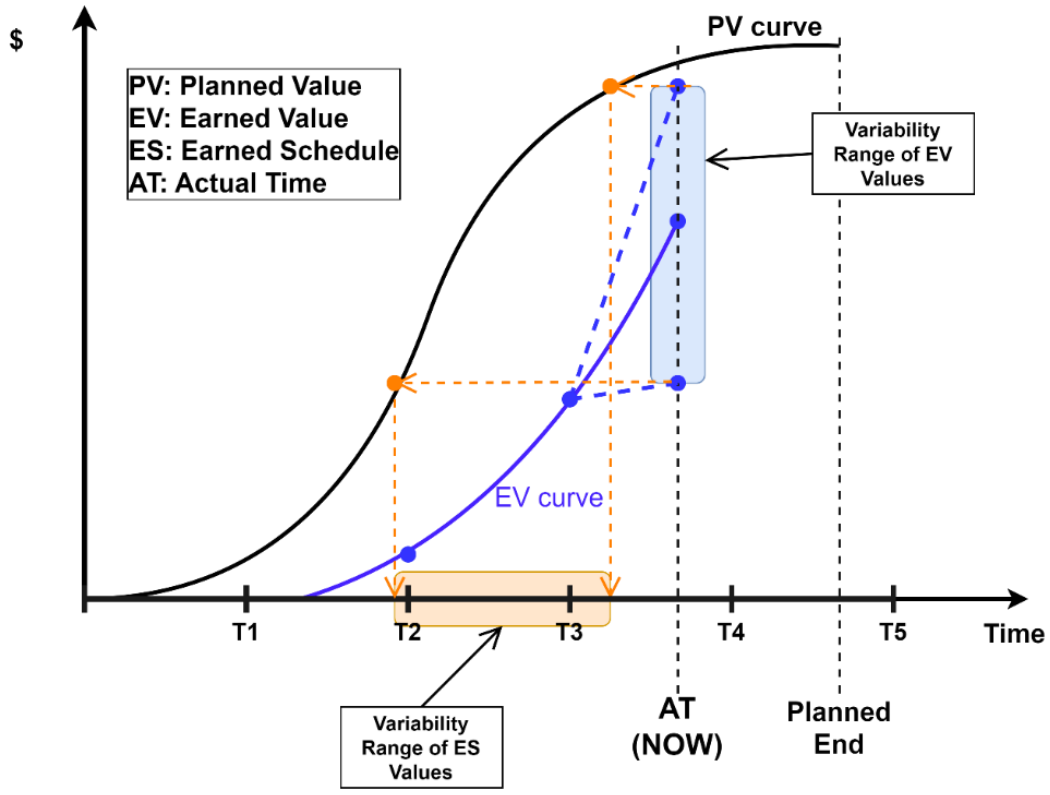


Figure 5.26 Relationship between EV and ES for a Same Period.

The following outlines the sensitivity analysis per work package utilizing the Deep Learning forecasting model.

5.5.1 Analysis of Concrete Work Package Prediction Model

Table 5.13 presents concrete work package parameters, as well as, in Table 5.14 shows the timestamps analyzed, highlighting the fifteen period, which has been selected, and three previous timestamps as the forecasting model needs three past records to predict the future. Range of values greater than $705,323.5 (previous EV) and less than $843,059.0 (following EV) are considered for the EV in the simulations. In total, were conducted 50 simulations.

Table 5.13 Work Package Parameters for Concrete

Work Package Parameters	**Value**
Planned periods	24
Actual periods	26
Current analysed period (randomly selected)	15
Budget (dollars)	1’100,000
DTC $_{15th}$ (predicted with actual values)	178.2

Table 5.14 Concrete Work Package - Records for Sensitivity Analysis

Actual Duration (weeks)	Actual Duration (days)	Cumulative EV ($)	Cumulative Earned Schedule (weeks)	Time Performance Index (TPI)	Contract Amount ($)	Duration To complete (DTC, days)
….						
12	84	563,399.6	9.2	0.768	1’100,000	174.8
13	91	634,467.7	10.8	0.827	1’100,000	176.3
14	98	705,323.5	12.3	0.878	1’100,000	177.4
15	105	**775,132.2**	13.9	0.925	1’100,000	DTC $_{15th}$
16	112	843,059	15.5	0.968	1’100,000	---
….						

As a result, fifty predictor’s replicates of the 15th period were obtained, as detailed in Table 5.15. Following the previous steps explained, the forecasting model was run to predict Duration to

Complete (DTC) considering each new record which have replaced the "fifteen" record within the whole list of timestamps. After that, Figure 5.27 depicts the interaction between independent and target variables. The analysis showed that a variation between -8% and 7% in the EV values represents a variability between -8% and -12% in the target variable (Duration to Complete), respectively. The ES is between -8% and 9% in this scenario.

Table 5.15 Results of Fifty Simulations for Sensitivity Analysis - Concrete Forecasting Model

Independent Variables and Predicted DAC for 15th period					Variations (%)				
Cumulative EV ($)	Cumulative Earned Schedule (weeks)	Time Performance Index (TPI)	Contract Amount ($)	Duration To complete (DTC, days)	EV	ES	TPI	Contract Amount	DTC
783,953.20	14.08	0.94	1,100,000.00	80.9	-1%	-1%	-1%	0%	-10%
811,599.10	14.74	0.98	1,100,000.00	79.8	-5%	-6%	-6%	0%	-9%
823,641.60	15.02	1.00	1,100,000.00	79.3	-6%	-8%	-8%	0%	-8%
727,838.40	12.79	0.85	1,100,000.00	82.9	6%	8%	8%	0%	-13%
781,783.20	14.03	0.94	1,100,000.00	81.0	-1%	-1%	-1%	0%	-11%
771,058.10	13.78	0.92	1,100,000.00	81.4	1%	1%	1%	0%	-11%
795,542.70	14.36	0.96	1,100,000.00	80.4	-3%	-3%	-3%	0%	-10%
731,679.70	12.87	0.86	1,100,000.00	82.8	6%	7%	7%	0%	-13%
814,223.40	14.80	0.99	1,100,000.00	79.7	-5%	-7%	-7%	0%	-9%
813,256.50	14.78	0.99	1,100,000.00	79.7	-5%	-7%	-7%	0%	-9%
808,615.30	14.67	0.98	1,100,000.00	79.9	-4%	-6%	-6%	0%	-9%
816,519.30	14.85	0.99	1,100,000.00	79.6	-5%	-7%	-7%	0%	-9%
731,523.00	12.87	0.86	1,100,000.00	82.8	6%	7%	7%	0%	-13%
752,780.40	13.36	0.89	1,100,000.00	82.1	3%	4%	4%	0%	-12%
809,571.80	14.69	0.98	1,100,000.00	79.9	-4%	-6%	-6%	0%	-9%
785,875.50	14.12	0.94	1,100,000.00	80.8	-1%	-2%	-2%	0%	-10%
771,611.60	13.79	0.92	1,100,000.00	81.4	0%	1%	1%	0%	-11%
748,021.50	13.25	0.88	1,100,000.00	82.3	3%	5%	5%	0%	-12%
764,235.60	13.62	0.91	1,100,000.00	81.7	1%	2%	2%	0%	-12%
739,139.00	13.04	0.87	1,100,000.00	82.6	5%	6%	6%	0%	-13%
784,867.10	14.10	0.94	1,100,000.00	80.8	-1%	-2%	-2%	0%	-10%
749,080.90	13.27	0.88	1,100,000.00	82.3	3%	4%	4%	0%	-12%
749,619.70	13.28	0.89	1,100,000.00	82.3	3%	4%	4%	0%	-12%
822,186.50	14.99	1.00	1,100,000.00	79.4	-6%	-8%	-8%	0%	-8%
814,637.50	14.81	0.99	1,100,000.00	79.7	-5%	-7%	-7%	0%	-9%
750,418.70	13.30	0.89	1,100,000.00	82.2	3%	4%	4%	0%	-12%
724,716.00	12.72	0.85	1,100,000.00	83.0	7%	8%	8%	0%	-13%

727,212.90	12.78	0.85	1,100,000.00	83.0	6%	8%	8%	0%	-13%
741,052.60	13.08	0.87	1,100,000.00	82.6	4%	6%	6%	0%	-13%
741,640.10	13.10	0.87	1,100,000.00	82.6	4%	6%	6%	0%	-13%
815,927.20	14.84	0.99	1,100,000.00	79.6	-5%	-7%	-7%	0%	-9%
803,784.60	14.55	0.97	1,100,000.00	80.1	-4%	-5%	-5%	0%	-9%
755,635.00	13.42	0.89	1,100,000.00	82.0	3%	3%	3%	0%	-12%
748,661.80	13.26	0.88	1,100,000.00	82.3	3%	4%	4%	0%	-12%
820,284.70	14.94	1.00	1,100,000.00	79.5	-6%	-8%	-8%	0%	-9%
756,858.50	13.45	0.90	1,100,000.00	82.0	2%	3%	3%	0%	-12%
812,107.70	14.75	0.98	1,100,000.00	79.8	-5%	-6%	-6%	0%	-9%
730,851.00	12.86	0.86	1,100,000.00	82.9	6%	7%	7%	0%	-13%
782,089.70	14.03	0.94	1,100,000.00	81.0	-1%	-1%	-1%	0%	-11%
820,077.30	14.94	1.00	1,100,000.00	79.5	-6%	-8%	-8%	0%	-9%
784,629.90	14.10	0.94	1,100,000.00	80.8	-1%	-2%	-2%	0%	-10%
794,978.30	14.34	0.96	1,100,000.00	80.4	-3%	-3%	-3%	0%	-10%
756,453.90	13.44	0.90	1,100,000.00	82.0	2%	3%	3%	0%	-12%
749,671.90	13.28	0.89	1,100,000.00	82.3	3%	4%	4%	0%	-12%
811,019.00	14.72	0.98	1,100,000.00	79.8	-5%	-6%	-6%	0%	-9%
794,410.50	14.33	0.96	1,100,000.00	80.5	-2%	-3%	-3%	0%	-10%
734,573.10	12.94	0.86	1,100,000.00	82.8	5%	7%	7%	0%	-13%
751,017.80	13.31	0.89	1,100,000.00	82.2	3%	4%	4%	0%	-12%
821,660.40	14.98	1.00	1,100,000.00	79.4	-6%	-8%	-8%	0%	-8%
748,509.40	13.26	0.88	1,100,000.00	82.3	3%	4%	4%	0%	-12%

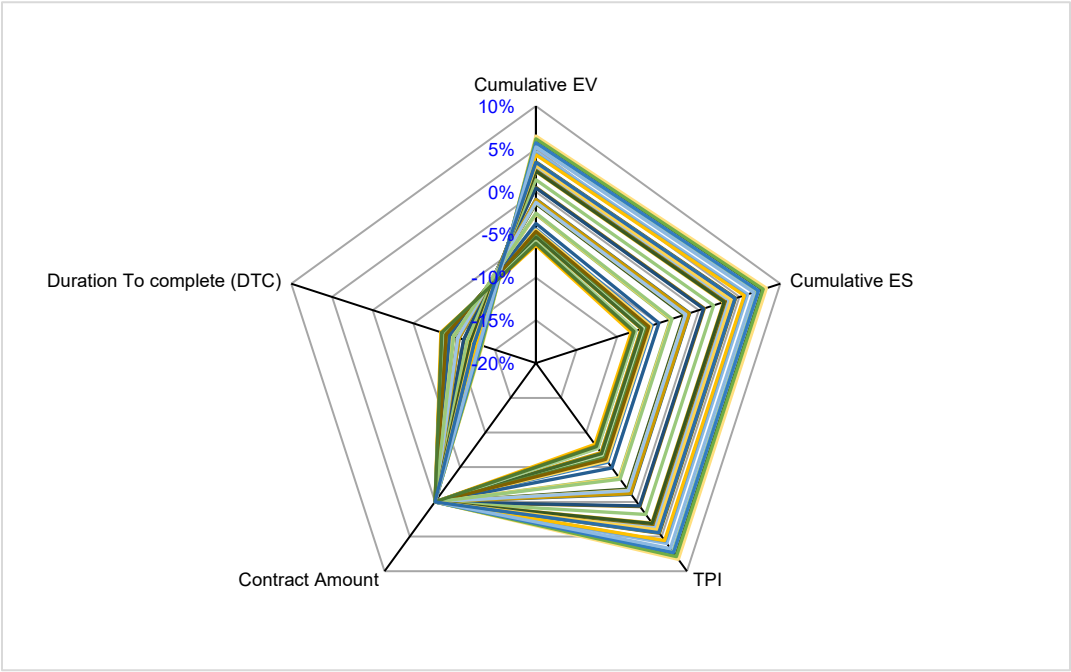


Figure 5.27 Interaction Between Predictor and Target Variables – Concrete Forecasting Model

5.5.2 Analysis of Excavation Work Package Prediction Model

Table 5.16 presents the parameters of the excavation work package, as well as, Table 5.17 depicts the portion of the records analyzed, highlighting the eighteen period which is studied. Range of values greater than $1'364,048 (previous EV) and less than $1'553,022 (following EV) are considered for the EV in the simulations. In total, were carried out 50 simulations.

Table 5.16 Work Package Parameters for Excavation

Work Package Parameters	**Value**
Planned periods	14
Actual periods	27
Current analyzed period (randomly selected)	18
Budget (dollars)	2'000,000
DTC $_{18th}$ (predicted with actual values)	49.5

Table 5.17 Excavation Work Package - Records for Sensitivity Analysis

Actual Duration (weeks)	Actual Duration (days)	Cumulative EV ($)	Cumulative Earned Schedule (weeks)	Time Performance Index (TPI)	Contract Amount ($)	Duration To complete (DTC, days)
….						
15	105	1'157,427	6.67	0.444	2'000,000	68.2
16	112	1'262,355	6.88	0.430	2'000,000	61.9
17	119	1'364,048	7.13	0.419	2'000,000	55.6
18	126	1'461,330	7.41	0.411	2'000,000	DTC $_{18th}$
19	133	1'553,022	7.67	0.404	2'000,000	---
….						

As a result, fifty replicates were generated, as presented in Table 5.18. The predictors corresponding to the "eighteen period" and, consequently, the predicted Duration to Complete (DTC) are described below. These predictions encompass variations relative to the original parameters for this period. Furthermore, Figure 5.28 illustrates the interaction between independent and target variables. It reveals that a variation ranging from -4% to 4% in the Earned

Value (EV) values corresponds to a variability between 3% and 2% in the target variable (Duration to Complete), respectively. In this scenario, the Earned Schedule (ES) ranges between -2% and 2%.

Table 5.18 Results of Fifty Simulations for Sensitivity Analysis - Excavation Forecasting Model

Independent Variables and Predicted DAC for 18th period					Variations (%)				
Cumulative EV ($)	Cumulative Earned Schedule (weeks)	Time Performance Index (TPI)	Contract Amount ($)	Duration To complete (DTC, days)	EV	ES	TPI	Contract Amount	DTC
1,508,594.20	7.55	0.42	2,000,000.00	60.6	-3%	-2%	-2%	0%	4%
1,460,855.90	7.41	0.41	2,000,000.00	61.3	0%	0%	0%	0%	3%
1,479,068.50	7.47	0.41	2,000,000.00	61.1	-1%	-1%	-1%	0%	3%
1,493,230.20	7.51	0.42	2,000,000.00	60.8	-2%	-1%	-1%	0%	3%
1,436,162.60	7.34	0.41	2,000,000.00	61.6	2%	1%	1%	0%	2%
1,464,695.50	7.42	0.41	2,000,000.00	61.3	0%	0%	0%	0%	3%
1,451,758.20	7.39	0.41	2,000,000.00	61.4	1%	0%	0%	0%	2%
1,460,862.20	7.41	0.41	2,000,000.00	61.3	0%	0%	0%	0%	3%
1,411,481.40	7.27	0.40	2,000,000.00	62.0	3%	2%	2%	0%	2%
1,461,094.50	7.41	0.41	2,000,000.00	61.3	0%	0%	0%	0%	3%
1,488,630.00	7.49	0.42	2,000,000.00	60.9	-2%	-1%	-1%	0%	3%
1,444,311.10	7.36	0.41	2,000,000.00	61.5	1%	1%	1%	0%	2%
1,508,625.20	7.55	0.42	2,000,000.00	60.6	-3%	-2%	-2%	0%	4%
1,464,621.80	7.42	0.41	2,000,000.00	61.3	0%	0%	0%	0%	3%
1,520,059.10	7.58	0.42	2,000,000.00	60.5	-4%	-2%	-2%	0%	4%
1,434,401.40	7.34	0.41	2,000,000.00	61.7	2%	1%	1%	0%	2%
1,470,394.00	7.44	0.41	2,000,000.00	61.2	-1%	0%	0%	0%	3%
1,417,978.30	7.29	0.40	2,000,000.00	61.9	3%	2%	2%	0%	2%
1,450,803.70	7.38	0.41	2,000,000.00	61.4	1%	0%	0%	0%	2%
1,419,797.30	7.29	0.41	2,000,000.00	61.9	3%	2%	2%	0%	2%
1,419,427.00	7.29	0.41	2,000,000.00	61.9	3%	2%	2%	0%	2%
1,416,766.00	7.29	0.40	2,000,000.00	61.9	3%	2%	2%	0%	2%
1,402,138.90	7.24	0.40	2,000,000.00	62.1	4%	2%	2%	0%	1%
1,490,389.00	7.50	0.42	2,000,000.00	60.9	-2%	-1%	-1%	0%	3%
1,450,459.40	7.38	0.41	2,000,000.00	61.5	1%	0%	0%	0%	2%
1,403,584.50	7.25	0.40	2,000,000.00	62.1	4%	2%	2%	0%	1%
1,485,110.80	7.48	0.42	2,000,000.00	61.0	-2%	-1%	-1%	0%	3%
1,487,331.20	7.49	0.42	2,000,000.00	60.9	-2%	-1%	-1%	0%	3%
1,483,286.90	7.48	0.42	2,000,000.00	61.0	-2%	-1%	-1%	0%	3%
1,485,096.80	7.48	0.42	2,000,000.00	61.0	-2%	-1%	-1%	0%	3%
1,427,391.60	7.32	0.41	2,000,000.00	61.8	2%	1%	1%	0%	2%
1,515,408.70	7.57	0.42	2,000,000.00	60.6	-4%	-2%	-2%	0%	4%
1,473,639.70	7.45	0.41	2,000,000.00	61.1	-1%	0%	0%	0%	3%
1,457,443.20	7.40	0.41	2,000,000.00	61.4	0%	0%	0%	0%	3%
1,513,283.90	7.56	0.42	2,000,000.00	60.6	-4%	-2%	-2%	0%	4%
1,501,774.10	7.53	0.42	2,000,000.00	60.7	-3%	-2%	-2%	0%	4%
1,468,141.20	7.43	0.41	2,000,000.00	61.2	0%	0%	0%	0%	3%
1,515,690.70	7.57	0.42	2,000,000.00	60.6	-4%	-2%	-2%	0%	4%
1,399,750.30	7.24	0.40	2,000,000.00	62.1	4%	2%	2%	0%	1%
1,512,901.30	7.56	0.42	2,000,000.00	60.6	-4%	-2%	-2%	0%	4%
1,455,018.80	7.40	0.41	2,000,000.00	61.4	0%	0%	0%	0%	3%
1,418,730.90	7.29	0.41	2,000,000.00	61.9	3%	2%	2%	0%	2%
1,425,084.40	7.31	0.41	2,000,000.00	61.8	2%	1%	1%	0%	2%
1,511,146.40	7.56	0.42	2,000,000.00	60.6	-3%	-2%	-2%	0%	4%
1,443,271.30	7.36	0.41	2,000,000.00	61.5	1%	1%	1%	0%	2%

1,476,756.00	7.46	0.41	2,000,000.00	61.1	-1%	-1%	-1%	0%	3%
1,515,480.00	7.57	0.42	2,000,000.00	60.6	-4%	-2%	-2%	0%	4%
1,399,326.50	7.23	0.40	2,000,000.00	62.1	4%	2%	2%	0%	1%
1,493,600.30	7.51	0.42	2,000,000.00	60.8	-2%	-1%	-1%	0%	3%
1,444,809.00	7.37	0.41	2,000,000.00	61.5	1%	1%	1%	0%	2%

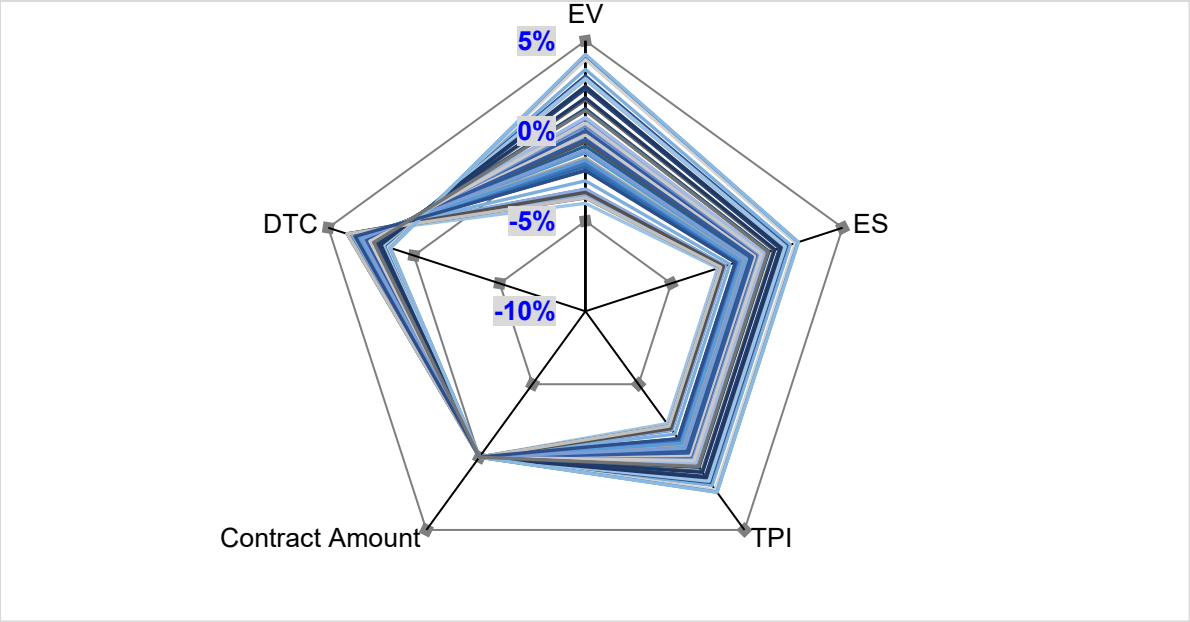


Figure 5.28 Interaction Between Predictor and Target Variables – Excavation Forecasting Model

5.5.3 Analysis of Backfill Work Package Prediction Model.

Table 5.19 presents the parameters of the backfill work package, as well as, in Table 5.20 depicts the portion of the records analyzed, highlighting the fifteen period which is studied. Range of values greater $1’676,408 (previous EV) and, less than $1’837,939 (next EV)

Table 5.19 Work Package Parameters for Backfill

Project characteristics	**Value**
Planned periods	7
Actual periods	18
Current analysed period (randomly selected)	15
Budget (dollars)	1’900,000
DTC $_{15th}$ (predicted with actual values)	3.3

Table 5.20 Backfill Work Package - Records for Sensitivity Analysis

Actual Duration (weeks)	Actual Duration (days)	Cumulative EV ($)	Cumulative Earned Schedule (weeks)	Time Performance Index (TPI)	Contract Amount ($)	Duration To complete (DTC, days)
….						
12	84	1’436,694	4.51	0.37	1’900,000	21.36
13	91	1’564,489	4.69	0.36	1’900,000	15.26
14	98	1’676,408	4.85	0.34	1’900,000	9.24
15	105	1’768,781	4.98	0.33	1’900,000	DTC $_{15th}$
16	112	1’837,939	5.55	0.34	1’900,000	---
….						

As a result, fifty predictor’s replicates were obtained, shown in Table 5.21. The independent inputs yielded for the fifteen period and, consequently, the predicted Duration to Complete (DAC) are shown below. It includes the variations with respect to the original parameters for this period. The analysis showed that a variation between -8% and 5% in the EV values represents a variability between 52% and 59% in the target variable (Duration to Complete), respectively. The ES is between -5% and 2% in this scenario. This also shows that this model is highly sensitive to variations of input variables, unlike the excavation and concrete forecasting models.

Table 5.21 Results of Fifty Simulations for Sensitivity Analysis in the Backfill Forecasting

Independent Variables and Predicted DAC for 15th period					Variations (%)				
Cumulative EV ($)	Cumulative Earned Schedule (weeks)	Time Performance Index (TPI)	Contract Amount ($)	Duration To complete (DTC, days)	EV	ES	TPI	Contract Amount	DTC
1,720,719.80	4.91	0.33	1,900,000.00	10.94	3%	1%	1%	0%	48%
1,755,905.70	4.96	0.33	1,900,000.00	11.02	1%	0%	0%	0%	48%
1,794,965.50	5.13	0.34	1,900,000.00	11.21	-1%	-3%	-3%	0%	47%
1,794,347.70	5.12	0.34	1,900,000.00	11.21	-1%	-3%	-3%	0%	47%
1,743,900.50	4.95	0.33	1,900,000.00	10.99	1%	1%	1%	0%	48%
1,726,552.40	4.92	0.33	1,900,000.00	10.95	2%	1%	1%	0%	48%
1,788,674.10	5.06	0.34	1,900,000.00	11.14	-1%	-2%	-2%	0%	47%
1,748,224.60	4.95	0.33	1,900,000.00	11.00	1%	1%	1%	0%	48%
1,742,922.60	4.95	0.33	1,900,000.00	10.99	1%	1%	1%	0%	48%
1,726,492.10	4.92	0.33	1,900,000.00	10.95	2%	1%	1%	0%	48%

1,800,778.00	5.19	0.35	1,900,000.00	11.29	-2%	-4%	-4%	0%	46%
1,779,296.80	5.00	0.33	1,900,000.00	11.07	-1%	0%	0%	0%	47%
1,721,063.50	4.91	0.33	1,900,000.00	10.94	3%	1%	1%	0%	48%
1,762,346.10	4.97	0.33	1,900,000.00	11.03	0%	0%	0%	0%	47%
1,727,037.50	4.92	0.33	1,900,000.00	10.95	2%	1%	1%	0%	48%
1,748,420.60	4.95	0.33	1,900,000.00	11.00	1%	1%	1%	0%	48%
1,779,119.80	5.00	0.33	1,900,000.00	11.07	-1%	0%	0%	0%	47%
1,773,408.70	4.99	0.33	1,900,000.00	11.05	0%	0%	0%	0%	47%
1,765,600.20	4.98	0.33	1,900,000.00	11.04	0%	0%	0%	0%	47%
1,726,513.00	4.92	0.33	1,900,000.00	10.95	2%	1%	1%	0%	48%
1,755,296.40	4.96	0.33	1,900,000.00	11.02	1%	0%	0%	0%	48%
1,751,353.10	4.96	0.33	1,900,000.00	11.01	1%	0%	0%	0%	48%
1,763,396.70	4.97	0.33	1,900,000.00	11.03	0%	0%	0%	0%	47%
1,729,824.30	4.93	0.33	1,900,000.00	10.96	2%	1%	1%	0%	48%
1,774,851.70	4.99	0.33	1,900,000.00	11.06	0%	0%	0%	0%	47%
1,769,824.10	4.98	0.33	1,900,000.00	11.05	0%	0%	0%	0%	47%
1,794,640.80	5.12	0.34	1,900,000.00	11.21	-1%	-3%	-3%	0%	47%
1,787,410.30	5.05	0.34	1,900,000.00	11.13	-1%	-1%	-1%	0%	47%
1,758,170.70	4.97	0.33	1,900,000.00	11.02	1%	0%	0%	0%	48%
1,777,235.20	4.99	0.33	1,900,000.00	11.06	0%	0%	0%	0%	47%
1,756,678.40	4.96	0.33	1,900,000.00	11.02	1%	0%	0%	0%	48%
1,752,060.50	4.96	0.33	1,900,000.00	11.01	1%	0%	0%	0%	48%
1,767,697.30	4.98	0.33	1,900,000.00	11.04	0%	0%	0%	0%	47%
1,794,263.50	5.12	0.34	1,900,000.00	11.20	-1%	-3%	-3%	0%	47%
1,754,465.30	4.96	0.33	1,900,000.00	11.01	1%	0%	0%	0%	48%
1,780,718.30	5.00	0.33	1,900,000.00	11.07	-1%	0%	0%	0%	47%
1,774,837.70	4.99	0.33	1,900,000.00	11.06	0%	0%	0%	0%	47%
1,799,092.80	5.17	0.34	1,900,000.00	11.27	-2%	-4%	-4%	0%	46%
1,770,667.00	4.98	0.33	1,900,000.00	11.05	0%	0%	0%	0%	47%
1,791,742.80	5.10	0.34	1,900,000.00	11.17	-1%	-2%	-2%	0%	47%
1,721,493.40	4.92	0.33	1,900,000.00	10.94	3%	1%	1%	0%	48%
1,741,675.80	4.94	0.33	1,900,000.00	10.99	2%	1%	1%	0%	48%
1,750,503.40	4.96	0.33	1,900,000.00	11.01	1%	1%	1%	0%	48%
1,776,568.20	4.99	0.33	1,900,000.00	11.06	0%	0%	0%	0%	47%
1,787,137.40	5.05	0.34	1,900,000.00	11.12	-1%	-1%	-1%	0%	47%
1,745,521.10	4.95	0.33	1,900,000.00	10.99	1%	1%	1%	0%	48%
1,723,349.10	4.92	0.33	1,900,000.00	10.94	3%	1%	1%	0%	48%
1,793,350.70	5.11	0.34	1,900,000.00	11.19	-1%	-3%	-3%	0%	47%
1,770,235.70	4.98	0.33	1,900,000.00	11.05	0%	0%	0%	0%	47%
1,736,756.90	4.94	0.33	1,900,000.00	10.97	2%	1%	1%	0%	48%

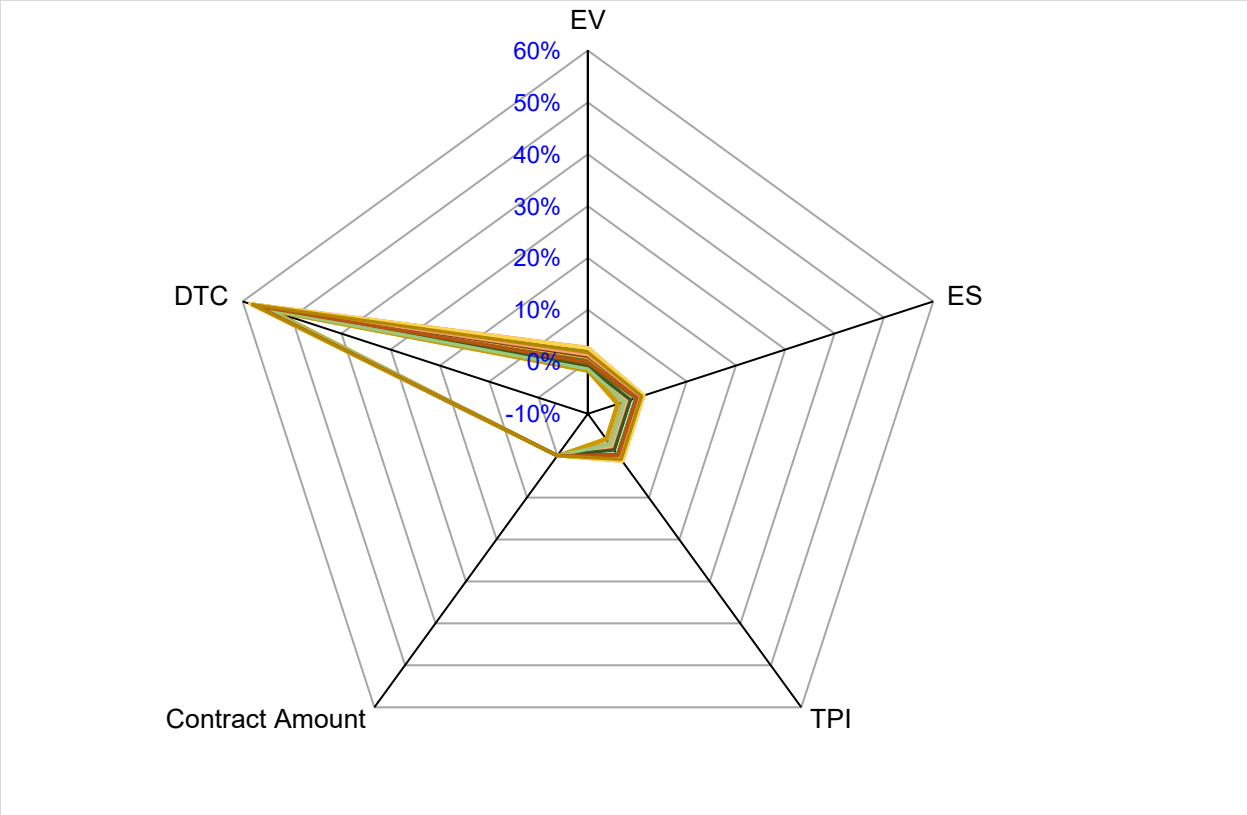


Figure 5.29 Interaction Between Predictor and Target Variables – Backfill Forecasting Model

Chapter 6 Conclusions

6.1 Research Summary

Construction projects are executed complexly and dynamically, yielding large quantities of data each time. That is why accurately forecasting outcomes, such as project duration, becomes paramount and challenging simultaneously. Typical forecasting approaches contractors adopt when addressing ongoing projects consider traditional methods such as Earned Value Management, Critical Path Method, or Montecarlo simulation. However, each presents multiple drawbacks, even when powered by sophisticated software. Some inconveniences related to EVM can be the static performance consideration applied over the remaining project, CPM might be the inaccuracy assumptions of remaining work durations, and Monte Carlo simulation could be biased when incorporating uncertainty parameters. To address these shortfalls, this research proposes whole management that starts with data collection and finishes with accurate project duration prediction by applying deep learning algorithms. It aims to leverage large amounts of data created during project executions, suggesting identifying, managing, and organizing data, which should be used to make well-based data-driven forecasts.

The research went through four stages. First, an extensive review of existing research was conducted on project duration management, delay factors affecting the project schedule management, forecasting duration methods, the latest machine learning algorithms to address time series datasets on regression problems, and machine learning applications in construction projects. This work is detailed in the second chapter of the thesis. The second stage involved an in-depth examination of the construction projects by facing the execution phase to identify potential factors associated with project duration forecasting. It assessed the adopted operational organization

(vertical analysis), process involved, and project time progress (horizontal analysis) to understand the project intricacies at this phase. Then, combined with well-known forecasting practices such as Monte Carlo simulation and Earned Schedule Management (a derived of EVM), led to the identification of four essential project duration influencing factors for the Data Acquisition modelling. DAM was developed through a relational database and an ERD. Finally, an intensive inspection of actual, available data was performed. The findings from this examination are presented in the third chapter of the thesis.

The third stage explained the development of the machine learning model and the User Interface. The former describes the assessment of three alternatives to arrange the dataset to get the optimal outcome. After that, the data preprocessing step was explained, including the feature selection conducted through multicollinearity analysis and Spearman correlation matrix, data splitting (involving the training, validation, and test datasets) and data normalization using the min-max technique. Next, the preprocessed data is fed to the forecasting model by the rolling window technique, emphasizing the importance of the autoregressive time series dataset property. It also described specific data treatment per each algorithm (LSTM, CONV-1D and MLP), like the hyperparameters tuning. The performance metrics that fit more when assessing time series datasets were explained, such as MAE, RMSE, MASE, and sMAPE. Once machine learning predictions are obtained, the way to calculate the whole project duration is explained using PDM-CPM methods. On the other hand, each component of the User Interface was described, and its application was properly sequenced since data is inputted until the duration forecast reports. The discussion on this approach is found in the fourth chapter of the thesis.

The last stage involved applying the machine learning model and the user interface as a case study to evaluate its efficacy and friendly usage by practitioners. The pipeline used actual data from a

mining civil project portfolio. The data preprocessing comprised selecting the optimal input variable by feature selection techniques, and then they were split and normalized. This step delivered 173, 132, and 85 records for the Concrete, Excavation, and Backfill work packages. After that, each work package was performed per each algorithm, namely LSTM, CONV-1D and MLP. As a result, the LSTM showed the best performance by comparing three aspects: loss curves aligned with the learning process, performance metrics and adjusted R-squared aligned with unseen dataset outcomes performance. Accordingly, the LSTM model was incorporated into the User Interface. Finally, the model was compared to the EVM and ESM forecasting methods, and a sensitivity analysis was performed.

This thesis has successfully developed a proof of concept (PoC) to validate the feasibility of using artificial intelligence for forecasting duration-at-completion in construction projects. The results of the PoC confirmed that deep learning algorithms can overcome the inaccuracy problems of traditional methods by effectively managing historical data. Additionally, a user interface was proposed to make the solution more accessible to non-expert practitioners, demonstrating its potential. The study also considered and addressed challenges such as unstructured and insufficient actual data using techniques like data augmentation. This new approach paves the way for innovative methods of project duration forecasting using artificial intelligence, particularly deep learning.

6.2 Expected Contributions

6.2.1 Academic Contributions

In the academic field, this research proposes a new approach to project duration forecasting:

- The model training has used data from multiple projects, not only one, representing better different scenarios.
- Unlike most previous studies, this study approaches the forecasting problem as a regression, Earlier works, aim to set classifiers as targets and assign outcomes to such classes.
- A novel framework to consolidate work package predictions into the overall project prediction based on CPM and PDM methodologies.
- This model handles Multivariate Time Series datasets to address the project monitoring dynamic during the execution phase. It struggles to select the proper algorithm to handle time dependencies. Moreover, the multivariate feature added complexity because it implies managing multiple predictors simultaneously and understanding the relationship between variables. Due to the complexity of construction projects, the research problem should be solved naturally by involving more than one predictor.
- This study presented a forecasting model comparing three Deep Learning algorithms: LSTM from Recurrent Neural Networks (RNN), CONV-1D from Convolutional Neural Networks (CNN) and the well-known MLP to learn complex sequential data patterns. They were chosen for their demonstrated capabilities in managing multivariate time series datasets.

This investigation aims to contribute to AI applications in the construction sector, especially in forecasting, considering its early exploration stage.

6.2.2 Industry Contributions

On the industry side, this research is expected to contribute as follows:

- Delivering accurate predictions earlier than traditional methods enables project managers to handle resources efficiently and mitigate risks effectively. This model can also prevent indirect cost overruns and additional expenses due to liquidated damages.

- Enhancing forecasting performance at work package and project levels compared to traditional methods. This granular approach allows the project manager to mitigate potential risks or leverage potential opportunities. By acting, the overall project duration will also be positively impacted in a timely manner.
- Additionally, this research proposed a data collection framework oriented toward overcoming the challenging problem of poor data management and helping to identify impacting duration factors during project monitoring. Implementing it mitigates processing delays in obtaining forecasts. Similarly, the machine learning model improves accuracy by being fed new, quality data collected over time.
- The GUI was designed to be user-friendly and to bridge the gap between understanding complex deep-learning algorithms and using them for predictions. The GUI also aims to give practitioners forecasting reports with charts and tables. Gantt charts are provided to compare planned, actual, and forecasts; behaviour curves are provided to visualize the deviation of finish dates; and summarised tables are provided to show the historical project predictions. Additionally, interaction with the project schedule to gather essential information and a visual representation of the critical path through CPM is returned on the overall project calculation.

6.3 Limitations

One of the challenges was the lack of sufficient, actual project data at the work package level with shared characteristics such as type of construction, location, remoteness, complexity, technology used, owner type, contract type, and so on. Machine learning algorithms rely heavily on consistent data to identify patterns. To address this issue, the research assigned actual data for the training and validation datasets, which perform the learning process, and augmented data (generated from

original data) for the test process, which requests unseen data. Overcoming this issue would provide a significant spectrum of duration-affecting factors that can be selected when performing the feature selection step.

Another challenge in this study was the quality of data available without missing values or inconsistencies. Data quality also implies that data is tracked sequentially during the project's execution and should be coherent across different sources. For example, the project schedule information should be aligned with the project cost at the same analysis period.

6.4 Recommendations for future research

This study addresses three algorithms chosen due to their better predominance when performing predictions with a time series data set according to the literature review; however, it can be added more to this comparison evaluation. As mentioned earlier, knowing other scenarios, such as new types of projects, new construction locations, or, more specifically, weather and transportation challenges, among others, will be necessary. Those will influence the algorithm selected. While this study verifies that LSTM performs well by managing time series datasets, further investigation is required, especially regarding the challenges in the construction sector. It would provide new insights into the deep learning model performance. Similarly, as the classification problem is prominent so far, more regression problem approaches are needed. In dynamic construction project tracking, data is collected sequentially over the project time. Therefore, more investigation considering multivariate time series datasets will improve the forecasting duration.

On the other hand, the CPM-PDM techniques were employed when integrating work package-level predictions to obtain overall project duration; nevertheless, it can be experimented with using other methods such as Critical Chain Project Management, Constraint Programming, Line of

Balance, or Pull-Driven Scheduling. Whereas drawbacks were related earlier, they can be addressed using machine learning algorithms. Once done, integrating the improved scheduling technique selected with individual predictions would offer new insights. This matter would need more investigation. Finally, migrating to any cloud service can improve the standalone GUI. It will allow real-time interaction with data sources, thus providing quick deep learning-based predictions.

References

Abdel Azeem, S. A., Hosny, H. E., & Ibrahim, A. H. (2014). Forecasting project schedule performance using probabilistic and deterministic models. *HBRC Journal*, *10*(1), 35–42. https://doi.org/10.1016/j.hbrcj.2013.09.002

Abioye, S. O., Oyedele, L. O., Akanbi, L., Ajayi, A., Davila Delgado, J. M., Bilal, M., Akinade, O. O., & Ahmed, A. (2021). Artificial intelligence in the construction industry: A review of present status, opportunities and future challenges. *Journal of Building Engineering*, *44*, 103299. https://doi.org/10.1016/j.jobe.2021.103299

Adekunle, P., Aigabvboa, C., Thwala, W., Akinradewo, O., & Oke, A. E. (2022). *Challenges confronting construction information management*. *8*. https://doi.org/10.3389/fbuil.2022.1075674

Aggarwal, C. (2015). *Data mining: The textbook* (1st edition). Springer Science+Business Media.

Ahiaga-Dagbui, D. D., & Smith, S. D. (2014). Dealing with construction cost overruns using data mining. *Construction Management and Economics*, *32*(7–8), 682–694. https://doi.org/10.1080/01446193.2014.933854

Ajayi, B. O., & Chinda, T. (2022). Impact of Construction Delay-Controlling Parameters on Project Schedule: DEMATEL-System Dynamics Modeling Approach. *Frontiers in Built Environment*, *8*, 799314. https://doi.org/10.3389/fbuil.2022.799314

Alachiotis, N. S., Kotsiantis, S., Sakkopoulos, E., & Verykios, V. S. (2022). Supervised machine learning models for student performance prediction. *Intelligent Decision Technologies*, *16*(1), 93–106. https://doi.org/10.3233/IDT-210251

Al-Saggaf, H. A. (1998). The five commandments of construction project delay analysis. *Cost Engineering*, *40*(4), 37.

American Automatic Control Council, & Internationale Förderung für Automatische Lenkung (Eds.). (2001). *Proceedings of the 2001 American Control Conference, ACC: June 25 - 27, 2001, Crystal Gateway Marriot, Arlington, VA, USA. Vol. 6* (Vol. 6). IEEE Service Center.

Ammar, M. A. (2020). Resource optimisation in line of balance scheduling. *Construction Management and Economics*, *38*(8), 715–725. https://doi.org/10.1080/01446193.2019.1606924

Andrade, P. A. D., Martens, A., & Vanhoucke, M. (2019). Using real project schedule data to compare earned schedule and earned duration management project time forecasting capabilities. *Automation in Construction*, *99*, 68–78. https://doi.org/10.1016/j.autcon.2018.11.030

Ansah, R. H., & Sorooshian, S. (2018). 4P delays in project management. *Engineering, Construction and Architectural Management*, *25*(1), 62–76.

Ansar, A., Flyvbjerg, B., Budzier, A., & Lunn, D. (2014). Should we build more large dams? The actual costs of hydropower megaproject development. *Energy Policy*, *69*, 43–56. https://doi.org/10.1016/j.enpol.2013.10.069

Armstrong, G., Gilge, C., Max, K., & Vora, S. (2023). *Familiar Challenges—New Approaches—2023 Global Construction Survey* (p. 40). KPMG International.

Arsov, M., Zdravevski, E., Lameski, P., Corizzo, R., Koteli, N., Gramatikov, S., Mitreski, K., & Trajkovik, V. (2021). Multi-horizon air pollution forecasting with deep neural networks. *Sensors*, *21*(4), 1235.

Assaf, S. A., & Al-Hejji, S. (2006). Causes of delay in large construction projects. *International Journal of Project Management*, *24*(4), 349–357. https://doi.org/10.1016/j.ijproman.2005.11.010

Auffarth, B. (2021). *Machine learning for time-series with Python: Forecast, predict, and detect anomalies with state-of-the-art machine learning methods*. Packt.

Augenstein, S., McMahan, H. B., Ramage, D., Ramaswamy, S. I., Kairouz, P., Chen, M., Mathews, R., & Arcas, B. a. Y. (2019). Generative Models for Effective ML on Private, Decentralized Datasets. *ArXiv*, *abs/1911.06679*. https://consensus.app/papers/models-effective-private-decentralized-datasets-augenstein/8d7ecf0be190505086f391609c4896fe/

Ayalew, T., Dakhli, Z., & Lafhaj, Z. (2016). Assessment on performance and challenges of Ethiopian construction industry. *Journal of Architecture and Civil Engineering*, *2*(11), 01–11.

Bagui, S., & Earp, R. (2011). *Database Design Using Entity-Relationship Diagrams, Second Edition*. https://doi.org/10.1201/9781439861776

Bai, S., Kolter, J. Z., & Koltun, V. (2018). *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. https://doi.org/10.48550/ARXIV.1803.01271

Baldwin, A., & Bordoli, D. (2014). *A handbook for construction planning and scheduling*. Wiley Blackwell.

Ballesteros-Pérez, P., Cerezo-Narváez, A., Otero-Mateo, M., Pastor-Fernández, A., Zhang, J., & Vanhoucke, M. (2020). Forecasting the Project Duration Average and Standard Deviation from Deterministic Schedule Information. *Applied Sciences*, *10*(2), 654. https://doi.org/10.3390/app10020654

Bandara, K., Hewamalage, H., Liu, Y.-H., Kang, Y., & Bergmeir, C. (2021). Improving the accuracy of global forecasting models using time series data augmentation. *Pattern Recognition*, *120*, 108148. https://doi.org/10.1016/j.patcog.2021.108148

Baptiste, P., Laborie, P., Pape, C. L., & Nuijten, W. (2006). Constraint-Based Scheduling and Planning. In *Foundations of Artificial Intelligence* (Vol. 2, pp. 761–799). Elsevier. https://doi.org/10.1016/S1574-6526(06)80026-X

Barraza, G. A., Back, W. E., & Mata, F. (2004). Probabilistic Forecasting of Project Performance Using Stochastic S Curves. *Journal of Construction Engineering and Management*, *130*(1), 25–32. https://doi.org/10.1061/(ASCE)0733-9364(2004)130:1(25)

Barrientos-Orellana, A., Ballesteros-Pérez, P., Mora-Melia, D., González-Cruz, M. C., & Vanhoucke, M. (2021). Stability and accuracy of deterministic project duration forecasting methods in earned value management. *Engineering, Construction and Architectural Management*. https://doi.org/10.1108/ECAM-12-2020-1045

Bayhan, H., Damci, A., & Demirkesen Çakır, S. (2020). *Line of Balance (LoB) Scheduling versus Takt Time Planning (TTP): A Comparison of Two Methods*.

Bhanja, S., & Das, A. (2018). Impact of data normalization on deep neural network for time series forecasting. *arXiv Preprint arXiv:1812.05519*.

Bhattacharyya, A., Chakraborty, T., & Rai, S. N. (2020). *Stochastic forecasting of COVID-19 daily new cases across countries with a novel hybrid time series model*. https://doi.org/10.1101/2020.10.01.20205021

Biskup, J., & Menzel, R. (2007). Building a Tool for Cost-Based Design of Object-Oriented Database Schemas. In C. Parent, K.-D. Schewe, V. C. Storey, & B. Thalheim (Eds.), *Conceptual*

Modeling—ER 2007 (Vol. 4801, pp. 120–131). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-75563-0_10

Bobrova, T. (2023). Updating parameters of an information model for road construction flow in work development project. *The Russian Automobile and Highway Industry Journal*. https://doi.org/10.26518/2071-7296-2022-19-6-916-927

Boskers, N. D., & AbouRizk, S. M. (2005). Modeling Scheduling Uncertainty in Capital Construction Projects. *Proceedings of the Winter Simulation Conference, 2005.*, 1500–1507. https://doi.org/10.1109/WSC.2005.1574417

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2016). *Time series analysis: Forecasting and control* (Fifth edition). Wiley.

Brownlee, J. (2018). *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery. https://books.google.ca/books?id=o5qnDwAAQBAJ

Caniato, F., Kalchschmidt, M., & Ronchi, S. (2011). Integrating quantitative and qualitative forecasting approaches: Organizational learning in an action research case. *Journal of the Operational Research Society*, *62*(3), 413–424. https://doi.org/10.1057/jors.2010.142

Carlo, M. (2017). *Markov Chain*. https://doi.org/10.1007/978-1-4899-7687-1_100285

Carney, M., Cunningham, P., & Lucey, B. (2006). Making Density Forecasting Models Statistically Consistent. *Frontiers in Finance & Economics*. https://doi.org/10.2139/ssrn.877629

Chandra, R., Goyal, S., & Gupta, R. (2021). Evaluation of Deep Learning Models for Multi-Step Ahead Time Series Prediction. *IEEE Access*, *9*, 83105–83123. https://doi.org/10.1109/ACCESS.2021.3085085

Chao, L.-C., & Chien, C.-F. (2009). Estimating Project S-Curves Using Polynomial Function and Neural Networks. *Journal of Construction Engineering and Management*, *135*, 169–177. https://doi.org/10.1061/(ASCE)0733-9364(2009)135:3(169)

Chekanov, S. (2016). *Data Analysis and Data Mining*. 431–473. https://doi.org/10.1007/978-3-319-28531-3_12

Chen, H., Chen, W., & Lin, Y.-L. (2016). Earned value project management: Improving the predictive power of planned value. *International Journal of Project Management*, *34*, 22–29. https://doi.org/10.1016/J.IJPROMAN.2015.09.008

Cheng, M.-Y., Chang, Y.-H., & Korir, D. (2019). Novel Approach to Estimating Schedule to Completion in Construction Projects Using Sequence and Nonsequence Learning. *Journal of Construction Engineering and Management*, *145*(11), 04019072. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001697

Cheng, M.-Y., & Wu, Y.-W. (2009). Evolutionary support vector machine inference system for construction management. *Automation in Construction*, *18*(5), 597–604. https://doi.org/10.1016/j.autcon.2008.12.002

Chhotelal, B. Y., Tophique, Q., & Hira, L. Y. (2023). A study of delays in indian road construction projects under different contracts. *I-Manager's Journal on Structural Engineering*, *12*(1), 34. https://doi.org/10.26634/jste.12.1.19974

Cho, J.-E., & Lim, J. (2020). Research on Improving Schedule Forecasting Method for Delayed Defense Research & Development Project. *Journal of the Korea Institute of Military Science and Technology*. https://doi.org/10.9766/kimst.2020.23.3.286

Chollet, F. (2018). *Deep learning with Python*. Manning Publications Co.

Chou, J.-S., Chen, H.-M., Hou, C.-C., & Lin, C.-W. (2010). Visualized EVM system for assessing project performance. *Automation in Construction*, *19*(5), 596–607. https://doi.org/10.1016/j.autcon.2010.02.006

CII. (2019). *PDRI: Project Definition Rating Index Industrial Projects* (5). Construction Industry Institute.

Cristóbal, J. R. S. (2017). The S-curve envelope as a tool for monitoring and control of projects. *Procedia Computer Science*, *121*, 756–761. https://doi.org/10.1016/j.procs.2017.11.097

Cropper, A., Tamaddoni-Nezhad, A., & Muggleton, S. H. (2016). Meta-Interpretive Learning of Data Transformation Programs. In K. Inoue, H. Ohwada, & A. Yamamoto (Eds.), *Inductive Logic Programming* (Vol. 9575, pp. 46–59). Springer International Publishing. https://doi.org/10.1007/978-3-319-40566-7_4

Croux, C., & Dehon, C. (2010). Influence functions of the Spearman and Kendall correlation measures. *Statistical Methods & Applications*, *19*, 497–515. https://doi.org/10.2139/ssrn.1585216

Daoud, J. I. (2017). Multicollinearity and Regression Analysis. *Journal of Physics: Conference Series*, *949*(1), 012009. https://doi.org/10.1088/1742-6596/949/1/012009

Dibert, J. C., & Velez, J. C. (2006). *An Analysis of Earned Value Management Implementation Within the F-22 System Program Office's Software Development:* Defense Technical Information Center. https://doi.org/10.21236/ADA460316

Dilipkumar, S., & Durairaj, M. (2022). Detection of Attacks Using Multilayer Perceptron Algorithm. In G. Ranganathan, X. Fernando, & F. Shi (Eds.), *Inventive Communication and Computational Technologies* (Vol. 311, pp. 943–950). Springer Nature Singapore. https://doi.org/10.1007/978-981-16-5529-6_71

Dinov, I. D. (2023). Deep Learning, Neural Networks. In I. D. Dinov, *Data Science and Predictive Analytics* (pp. 773–901). Springer International Publishing. https://doi.org/10.1007/978-3-031-17483-4_14

Duan, Y., Edwards, J. S., & Dwivedi, Y. K. (2019). Artificial intelligence for decision making in the era of Big Data – evolution, challenges and research agenda. *International Journal of Information Management*, *48*, 63–71. https://doi.org/10.1016/j.ijinfomgt.2019.01.021

Durbach, I., Merven, B., & McCall, B. (2017). Expert elicitation of autocorrelated time series with application to e3 (energy-environment-economic) forecasting models. *Environmental Modelling & Software*, *88*, 93–105. https://doi.org/10.1016/j.envsoft.2016.11.007

Durdyev, S., & Hosseini, M. R. (2019). Causes of delays on construction projects: A comprehensive list. *International Journal of Managing Projects in Business*, *13*(1), 20–46. https://doi.org/10.1108/IJMPB-09-2018-0178

Enshassi, A., Al-Najjar, J., & Kumaraswamy, M. (2009). Delays and cost overruns in the construction projects in the Gaza Strip. *Journal of Financial Management of Property and Construction*, *14*(2), 126–151. https://doi.org/10.1108/13664380910977592

Faghihi, V., Reinschmidt, K. F., & Kang, J. H. (2014). Construction scheduling using Genetic Algorithm based on Building Information Model. *Expert Systems with Applications*, *41*(16), 7565–7578. https://doi.org/10.1016/j.eswa.2014.05.047

Fayad, A. M., Hussein, B. A., Hajj-Hassan, M., & Haj-Ali, A. (2019). A New Approach for Complementing the Earned Value Method for Project Progress Monitoring and Controlling. *New Challenges in Accounting and Finance*. https://doi.org/10.32038/ncaf.2019.02.02

Fearnley, N., Robinson, G., & Leonard, J. (2023). *Global Construction Futures* (p. 600). Oxford Economics. https://www.oxfordeconomics.com/

Field, A. (2024). *Discovering Statistics Using IBM SPSS Statistics* (6th ed. edition). Sage Publications Ltd.

Flyvbjerg, B., Bruzelius, N., & Rothengatter, W. (2003). *Megaprojects and risk: An anatomy of ambition*. Cambridge University Press.

Flyvbjerg, B., Garbuio, M., & Lovallo, D. (2009). Delusion and Deception in Large Infrastructure Projects: Two Models for Explaining and Preventing Executive Disaster. *California Management Review*, *51*(2), 170–194. https://doi.org/10.2307/41166485

Frandson, A. G., Seppänen, O., & Tommelein, I. D. (2015). *Comparison between location based management and takt time planning*. 3–12.

Fukushima, K. (2000). Defining 'Pull Scheduling.' *Bulletin of the Department of Economics*, *1*, 35–36.

Gan, G., Ma, C., & Wu, J. (2007). *Data clustering: Theory, algorithms, and applications*. SIAM, Society for Industrial and Applied Mathematics ; American Statistical Association.

Ghanem, M., Hamzeh, F., Seppänen, O., Shehab, L., & Zankoul, E. (2022). Pull planning versus push planning: Investigating impacts on crew performance from a location-based perspective. *Frontiers in Built Environment*, *8*, 980023. https://doi.org/10.3389/fbuil.2022.980023

Gneiting, T., & Katzfuss, M. (2014). Probabilistic Forecasting. *Annual Review of Statistics and Its Application*, *1*(1), 125–151. https://doi.org/10.1146/annurev-statistics-062713-085831

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. The MIT Press.

Grandage, A. J. (2022). Managing Performance for Capital Projects. *State and Local Government Review*, *54*(3), 221–235. https://doi.org/10.1177/0160323X221079675

Gridin, I. (2022). *Time series forecasting using deep learning: Combining PyTorch, RNN, TCN, and deep neural network models to provide production-ready prediction solutions* (First edition). BPB Publications.

Gutman, A. J., & Goldmeier, J. (2021). *Becoming a data head: How to think, speak, and understand data science, statistics, and machine learning*. Wiley.

Hajdu, M. (1997). *Network Scheduling Techniques for Construction Project Management* (Vol. 16). Springer US. https://doi.org/10.1007/978-1-4757-5951-8

Haque, A., Hossain, T., Murshed, M. N., Iqbal, K. I. B., & Uddin, M. M. (2022). Estimating Aerodynamic Data via Supervised Learning. *2022 25th International Conference on Computer and Information Technology (ICCIT)*, 67–71. https://doi.org/10.1109/ICCIT57492.2022.10054896

Harrington, P. (2012). *Machine learning in action*. Manning Publications Co.

Hegazy, T., & Mostafa, K. (2021). *Enhanced CPM/LOB repetitive scheduling formulation to meet deadlines*. 337–346.

Hoberman, S. (2015). *Data Modeling Made Simple with ER/Studio Data Architect: Adapting to Agile Data Modeling in a Big Data World* (2nd edition). Technics Publications.

Hopkins, D., Rickwood, D. J., Hallford, D. J., & Watsford, C. (2022). Structured data vs. unstructured data in machine learning prediction models for suicidal behaviors: A systematic review and meta-analysis. *Frontiers in Digital Health*, *4*, 945006. https://doi.org/10.3389/fdgth.2022.945006

Hyndman, R. J. (2006). Another look at forecast-accuracy metrics for intermittent demand. *Foresight: The International Journal of Applied Forecasting*, *4*(4), 43–46.

Inoue, A., Jin, L., & Rossi, B. (2017). Rolling window selection for out-of-sample forecasting with time-varying parameters. *Journal of Econometrics*, *196*(1), 55–67.

Iranmanesh, H., Mojir, N., & Kimiagari, S. (2007). A new formula to “Estimate At Completion” of a Project’s time to improve “Earned Value Management System.” *2007 IEEE International Conference on Industrial Engineering and Engineering Management*, 1014–1017. https://doi.org/10.1109/IEEM.2007.4419345

Javeri, I. Y., Toutiaee, M., Arpinar, I. B., Miller, J. A., & Miller, T. W. (2021). Improving Neural Networks for Time-Series Forecasting using Data Augmentation and AutoML. *2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService)*, 1–8. https://doi.org/10.1109/BigDataService52369.2021.00006

Kandel, I., & Castelli, M. (2020). The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, *6*(4), 312–315. https://doi.org/10.1016/j.icte.2020.04.010

Karimi-Bidhendi, S., Munshi, F., & Munshi, A. (2018). Scalable Classification of Univariate and Multivariate Time Series. *2018 IEEE International Conference on Big Data (Big Data)*, 1598–1605. https://doi.org/10.1109/BigData.2018.8621889

Kelleher, J. D. (2019). *Deep learning*. The MIT Press.

Kermanshachi, S., & Pamidimukkala, A. (2023). Uncertainty Analysis of Key Schedule Performance Indicators in Design, Procurement, and Construction Phases of Heavy Industrial Projects. *Journal of Legal Affairs and Dispute Resolution in Engineering and Construction*, *15*(1), 04522042. https://doi.org/10.1061/(ASCE)LA.1943-4170.0000588

Kerzner, H. (2017). *Project management: A systems approach to planning, scheduling, and controlling* (Twelfth edition). Wiley.

Kesavaraj, G., & Sukumaran, S. (2013). A study on classification techniques in data mining. *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, 1–7. https://doi.org/10.1109/ICCCNT.2013.6726842

Khafri, A. (2018, Fall). *A Framework for Forecasting Project Estimate at Completion Using Historical and Current Performance Data*. ERA. https://doi.org/10.7939/R32F7K69P

Khamooshi, H., & Abdi, A. (2017). Project Duration Forecasting Using Earned Duration Management with Exponential Smoothing Techniques. *Journal of Management in Engineering*, *33*. https://doi.org/10.1061/(ASCE)ME.1943-5479.0000475

Khamooshi, H., & Golafshani, H. (2014). EDM: Earned Duration Management, a new approach to schedule performance management and measurement. *International Journal of Project Management*, *32*, 1019–1041. https://doi.org/10.1016/J.IJPROMAN.2013.11.002

Kim, B. C. (2007). *Forecasting Project Progress and Early Warning of Project Overruns with Probabilistic Methods* [Doctoral dissertation]. Texas A&M University.

Kim, J. H. (2019). Multicollinearity and misleading statistical results. *Korean Journal of Anesthesiology*, *72*(6), 558–569. https://doi.org/10.4097/kja.19087

Kostelyk, J. D. (2012, Spring). *Project Controls for Engineering Work in Practice*. ERA. https://doi.org/10.7939/R3XM73

Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. (2006). Machine learning: A review of classification and combining techniques. *Artificial Intelligence Review*, *26*(3), 159–190. https://doi.org/10.1007/s10462-007-9052-3

Kraiem, Z. M., & Diekmann, J. E. (1987). Concurrent delays in construction projects. *Journal of Construction Engineering and Management*, *113*(4), 591–602.

Kroese, D. P., Taimre, T., & Botev, Z. I. (2011). *Handbook for Monte Carlo methods*. Wiley.

Lara-Benítez, P., Carranza-García, M., & Riquelme, J. C. (2021). An experimental review on deep learning architectures for time series forecasting. *International Journal of Neural Systems*, *31*(03), 2130001.

Lawal, H. S., Ahmadu, H. A., Abdullahi, M., Yamusa, M. A., & Abdulrazaq, M. (2023). Modeling duration of building renovation projects. *Journal of Financial Management of Property and Construction*, *28*(3), 423–438. https://doi.org/10.1108/JFMPC-06-2022-0030

Lazzeri, F. (2021). *Machine learning for time series forecasting with Python*. Wiley.

Leach, L. P. (1999). Critical Chain Project Management Improves Project Performance. *Project Management Journal*, *30*(2), 39–51. https://doi.org/10.1177/875697289903000207

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Leu, S.-S., & Liu, C.-M. (2016). USING PRINCIPAL COMPONENT ANALYSIS WITH A BACK-PROPAGATION NEURAL NETWORK TO PREDICT INDUSTRIAL BUILDING CONSTRUCTION DURATION. *Journal of Marine Science and Technology*, *24*(2). https://doi.org/10.6119/JMST-015-0325-2

Li, G., & Jung, J. J. (2023). Deep learning for anomaly detection in multivariate time series: Approaches, applications, and challenges. *Information Fusion*, *91*, 93–102. https://doi.org/10.1016/j.inffus.2022.10.008

Li, Q., & Chen, Y. (2009). *Entity-Relationship Diagram*. 125–139. https://doi.org/10.1007/978-3-540-89556-5_6

Li, S. Y. (2020). Automating data augmentation: Practice, theory and new direction. *SAIL Blog. Retrieved March*, *8*, 2023.

Ling, S. (2023). *Exploring Timescale in Language Comprehension with EEG*. https://doi.org/10.7939/R3-547N-9P79

Linoff, G., Berry, M. J. A., & Linoff, G. S. (2011). *Data mining techniques: For marketing, sales, and customer relationship management* (3. ed). Wiley.

Lishner, I., & Shtub, A. (2022). Using an Artificial Neural Network for Improving the Prediction of Project Duration. *Mathematics*, *10*(22), 4189. https://doi.org/10.3390/math10224189

Litsiou, K., Polychronakis, Y., Karami, A., & Nikolopoulos, K. (2022). Relative performance of judgmental methods for forecasting the success of megaprojects. *International Journal of Forecasting*, *38*(3), 1185–1196. https://doi.org/10.1016/j.ijforecast.2019.05.018

Lorterapong, P., & Ussavadilokrit, M. (2013). Construction Scheduling Using the Constraint Satisfaction Problem Method. *Journal of Construction Engineering and Management*, *139*(4), 414–422. https://doi.org/10.1061/(ASCE)CO.1943-7862.0000582

Loshin, D. (2011). Evaluating the business impacts of poor data quality. *Information Quality Journal*.

Lovallo, D., & Kahneman, D. (2003). Delusions of Success: How Optimism Undermines Executives' Decisions. *Harvard Business Review*, *81*(7), 1–10.

Lu, M. (2020). *Construction Planning by Engineer-In-Training* (1st ed.). Kendall Hunt.

Makridakis, S. (1993). Accuracy measures: Theoretical and practical concerns. *International Journal of Forecasting*, *9*(4), 527–529. https://doi.org/10.1016/0169-2070(93)90079-3

Makridakis, S., & Gaba, A. (1998). *Judgment: Its role and value for strategy*. INSEAD.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLOS ONE*, *13*(3), e0194889. https://doi.org/10.1371/journal.pone.0194889

Makridakis, S., Spiliotis, E., Assimakopoulos, V., Semenoglou, A.-A., Mulder, G., & Nikolopoulos, K. (2023). Statistical, machine learning and deep learning forecasting methods: Comparisons and ways forward. *Journal of the Operational Research Society*, *74*(3), 840–859. https://doi.org/10.1080/01605682.2022.2118629

Mao, Z., Xu, Y., & Suarez, E. (2023). *Dataset Management Platform for Machine Learning*. https://doi.org/10.48550/ARXIV.2303.08301

Matey, T. K., Bhonde, C. B. K., Pathak, S., & Kholia, B. (2017). A Case Study: Line of Balance (LOB) Method for High Rise Residential Project. *International Journal of Advance Research and Innovative Ideas in Education*, *3*, 1729–1736.

Matti, T., & Antti, L. (2020). Improving the Information Flow in the Construction Phase of a Construction Project. *Proceedings of the Creative Construction E-Conference 2020*. https://doi.org/10.3311/ccc2020-044

Mayo-Alvarez, L., Alvarez-Risco, A., Del-Aguila-Arcentales, S., Sekar, M. C., & Yañez, J. A. (2022). A Systematic Review of Earned Value Management Methods for Monitoring and Control of Project Schedule Performance: An AHP Approach. *Sustainability*, *14*(22), 15259. https://doi.org/10.3390/su142215259

Meng, F., Yu, S., & Xue, J. (2022). Construction Schedule Management System for Large-Scale Construction Projects Based on Multisensor Network. *Computational Intelligence and Neuroscience*, *2022*. https://doi.org/10.1155/2022/3003552

Merrow, E. W. (2011). *Industrial megaprojects: Concepts, strategies, and practices for success*. Wiley.

Miller, J. D. (2017). *Statistics for data science: Leverage the power of statistics for data analysis, classification, regression, machine learning, and neural networks*. Packt.

Mishra, S., & Misra, A. (2017). Structured and Unstructured Big Data Analytics. *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*, 740–746. https://doi.org/10.1109/CTCEEC.2017.8454999

Mitchell, T. M. (2013). *Machine learning* (Nachdr.). McGraw-Hill.

Montgomery, D. C., Jennings, C. L., & Kulahci, M. (2015). *Introduction to time series analysis and forecasting* (Second edition). Wiley.

Mubarak, S. (2019). *Construction project scheduling and control* (Fourth edition). Wiley.

Müller, D., Müller, M. G., Kress, D., & Pesch, E. (2022). An algorithm selection approach for the flexible job shop scheduling problem: Choosing constraint programming solvers through machine learning. *European Journal of Operational Research*, *302*(3), 874–891. https://doi.org/10.1016/j.ejor.2022.01.034

Nayak, S., Misra, B. B., & Behera, H. S. (2014). Impact of data normalization on stock index forecasting. *International Journal of Computer Information Systems and Industrial Management Applications*, *6*, 13–13.

Nelles, O. (2001). Unsupervised Learning Techniques. In O. Nelles, *Nonlinear System Identification* (pp. 137–155). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-04323-3_6

Nielsen, A. (2019). *Practical time series analysis: Prediction with statistics and machine learning* (First edition). O'Reilly.

Nowrin, T. (2022). *Forecasting Short-Term Road Surface Temperatures – A Neural Network-based Approach*. https://doi.org/10.7939/R3-3PK3-KP19

Oberlender, G. D., & Spencer, G. R. (2022). *Project management for engineering and construction: A life-cycle approach* (Fourth edition). McGraw Hill Education.

Obite, C., Olewuezi, N., Ugwuanyim, G., & Bartholomew, D. (2020). Multicollinearity Effect in Regression Analysis: A Feed Forward Artificial Neural Network Approach. *Asian Journal of Probability and Statistics*, 22–33. https://doi.org/10.9734/ajpas/2020/v6i130151

Pabuccu, H., & Barbu, A. (2023). Feature Selection for Forecasting. *arXiv Preprint arXiv:2303.02223*.

Pan, Y., & Zhang, L. (2021). Roles of artificial intelligence in construction engineering and management: A critical review and future trends. *Automation in Construction*, *122*, 103517. https://doi.org/10.1016/j.autcon.2020.103517

Papadopoulos, C., & Yeung, H. (2001). Uncertainty estimation and Monte Carlo simulation method. *Flow Measurement and Instrumentation*, *12*, 291–298. https://doi.org/10.1016/S0955-5986(01)00015-2

Park, J. E. (2021). Schedule delays of major projects: What should we do about it? *Transport Reviews*, *41*(6), 814–832. https://doi.org/10.1080/01441647.2021.1915897

Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2020). Deep Adaptive Input Normalization for Time Series Forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(9), 3760–3765. https://doi.org/10.1109/TNNLS.2019.2944933

Pavlova, O., Yushak, A., & Antipova, M. (2021). MANAGEMENT OF INFORMATION FLOWS IN THE ACTIVITIES OF CONSTRUCTION COMPANIES IN THE REGION. *Market Infrastructure*. https://doi.org/10.32843/infrastruct61-19

Petropoulos, F., Apiletti, D., Assimakopoulos, V., Babai, M. Z., Barrow, D. K., Ben Taieb, S., Bergmeir, C., Bessa, R. J., Bijak, J., Boylan, J. E., Browell, J., Carnevale, C., Castle, J. L., Cirillo, P., Clements, M. P., Cordeiro, C., Cyrino Oliveira, F. L., De Baets, S., Dokumentov, A., … Ziel, F. (2022). Forecasting: Theory and practice. *International Journal of Forecasting*, *38*(3), 705–871. https://doi.org/10.1016/j.ijforecast.2021.11.001

Pirbazari, A. M., Chakravorty, A., & Rong, C. (2019). Evaluating Feature Selection Methods for Short-Term Load Forecasting. *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 1–8. https://doi.org/10.1109/BIGCOMP.2019.8679188

Priyo, M. (2021). Earned Value Management System in Indonesian Construction Projects. *International Journal of Integrated Engineering*. https://doi.org/10.30880/ijie.2021.13.03.005

Program Evaluation Research Task (PERT): Summary Report (p. 35). (1958). Bureau of Naval Weapons, Department of the Navy.

Project Management Institute (Ed.). (2017). *A guide to the project management body of knowledge / Project Management Institute* (Sixth edition). Project Management Institute.

Project Management Institute (Ed.). (2019). *Practice standard for scheduling* (Third edition). Project Management Institute, Inc.

Project Management Institute (Ed.). (2023). *Process groups: A practice guide*. Project Management Institute, Inc.

Ramchoun, H., Amine, M., Idrissi, J., Ghanou, Y., & Ettaouil, M. (2016). Multilayer Perceptron: Architecture Optimization and Training. *International Journal of Interactive Multimedia and Artificial Intelligence*, *4*(1), 26. https://doi.org/10.9781/ijimai.2016.415

Raz, T., Barnes, R., & Dvir, D. (2003). A Critical Look at Critical Chain Project Management. *Project Management Journal*, *34*(4), 24–32. https://doi.org/10.1177/875697280303400404

Sallabi, A. K. (2011). *Monte Carlo Simulations of Adsorbed Molecules on Ionic Surfaces*. https://doi.org/10.5772/16047

Sanders, N. R., & Manrodt, K. B. (1994). Forecasting practices in US corporations: Survey results. *Interfaces*, *24*(2), 92–100.

Sanni-Anibire, M. O., Mohamad Zin, R., & Olatunji, S. O. (2022). Causes of delay in the global construction industry: A meta analytical review. *International Journal of Construction Management*, *22*(8), 1395–1407. https://doi.org/10.1080/15623599.2020.1716132

Sanni-Anibire, M. O., Zin, R. M., & Olatunji, S. O. (2022). Machine learning model for delay risk assessment in tall building projects. *International Journal of Construction Management*, *22*(11), 2134–2143. https://doi.org/10.1080/15623599.2020.1768326

Schober, P., Boer, C., & Schwarte, L. (2018). Correlation Coefficients: Appropriate Use and Interpretation. *Anesthesia & Analgesia*, *126*. https://doi.org/10.1213/ANE.0000000000002864

Selcuk, O., Turkoglu, H., Polat, G., & Hajdu, M. (2024). An integrative literature review on the causes of delays in construction projects: Evidence from developing countries. *International Journal of Construction Management*, *24*(6), 610–622. https://doi.org/10.1080/15623599.2022.2135939

Senaviratna, N. A. M. R., & A. Cooray, T. M. J. (2019). Diagnosing Multicollinearity of Logistic Regression Model. *Asian Journal of Probability and Statistics*, *5*(2), Article 2. https://doi.org/10.9734/ajpas/2019/v5i230132

Sepasgozar, S. M. E., Karimi, R., Shirowzhan, S., Mojtahedi, M., Ebrahimzadeh, S., & McCarthy, D. (2019). Delay Causes and Emerging Digital Tools: A Novel Model of Delay Analysis, Including Integrated Project Delivery and PMBOK. *Buildings*, *9*(9), 191. https://doi.org/10.3390/buildings9090191

Shahhossein, V., Afshar, M. R., & Amiri, O. (2017). The Root Causes of Construction Project Failure. *Scientia Iranica*, *0*(0), 0–0. https://doi.org/10.24200/sci.2017.4178

Shahsavand, P., Marefat, A., & Parchamijalal, M. (2018). Causes of delays in construction industry and comparative delay analysis techniques with SCL protocol. *Engineering, Construction and Architectural Management*, *25*(4), 497–533. https://doi.org/10.1108/ECAM-10-2016-0220

Shrestha, N. (2020). Detecting Multicollinearity in Regression Analysis. *American Journal of Applied Mathematics and Statistics*, *8*, 39–42. https://doi.org/10.12691/ajams-8-2-1

Singh, J., & Banerjee, R. (2019). A Study on Single and Multi-layer Perceptron Neural Network. *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, 35–40. https://doi.org/10.1109/ICCMC.2019.8819775

Siu, M.-F. F., Lu, M., & AbouRizk, S. (2014). Bi-level project simulation methodology to integrate superintendent and project manager in decision making: Shutdown/turnaround applications.

Proceedings of the Winter Simulation Conference 2014, 3353–3364. https://doi.org/10.1109/WSC.2014.7020169

Statistics Canada. (2024). *Gross domestic product (GDP) at basic prices, by industry, monthly* [dataset]. [object Object]. https://doi.org/10.25318/3610043401-ENG

Staudemeyer, R. C., & Morris, E. R. (2019). Understanding LSTM--a tutorial into long short-term memory recurrent neural networks. *arXiv Preprint arXiv:1909.09586*.

Stephenson, H. L. (Ed.). (2015). *Total cost management framework: An integrated approach to portfolio, program, and project management* (Second edition). AACE International.

Tang, Y., Sun, Q., Liu, R., & Wang, F. (2018). Resource Leveling Based on Line of Balance and Constraint Programming. *Computer-Aided Civil and Infrastructure Engineering*, *33*. https://doi.org/10.1111/mice.12383

Tanga, O., Akinradewo, O., Aigbavboa, C., Oke, A., & Adekunle, S. (2022). Data Management Risks: A Bane of Construction Project Performance. *Sustainability*, *14*(19), 12793. https://doi.org/10.3390/su141912793

Taud, H., & Mas, J. F. (2018). Multilayer Perceptron (MLP). In M. T. Camacho Olmedo, M. Paegelow, J.-F. Mas, & F. Escobar (Eds.), *Geomatic Approaches for Modeling Land Change Scenarios* (pp. 451–455). Springer International Publishing. https://doi.org/10.1007/978-3-319-60801-3_27

The origins of schedule management: The concepts used in planning, allocating, visualizing and managing time in a project. (2018). *Frontiers of Engineering Management*, *0*(0), 0. https://doi.org/10.15302/J-FEM-2018012

The standard for organizational project management (OPM). (2018). Project Management Institute.

Tommelein, I. D. (1998). Pull-Driven Scheduling for Pipe-Spool Installation: Simulation of Lean Construction Technique. *Journal of Construction Engineering and Management*, *124*(4), 279–288. https://doi.org/10.1061/(ASCE)0733-9364(1998)124:4(279)

Torres, J. F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F., & Lora, A. T. (2020). Deep Learning for Time Series Forecasting: A Survey. *Big Data*. https://doi.org/10.1089/big.2020.0159

Turban, E., Sharda, R., & Delen, D. (2011). *Decision support and business intelligence systems* (9th ed). Prentice Hall.

Ullrich, T. (2021). On the Autoregressive Time Series Model Using Real and Complex Analysis. *Forecasting*. https://doi.org/10.3390/forecast3040044

Umadevi, S., & Marseline, K. S. J. (2017). A survey on data mining classification algorithms. *2017 International Conference on Signal Processing and Communication (ICSPC)*, 264–268. https://doi.org/10.1109/CSPC.2017.8305851

Vabalas, A., Gowen, E., Poliakoff, E., & Casson, A. J. (2019). Machine learning algorithm validation with a limited sample size. *PloS One*, *14*(11), e0224365.

Vandevoorde, S., & Vanhoucke, M. (2006). A comparison of different project duration forecasting methods using earned value metrics. *International Journal of Project Management*, *24*(4), 289–302. https://doi.org/10.1016/j.ijproman.2005.10.004

Vanhoucke, M. (2012). Earned Value Management. In M. Vanhoucke, *Project Management with Dynamic Scheduling* (pp. 215–238). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-25175-7_12

Vermeulen, A. F. (2020). Unsupervised Learning: Using Unlabeled Data. In A. F. Vermeulen, *Industrial Machine Learning* (pp. 181–206). Apress. https://doi.org/10.1007/978-1-4842-5316-8_6

Vertenten, M., Pretorius, L., & Pretorius, J. (2009). Earned Value as a performance measurement tool for small and large construction projects in a South African environment. *AFRICON 2009*, 1–5. https://doi.org/10.1109/AFRCON.2009.5308080

Wang, L. (2016). Discovering phase transitions with unsupervised learning. *Physical Review B*, *94*(19), 195105. https://doi.org/10.1103/PhysRevB.94.195105

Wen, Q., Sun, L., Song, X., Gao, J., Wang, X., & Xu, H. (2020). *Time Series Data Augmentation for Deep Learning: A Survey*. 4653–4660. https://doi.org/10.24963/ijcai.2021/631

Wheelan, C. J. (2014). *Naked statistics: Stripping the dread from the data* (First published as a Norton paperback). W.W. Norton & Company.

Wilson, J. M. (2003). Gantt charts: A centenary appreciation. *European Journal of Operational Research*, *149*(2), 430–437. https://doi.org/10.1016/S0377-2217(02)00769-5

Witten, I. H., & Witten, I. H. (Eds.). (2017). *Data mining: Practical machine learning tools and techniques* (Fourth Edition). Elsevier.

Wu, I.-C., Borrmann, A., Beißert, U., König, M., & Rank, E. (2010). Bridge construction schedule generation with pattern-based construction methods and constraint-based simulation. *Advanced Engineering Informatics*, *24*(4), 379–388. https://doi.org/10.1016/j.aei.2010.07.002

Wu, W., Fang, L., Ma, T., Yang, Y., Zhao, W., Yu, P., & Wang, C. (2022). Research on Project Duration Prediction Based on Artificial Neural Network. *2022 IEEE 5th International Electrical and Energy Conference (CIEEC)*, 1613–1618. https://doi.org/10.1109/CIEEC54735.2022.9845997

Xu, Y., & Goodacre, R. (2018). On splitting training and validation set: A comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. *Journal of Analysis and Testing*, *2*(3), 249–262.

Yang, I.-T., & Ioannou, P. G. (2001). *Resource-driven scheduling for repetitive projects: A pull-system approach*. *1*, 365–377.

Yang, J.-B., & Kao, C.-K. (2012). Critical path effect based delay analysis method for construction projects. *International Journal of Project Management*, *30*(3), 385–397. https://doi.org/10.1016/j.ijproman.2011.06.003

Yang, L., Ma, Y., & Zhang, Y. (2023). Measuring Consistency in Text-based Financial Forecasting Models. *ArXiv*, *abs/2305.08524*. https://doi.org/10.48550/arXiv.2305.08524

Yates, J. K., & Lockley, E. E. (2002). Documenting and Analyzing Construction Failures. *Journal of Construction Engineering and Management*, *128*(1), 8–17. https://doi.org/10.1061/(ASCE)0733-9364(2002)128:1(8)

Yildiz, B., Bilbao, J. I., & Sproul, A. B. (2017). A review and analysis of regression and machine learning models on commercial building electricity load forecasting. *Renewable and Sustainable Energy Reviews*, *73*, 1104–1122. https://doi.org/10.1016/j.rser.2017.02.023

Yu, F., Chen, X., Cory, C. A., Yang, Z., & Hu, Y. (2021). An Active Construction Dynamic Schedule Management Model: Using the Fuzzy Earned Value Management and BP Neural Network. *KSCE Journal of Civil Engineering*, *25*(7), 2335–2349. https://doi.org/10.1007/s12205-021-1041-6

Yudhi, M. R. A. (2022, September 21). Prediction Project Task Completion Using Supervised Machine Learning Method: A Conceptual Approach. *Proceedings of Indonesian Petroleum Association, 46th Annual Convention & Exhibition, 2022*. Indonesian Petroleum Association - 46th Annual Convention & Exhibition 2022. https://doi.org/10.29118/IPA22-F-104

Yue, X., Pye, S., DeCarolis, J., Li, F. G. N., Rogan, F., & Gallachóir, B. Ó. (2018). A review of approaches to uncertainty assessment in energy system optimization models. *Energy Strategy Reviews*, *21*, 204–217. https://doi.org/10.1016/j.esr.2018.06.003

Yue, Y., Shi, X., Qin, L., Zhang, X., Chen, Y., Xu, J., Zheng, Z., Cao, Y., Liu, D., Li, Z., & Li, Y. (2023). *Ultrafast-and-Ultralight ConvNet-Based Intelligent Monitoring System for Diagnosing Early-Stage Mpox Anytime and Anywhere*. https://doi.org/10.48550/ARXIV.2308.13492

Zahran, K., Nour, M., & Hosny, O. (2016). The effect of learning on line of balance scheduling: Obstacles and potentials. *International Journal of Engineering Science*, *6*(4), 3831–3841.

Zhang, X., Roy Chowdhury, R., Shang, J., Gupta, R., & Hong, D. (2023). Towards Diverse and Coherent Augmentation for Time-Series Forecasting. *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. https://doi.org/10.1109/ICASSP49357.2023.10097273

Zhang, Y., Wang, J., & Wang, X. (2014). Review on probabilistic forecasting of wind power generation. *Renewable and Sustainable Energy Reviews*, *32*, 255–270. https://doi.org/10.1016/j.rser.2014.01.033

Zivot, E., Wang, J., Zivot, E., & Wang, J. (2003). Rolling analysis of time series. *Modeling Financial Time Series with S-Plus®*, 299–346.

Zupančič, B., Karba, R., Blažič, S., & Univerza v Ljubljani (Eds.). (2007). *Proceedings of the 6th EUROSIM Congress on Modelling and Simulation, EUROSIM 2007: 9 - 13 September, 2007, Ljubljana, Slovenia*.

Appendix A: Python Script for Duration at Completion Forecasting using Long Short-Term Memory (LSTM) Algorithm – Work Package Concrete

1. Importing libraries and functions

```
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import numpy as np
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_percentage_error
from math import sqrt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import r2_score
from keras.models import load_model
from keras.regularizers import l2
from matplotlib.font_manager import FontProperties
import random
```

2. Reading raw data and initial Data Inspection

```
df_1 = pd.read_csv('Concrete_Data_NEW_3x.csv')
rev_1 = df_1.values.astype('float32')
titles_1 = list(df_1.columns)[1:]
feature_keys_1 = list(df_1.columns)[1:]
get_colors_1 = lambda n: ["#%06x" % random.randint(0, 0xFFFFFF) for _ in
range(n)]
colors_1 = get_colors_1(len(feature_keys_1))
date_time_key_1 = df_1.columns[0]

def show_raw_visualization(df, date_time_key, feature_keys, colors):
    data_0 = df[date_time_key]
    fig, axes = plt.subplots(nrows=round(len(feature_keys)/2+.1), ncols=2,
figsize=(15, 20), dpi=80, facecolor="w", edgecolor="k")
    for i in range(len(feature_keys)):
        key = feature_keys[i]
        c = colors[i % (len(colors))]
        t_data = df[key]
        t_data.index = data_0
        t_data.head()
        ax = t_data.plot(ax=axes[i // 2, i % 2], color=c, rot=0,)
        ax.legend([list(df.columns)[1:][i]])
        ax.set_title("{}".format(list(df.columns)[1:][i]),loc='left',color='blue'
)
        ax.grid()
```

```
        labels = ax.get_xticklabels() + ax.get_yticklabels()
        [label.set_fontname('Arial') for label in labels]
        plt.tight_layout()
show_raw_visualization(df_1, date_time_key_1, feature_keys_1, colors_1)

def show_heatmap(data):
    plt.matshow(data.corr())
    plt.xticks(range(data.shape[1]), data.columns, fontsize=8, rotation=90)
    plt.gca().xaxis.tick_bottom()
    plt.yticks(range(data.shape[1]), data.columns, fontsize=8)
    cb = plt.colorbar()
    cb.ax.tick_params(labelsize=12)
    plt.title("Feature Correlation Heatmap", fontsize=14)
    plt.show()
show_heatmap(df_1[list(df_1.columns)[1:]])

Selected_list= [1,9,10,11,17,18]
print("The selected parameters are:", ",".join([titles_1[i] for i in
Selected_list]))
selected_features_1 = [feature_keys_1[i] for i in Selected_list]
```

3. Data Preprocessing: Splitting, Normalization and Grouping for model training

```
split_1=0.7
past_1 = 3
future_1 = 0

### Data Splitting ###
def splitting(split, df, selected_features):
    train_split = int(split * int(df.shape[0]))
    df_train= df[selected_features][:train_split]
    df_test= df[selected_features][train_split:]
    return df_train, df_test
df_train1, df_test1 = splitting(split_1, df_1, selected_features_1)

### Data Normalization ###
def normalization(df):
    values = df.values.astype('float32')
    scaler = MinMaxScaler()
    values_scaled = scaler.fit_transform(values)
    df_scaled = pd.DataFrame(values_scaled)
    return df_scaled
df_train_scaled1 = normalization(df_train1)
df_test_scaled1 = normalization(df_test1)

##### Variables for denormalization ###
def denormalization(df):
    Max_orig = df.iloc[:,len(Selected_list)-1].max()
    Min_orig = df.iloc[:,len(Selected_list)-1].min()
    Delta = Max_orig-Min_orig
    return Delta, Min_orig
Delta_train1, Min_orig_train1 = denormalization(df_train1)
```

```
Delta_test1, Min_orig_test1 = denormalization(df_test1)

##### Grouping for Modelling ###
def df_to_X_y(df, past):
  df_as_np = df.to_numpy()
  X = []
  y = []
  for i in range(len(df_as_np)-past):
    row = [r for r in df_as_np[i:i+past]]
    X.append(row)
    label = df_as_np[i+past][df.shape[1]-1]
    y.append(label)
  return np.array(X), np.array(y)

X1_train, y1_train = df_to_X_y(df_train_scaled1, past_1)
X1_val, y1_val = df_to_X_y(df_test_scaled1, past_1)
X1_train.shape, y1_train.shape, X1_val.shape, y1_val.shape
```

4. Forecasting Model Training

```
learning_rate_1 = 0.001
epochs_1 = 200
batch_size_1=8
recurrent_dropout_1=0.30
l2_reg_factor_1=0.01

### Layer Designs ###
inputs_1 = keras.layers.Input(shape=(X1_train.shape[1], X1_train.shape[2]))
lstm_out_1 = keras.layers.LSTM(64, recurrent_dropout=recurrent_dropout_1,
kernel_regularizer=l2(l2_reg_factor_1),recurrent_regularizer=l2(l2_reg_factor_1))
(inputs_1)
outputs_1 = keras.layers.Dense(1,kernel_regularizer=l2(l2_reg_factor_1))
(lstm_out_1)

### Model development ###
model1 = keras.Model(inputs=inputs_1, outputs=outputs_1)
model1.summary()
model1.compile(optimizer=keras.optimizers.Adam(learning_rate=learning_rate_1),
loss='mae', metrics=["mse", "mape"])
es_callback_1 = keras.callbacks.EarlyStopping(monitor="val_loss", min_delta=0,
patience=5)
modelckpt_callback_1 = keras.callbacks.ModelCheckpoint(filepath='model1/',
monitor="val_loss", mode='min', verbose=1, save_best_only=True,)
history_1 = model1.fit(X1_train, y1_train, validation_data=(X1_val, y1_val),
batch_size=batch_size_1, epochs=epochs_1,
callbacks=[modelckpt_callback_1,es_callback_1], shuffle=False)
model1.save('model1.h5')

def visualize_loss(history, title):
    loss = history.history["loss"]
    val_loss = history.history["val_loss"]
    epochs = range(len(loss))
```

```
    plt.figure()
    plt.plot(epochs, loss, "b", label="Training loss")
    plt.plot(epochs, val_loss, "r", label="Validation loss")
    plt.title(title, fontsize=18, fontweight='bold', family='Arial')
    plt.xlabel("Epochs",fontsize=14, family='Arial')
    plt.ylabel("Loss",fontsize=14, family='Arial')
    font_prop = FontProperties(family='Arial', size=16)
    plt.legend(prop=font_prop)
    plt.grid(True)
    plt.show()

visualize_loss(history_1, "Training and Validation Loss\nModel: LSTM   Work
Package: Concrete")
```

5. Creating functions for Performance Metrics and Measurements

```
def performance_metrics(model, X, y, Delta, Min_orig, Seasonality_value, ytrain):
    Actuals_den = (y)*(Delta) + Min_orig
    Predicted_den = (model.predict(X).flatten())*(Delta) + Min_orig
    """MAPE"""
    MAPE= mean_absolute_percentage_error(Actuals_den, Predicted_den)*100
    """MAE"""
    MAE= mean_absolute_error(Actuals_den, Predicted_den)
    """RMSE"""
    RMSE = sqrt(mean_squared_error(Actuals_den, Predicted_den))
    """MSE"""
    MSE = mean_squared_error(Actuals_den, Predicted_den)
    """sMAPE"""
    sMAPE = (np.sum((abs(Predicted_den- Actuals_den))/(abs(Actuals_den)+
    abs(Predicted_den))))*200/(y.shape[0])
    """MASE"""
    ytrain_un = (ytrain)*(Delta) + Min_orig
    MASE = MAE / (np.sum(abs(ytrain_un[0:-Seasonality_value]-
    ytrain_un[Seasonality_value:]))/ (len(ytrain_un[0:-Seasonality_value])))

    Metrics = {'Value' : [round(MASE,3), f"{round(sMAPE,2)} %", f"{round(MAPE,3)}
    %", round(MAE,3), round(MSE,3), round(RMSE,3)]}
    df_metrics = pd.DataFrame(Metrics, index=['MASE', 'sMAPE', 'MAPE', 'MAE',
    'MSE', 'RMSE'])
    return print(df_metrics)

performance_metrics(model1, X1_train, y1_train, Delta_train1, Min_orig_train1 ,
20, y1_train)
performance_metrics(model1, X1_val, y1_val, Delta_test1, Min_orig_test1, 20,
y1_train)
```

6. Exporting and Storing Actual and Predicted Results

```
def results_to_csv(model, X, y, title, Delta, Min_orig):
    Actuals_den = (y)*(Delta) + Min_orig
    Predicted_den = (model.predict(X).flatten())*(Delta) + Min_orig
```

```
    df_results = pd.DataFrame({'Actuals':Actuals_den, 'Predicted':Predicted_den})
    df_results.to_csv(f"{title}.csv", index=False)

results_to_csv(model1, X1_train, y1_train,"Training_Concrete", Delta_train1,
Min_orig_train1)
results_to_csv(model1, X1_val, y1_val,"Validation_Concrete", Delta_test1,
Min_orig_test1)
```

7. Plotting actual and predicted results in a linear chart for comparison

```
def plot_predictions(model, Xtrain, ytrain, Xval, yval, past, Deltatrain,
Min_origtrain, Deltaval, Min_origval):
    predictions_train = (model.predict(Xtrain).flatten())*(Deltatrain) +
Min_origtrain
    dftrain = pd.DataFrame(data={'Predictions_train':predictions_train,
'Actuals_train':(ytrain)*(Deltatrain) + Min_origtrain})
    axisTrain= np.arange(past,past+ytrain.shape[0])
    plt.plot(axisTrain, dftrain['Predictions_train'],'r',ls='--', label='Training
Dataset Predictions')
    plt.plot(axisTrain, dftrain['Actuals_train'],'b', label='Training Dataset
Actuals')
    plt.title('Actual vs Predicted values - LSTM', fontname='Arial', fontsize=18,
fontweight='bold', color='blue')
    predictions_val = (model.predict(Xval).flatten())*(Deltaval) + Min_origval
    dfval = pd.DataFrame(data={'Predictions_val':predictions_val,
'Actuals_val':(yval)*(Deltaval) + Min_origval})
    axisVal= np.arange(past+ytrain.shape[0],past+ytrain.shape[0]+yval.shape[0])
    plt.plot(axisVal, dfval['Predictions_val'],'m',ls='--', label='Validation
Dataset Predictions')
    plt.plot(axisVal, dfval['Actuals_val'],'c', label='Validation Dataset
Actuals')
    plt.xlabel('Periods', fontname='Arial', fontsize=14)
    plt.ylabel('DTC (days)', fontname='Arial', fontsize=14)
    font_prop = FontProperties(family='Arial', size=12)
    plt.legend(prop=font_prop)
    plt.grid(True)
    return plt.show(), print(dftrain), print(dfval)

plot_predictions(model1, X1_train, y1_train, X1_val, y1_val, past_1,
Delta_train1, Min_orig_train1, Delta_test1, Min_orig_test1)
```

8. Testing of Forecasting Model using unseen data

```
model1 = load_model('model1.h5')
df_predic1= pd.read_csv('Synthetic_Concrete_data_SHORT_NEW_3.csv')
df_predic1 = df_predic1[selected_features_1]
df_predict_scaled1 = normalization(df_predic1)
X1_test, y1_test = df_to_X_y(df_predict_scaled1, past_1)
Xpredict1=model1.predict(X1_test).flatten()
Delta_test1, Min_orig_test1 = denormalization(df_predic1)
print(Xpredict1*Delta_test1+Min_orig_test1)
print(y1_test*Delta_test1+Min_orig_test1)
```

```
array1_1=y1_test*Delta_test1+Min_orig_test1
array2_1=Xpredict1*Delta_test1+Min_orig_test1
dfarraytest_1 = pd.DataFrame({'Column1': array1_1, 'Column2': array2_1})
dfarraytest_1.to_csv('test_concrete_results.csv', index=False)

performance_metrics(model1, X1_test, y1_test, Delta_test1, Min_orig_test1 , 20,
y1_train)
plot_scatter(model1, X1_test, y1_test,'Test Dataset\nModel: LSTM   Work Package:
Concrete')

def plot_test_predictions(model, Xtest, ytest, past, Deltatest, Min_origtest):
    predictions_test= (model.predict(Xtest).flatten())*(Deltatest) +
Min_origtest
    dftest = pd.DataFrame(data={'Predictions_test':predictions_test,
'Actuals_test':(ytest)*(Deltatest) + Min_origtest})
    axisTest= np.arange(past,past+ytest.shape[0])
    plt.plot(axisTest, dftest['Predictions_test'],'r',ls='--', label='Test
Dataset Predictions')
    plt.plot(axisTest, dftest['Actuals_test'],'b', label='Test Dataset Actuals')
    plt.title('Actual vs Predicted values - LSTM', fontname='Arial', fontsize=18,
fontweight='bold', color='blue')
    plt.xlabel('Periods', fontname='Arial', fontsize=14)
    plt.ylabel('DTC (days)', fontname='Arial', fontsize=14)
    font_prop = FontProperties(family='Arial', size=12)
    plt.legend(prop=font_prop)
    plt.grid(True)
    return plt.show(), print(dftest)

plot_test_predictions(model1, X1_test, y1_test, past_1, Delta_test1,
Min_orig_test1)
```

Appendix B: Python Script for the Graphical User Interface (GUI) for Project Duration at Completion Forecasting

1. Importing libraries and functions

```
import tkinter as tk
from tkinter import ttk
import tkinter.messagebox as messagebox
import json
import pandas as pd
from keras.models import load_model
from sklearn.metrics import mean_absolute_error
from sklearn.preprocessing import MinMaxScaler
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from tkinter import filedialog
from PIL import Image, ImageTk
import xml.etree.ElementTree as Xet
from datetime import timedelta
import networkx as nx
from datetime import datetime
import datetime
import os
import matplotlib.dates as mdates
from tkinter import font
from pandas import Timedelta
import math
from tkcalendar import Calendar
import re
from matplotlib.figure import Figure
from matplotlib.ticker import FixedLocator
from matplotlib.offsetbox import OffsetImage, AnnotationBbox
from PIL import Image
```

2. Initializing Tkinter as Tk, creating frames and storage for project information

```
root = tk.Tk()
root.title("Duration Project Forecasting for Time Series Data using Deep Learning")
root.configure(background='white')
style = ttk.Style()
style.theme_use('vista')

#Creation of frames:
project_hub_frame = ttk.Frame(root, style='Custom.TFrame')
progress_frame = ttk.Frame(root, style='Custom.TFrame')
forecasting_work_package_frame= ttk.Frame(root, style='Custom.TFrame')
forecasting_project_frame= ttk.Frame(root, style='Custom.TFrame')

# Load project info from a file if it exists
try:
    with open("project_info.json", "r") as file:
        project_info = json.load(file)
except FileNotFoundError:
    project_info = {}
def save_project_info_to_file():
    with open("project_info.json", "w") as file:
```

```
        json.dump(project_info, file, indent=4)
```

3. Defining the Project Hub to Entered project visualization

```
def project_hub():
    global treeview_projects, work_packages_list_tab2
    treeframe = ttk.Labelframe(project_hub_frame, text="Project Hub",
style='Custom2.TLabelframe', )
    treeframe.grid(row=0, column=0, padx=0, pady=0, sticky="news")
    project_hub_label = ttk.Label(treeframe, text='By selecting a listed project, detailed
information will be shown at the bottom. Also, press the "Add New Project" or "Update Project"
buttons when needed.', foreground='black', font=('Arial', 10, ), background='white')
    project_hub_label.grid(row=0, column=0, padx=10, pady=10, columnspan=3, sticky= 'w')
    treeview_projects = ttk.Treeview(treeframe, columns=("Project ID", "Project Name",
"Original Duration", "Original Budget",), show="headings", height=5, style='Custom1.Treeview')
    treeview_projects.grid(row=1, column=0, padx=10, pady=10)
    treeScroll = ttk.Scrollbar(treeframe)
    treeScroll.grid(row=1, column=1, sticky="ns")
    treeview_projects.heading("Project ID", text="Project ID")
    treeview_projects.heading("Project Name", text="Project Name")
    treeview_projects.heading("Original Duration", text="Original Duration (days)")
    treeview_projects.heading("Original Budget", text="Original Budget ($)")
    treeview_projects.column("Project ID", width=160)
    treeview_projects.column("Project Name", width=300)
    treeview_projects.column("Original Duration", width=200)
    treeview_projects.column("Original Budget", width=200)
    treeScroll.config(command=treeview_projects.yview)
    treeview_projects.config(yscrollcommand=treeScroll.set)
    treeview_projects.tag_configure('oddrow', background='#f9f9d6', font=("Arial", 11))
    treeview_projects.tag_configure('evenrow', background='lightgrey', font=("Arial", 11))
    for index, (project, attributes) in enumerate(project_info.items()):
        tag = 'oddrow' if index % 2 == 0 else 'evenrow'
        treeview_projects.insert("", "end", values=(
        project,
        attributes.get("Project Name"),
        str(attributes.get("IB-Duration (days)", "0")) + " days",
        "{:,.2f}".format(float(attributes.get("IB-Budget At Completion ($)", "0")))
        ), tags=(tag,))

    add_project_button = ttk.Button(treeframe, text="Add New Project...",
command= add_new_project, style='Custom1.TButton')
    update_button = ttk.Button(treeframe, text="Update Project...",
command= project_updating, style='Custom1.TButton',)
    add_project_button.configure(padding=(25,20))
    update_button.configure(padding=(32,20))
    add_project_button.grid(row=1, column=2, padx=10, pady=(5,0), sticky= 'nw')
    update_button.grid(row=1, column=2, padx=10, pady=(0,5), sticky='sw')

    def get_selected_project():
        global project_row
        project_row = treeview_projects.selection()
        projectid = treeview_projects.item(project_row, "text")
        return projectid
    project = get_selected_project()

    #Creation of Notebook:
    notebook = ttk.Notebook(project_hub_frame, style='Custom.TNotebook')
    notebook.grid(row=1, column=0, padx=0, pady=(10,0), sticky="news")
    tab1 = ttk.Frame(notebook)
    tab2 = ttk.Frame(notebook)
    notebook.add(tab1, text="At Project level", )
```

```
    notebook.add(tab2, text="At Work Packages level",)

    #Entries on general information tab:
    gen_info = ttk.Labelframe(tab1, text="General Information", style='Custom3.TLabelframe',
padding=(0,20))
    gen_info.grid(row=0, column=0, padx=10, pady=5, sticky='nwes')
    nameid_label_tab = ttk.Label(gen_info, text="Project ID:", style='Custom.TLabel')
    nameid_label_tab.grid(row=0, column=0,  sticky="w")
    nameid_entry_tab = ttk.Entry(gen_info, font=('Arial', 11))
    nameid_entry_tab.grid(row=0, column=1, sticky="w")
    bu_label_tab = ttk.Label(gen_info, text="Business Unit Code:" , style='Custom.TLabel')
    bu_label_tab.grid(row=0, column=2, sticky="w")
    bu_entry_tab = ttk.Entry(gen_info, font=('Arial', 11))
    bu_entry_tab.grid(row=0, column=3, sticky="w")
    portfolio_label_tab = ttk.Label(gen_info, text="Portfolio Code:" , style='Custom.TLabel')
    portfolio_label_tab.grid(row=0, column=4, sticky="w")
    portfolio_entry_tab = ttk.Entry(gen_info, font=('Arial', 11))
    portfolio_entry_tab.grid(row=0, column=5,  sticky="w")
    program_label_tab = ttk.Label(gen_info, text="Program Code:" , style='Custom.TLabel')
    program_label_tab.grid(row=0, column=6,  sticky="w")
    program_entry_tab = ttk.Entry(gen_info, font=('Arial', 11))
    program_entry_tab.grid(row=0, column=7, sticky="w")
    contract_label_tab = ttk.Label(gen_info, text="Contract Type:" , style='Custom.TLabel')
    contract_label_tab.grid(row=1, column=0,  sticky="w")
    contract_entry_tab = ttk.Entry(gen_info, font=('Arial', 11))
    contract_entry_tab.grid(row=1, column=1, sticky="w")
    owner_label_tab = ttk.Label(gen_info, text="Owner Internal Code:" , style='Custom.TLabel')
    owner_label_tab.grid(row=1, column=2,  sticky="w")
    owner_entry_tab = ttk.Entry(gen_info, font=('Arial', 11))
    owner_entry_tab.grid(row=1, column=3,sticky="w")
    construction_label_tab = ttk.Label(gen_info, text="Project Type:" , style='Custom.TLabel')
    construction_label_tab.grid(row=1, column=4,  sticky="w")
    construction_entry_tab = ttk.Entry(gen_info, font=('Arial', 11))
    construction_entry_tab.grid(row=1, column=5, sticky="w")
    location_label_tab = ttk.Label(gen_info, text="Location:" , style='Custom.TLabel')
    location_label_tab.grid(row=1, column=6,  sticky="w")
    location_entry_tab = ttk.Entry(gen_info, font=('Arial', 11))
    location_entry_tab.grid(row=1, column=7,  sticky="w")

    baseline_frame_tab = ttk.Labelframe(tab1, text="Baseline Information",
style='Custom3.TLabelframe', padding=(0,20))
    baseline_frame_tab.grid(row=1, column=0, padx=10, pady=5, sticky='nwes')

    initial_baseline_label_tab= ttk.Label(baseline_frame_tab, text="Project Initial Baseline",
font=('Arial',11,))
    initial_baseline_label_tab.grid(row=1, column=0, sticky="nwes")
    current_baseline_label_tab= ttk.Label(baseline_frame_tab, text="Project Current Baseline",
font=('Arial',11,))
    current_baseline_label_tab.grid(row=2, column=0,  sticky="nwes")

    bac_label_tab = ttk.Label(baseline_frame_tab, text="Budget At Completion ($)",
style='Custom.TLabel')
    bac_label_tab.grid(row=0, column=1,  sticky="nwes")
    baci_entry_tab = ttk.Entry(baseline_frame_tab, font=('Arial', 11))
    baci_entry_tab.grid(row=1, column=1,  sticky="nwes")
    bacc_entry_tab = ttk.Entry(baseline_frame_tab, font=('Arial', 11))
    bacc_entry_tab.grid(row=2, column=1,  sticky="nwes")
    duration_label_tab = ttk.Label(baseline_frame_tab, text="Duration (days)",
style='Custom.TLabel')
    duration_label_tab.grid(row=0, column=2,  sticky="nwes")
    durationi_entry_tab = ttk.Entry(baseline_frame_tab, font=('Arial', 11))
```

```
    durationi_entry_tab.grid(row=1, column=2,  sticky="nwes")
    durationc_entry_tab = ttk.Entry(baseline_frame_tab, font=('Arial', 11))
    durationc_entry_tab.grid(row=2, column=2,  sticky="nwes")
    startdate_label_tab = ttk.Label(baseline_frame_tab, text="Start Date (yyyy/mm/dd)",
style='Custom.TLabel')
    startdate_label_tab.grid(row=0, column=3,  sticky="nwes")
    startdatei_entry_tab = ttk.Entry(baseline_frame_tab, font=('Arial', 11))
    startdatei_entry_tab.grid(row=1, column=3,  sticky="nwes")
    startdatec_entry_tab = ttk.Entry(baseline_frame_tab, font=('Arial', 11))
    startdatec_entry_tab.grid(row=2, column=3, sticky="nwes")
    finishdate_label_tab = ttk.Label(baseline_frame_tab, text="Finish Date (yyyy/mm/dd)",
style='Custom.TLabel')
    finishdate_label_tab.grid(row=0, column=4,  sticky="nwes")
    finishdatei_entry_tab = ttk.Entry(baseline_frame_tab, font=('Arial', 11))
    finishdatei_entry_tab.grid(row=1, column=4,  sticky="nwes")
    finishdatec_entry_tab = ttk.Entry(baseline_frame_tab, font=('Arial', 11))
    finishdatec_entry_tab.grid(row=2, column=4,  sticky="nwes")

    dynamic_frame = ttk.Frame(tab2)
    dynamic_frame.grid(row=0, column=0, padx=0, pady=(10,0), sticky="news")

    def populate_entry_widgets(selected_item):
        item_values = treeview_projects.item(selected_item)['values']
        id = str(item_values[0])
        nameid_entry_tab.delete(0, tk.END)
        nameid_entry_tab.insert(0, id)
        bu_entry_tab.delete(0, tk.END)
        bu_entry_tab.insert(0, project_info[id]["Business Unit"])
        portfolio_entry_tab.delete(0, tk.END)
        portfolio_entry_tab.insert(0, project_info[id]["Portfolio Code"])
        program_entry_tab.delete(0, tk.END)
        program_entry_tab.insert(0, project_info[id]["Program Code"])
        contract_entry_tab.delete(0, tk.END)
        contract_entry_tab.insert(0, project_info[id]["Contract Type"])
        owner_entry_tab.delete(0, tk.END)
        owner_entry_tab.insert(0, project_info[id]["Owner Internal Code"])
        construction_entry_tab.delete(0, tk.END)
        construction_entry_tab.insert(0, project_info[id]["Project Type"])
        location_entry_tab.delete(0, tk.END)
        location_entry_tab.insert(0, project_info[id]["Location"])
        baci_entry_tab.delete(0, tk.END)
        baci_entry_tab.insert(0, project_info[id]["IB-Budget At Completion ($)"])
        durationi_entry_tab.delete(0, tk.END)
        durationi_entry_tab.insert(0, project_info[id]["IB-Duration (days)"])
        startdatei_entry_tab.delete(0, tk.END)
        startdatei_entry_tab.insert(0, project_info[id]["IB-Start Date"])
        finishdatei_entry_tab.delete(0, tk.END)
        finishdatei_entry_tab.insert(0, project_info[id]["IB-Finish Date"])

        if {key: value for key, value in project_info[id].items() if key.startswith("CB-")}:
            bacc_entry_tab.delete(0, tk.END)
            bacc_entry_tab.insert(0, project_info[id]["CB-Budget At Completion ($)"])
            durationc_entry_tab.delete(0, tk.END)
            durationc_entry_tab.insert(0, project_info[id]["CB-Duration (days)"])
            startdatec_entry_tab.delete(0, tk.END)
            startdatec_entry_tab.insert(0, project_info[id]["CB-Start Date"])
            finishdatec_entry_tab.delete(0, tk.END)
            finishdatec_entry_tab.insert(0, project_info[id]["CB-Finish Date"])
        else:
            bacc_entry_tab.delete(0, tk.END)
            bacc_entry_tab.insert(0, "No entry yet")
```

```
            durationc_entry_tab.delete(0, tk.END)
            durationc_entry_tab.insert(0, "No entry yet")
            startdatec_entry_tab.delete(0, tk.END)
            startdatec_entry_tab.insert(0, "No entry yet")
            finishdatec_entry_tab.delete(0, tk.END)
            finishdatec_entry_tab.insert(0, "No entry yet")

        # Clear the dynamic frame
        for widget in dynamic_frame.winfo_children():
            widget.destroy()


        work_packages = project_info[id]["Work Packages"]
        attribute_keys = ["Budget At Completion ($)", "Duration (days)", "Start Date", "Finish
Date"]

       # Row Labels for "Initial Baseline" and "Current Baseline"
            ttk.Label(dynamic_frame, text="Initial Baseline", font=('Arial',
11)).grid(row=row+1, column=0, padx=20, sticky='w')
            ttk.Label(dynamic_frame, text="Current Baseline", font=('Arial',
11)).grid(row=row+2, column=0, padx=20, sticky='w')
       # Populate entry widgets for each attribute under each baseline
            for col, key_suffix in enumerate(attribute_keys, start=1):
                # Initial Baseline Entries
                ib_key = f"IB-{key_suffix}"
                ib_entry = ttk.Entry(dynamic_frame, font=('Arial', 11))
                ib_entry.grid(row=row+1, column=col, sticky='ew')
                ib_entry.insert(0, wp_details.get(ib_key, ""))
       # Current Baseline Entries
                cb_key = f"CB-{key_suffix}"
                cb_entry = ttk.Entry(dynamic_frame, font=('Arial', 11))
                cb_entry.grid(row=row+2, column=col, sticky='ew')
                cb_entry.insert(0, wp_details.get(cb_key, "N/A"))
       # Increment row for the next work package
            row += 3
    treeview_projects.bind("<<TreeviewSelect>>",  lambda event:
populate_entry_widgets(event.widget.selection()[0]))
```

4. Entering new projects information to the GUI

```
def add_new_project():
    global original_project_entry_list, project_entries_list, new_work_package_entries
    add_window = tk.Toplevel(root, background='white')
    add_window.lift()
    add_window.title("Add New Project")
    project_info_label_frame = ttk.Labelframe(add_window, text="At Project Level",
style='Custom5.TLabelframe')
    project_info_label_frame.grid(row=0, column=0, padx=10, pady=10, sticky="nwes")
    add_window.grid_columnconfigure(0, weight=1)

    ### PROJECT GENERAL INFORMATION
    project_info_frame = ttk.Labelframe(project_info_label_frame, text="General Information",
style='Custom4.TLabelframe', padding= (10,20))
    project_info_frame.grid(row=0, column=0, columnspan=4, padx=10, pady=10, sticky="nwes")
    name_label = ttk.Label(project_info_frame, text="Project Name:", font='Arial 11',
background='white')
    name_label.grid(row=0, column=0,  pady= 2, sticky="w")
    name_text = ttk.Entry(project_info_frame, font=('Arial', 11), width= 75)
    name_text.grid(row=0, column=1, columnspan=7, sticky="w")
    id_label = ttk.Label(project_info_frame, text="Project ID:", font='Arial 11',
background='white')
```

```
    id_label.grid(row=1, column=0,  pady= 2, sticky="w")
    id_text = ttk.Entry(project_info_frame,font=('Arial', 11))
    id_text.grid(row=1, column=1,  sticky="w")
    bu_label = ttk.Label(project_info_frame, text="Business Unit Code:", font='Arial 11',
background='white')
    bu_label.grid(row=1, column=2, pady= 2, sticky="w")
    bu_text = ttk.Entry(project_info_frame,font=('Arial', 11))
    bu_text.grid(row=1, column=3,  pady= 2, sticky="w")
    pf_label = ttk.Label(project_info_frame, text="Portfolio Code:", font='Arial 11',
background='white')
    pf_label.grid(row=1, column=4,  pady= 2, sticky="w")
    pf_text = ttk.Entry(project_info_frame, font=('Arial', 11))
    pf_text.grid(row=1, column=5, sticky="w")
    pg_label = ttk.Label(project_info_frame, text="Program Code:", font='Arial 11',
background='white')
    pg_label.grid(row=1, column=6,  pady= 2, sticky="w")
    pg_text = ttk.Entry(project_info_frame, font=('Arial', 11))
    pg_text.grid(row=1, column=7,  sticky="w")
    contract_type_label = ttk.Label(project_info_frame, text="Contract Type:",font='Arial 11',
background='white')
    contract_type_label.grid(row=2, column=0,  pady= 2, sticky="w")
    contract_type_text = ttk.Combobox(project_info_frame, values=['Unit Price', 'Lump Sum',
'Time & Materials'], font=('Arial', 11) )
    contract_type_text.set('Select Contract Type')
    contract_type_text.grid(row=2, column=1, sticky="w")
    Owner_label = ttk.Label(project_info_frame, text="Owner Internal Code:", font='Arial 11',
background='white')
    Owner_label.grid(row=2, column=2,  pady= 2, sticky="w")
    Owner_text = ttk.Entry(project_info_frame, font=('Arial', 11))
    Owner_text.grid(row=2, column=3,  sticky="w")
    project_type_label = ttk.Label(project_info_frame, text="Project Type:", font='Arial 11',
background='white')
    project_type_label.grid(row=2, column=4,  pady= 2, sticky="w")
    project_type_text = ttk.Combobox(project_info_frame, values=['Residential', 'Commercial',
'Infrastructure', 'Industrial' ], font=('Arial', 11) )
    project_type_text.set('Select Project Type')
    project_type_text.grid(row=2, column=5,  sticky="w")
    location_label = ttk.Label(project_info_frame, text="Location:", font='Arial 11',
background='white')
    location_label.grid(row=2, column=6,  pady= 2, sticky="w")
    location_text = ttk.Entry(project_info_frame, font=('Arial', 11))
    location_text.grid(row=2, column=7, sticky="w")
    project_entries_list = [name_text, id_text, bu_text, pf_text, pg_text, contract_type_text,
Owner_text, project_type_text, location_text]

    ### PROJECT BASELINE INFO
    project_baseline_frame = ttk.Labelframe(project_info_label_frame, text="Project Initial
Baseline", style='Custom4.TLabelframe', padding= (10,20))
    project_baseline_frame.grid(row=1, column=0, columnspan=4, padx=10, pady=10,
sticky="nwes")
    original_project_entry_list = []
    label_values = ["Budget At Completion (BAC, $):","Duration (days):","Start Date
(yyyy/mm/dd):","Finish Date (yyyy/mm/dd):"]
    for i, label in enumerate(label_values):
        label = ttk.Label(project_baseline_frame, text=f"{label}", font=("Arial",
11), background='white')
        label.grid(row=0, column=2*i, sticky="w")
        original_entry = ttk.Entry(project_baseline_frame, font=("Arial", 11, ))
        original_entry.grid(row=0, column=2*i+1, padx=(0, 10),sticky="w")
        original_project_entry_list.append(original_entry)
```

```
    project_work_packages_label_frame = ttk.Labelframe(add_window, text="At Work Package
Level", style='Custom5.TLabelframe')
    project_work_packages_label_frame.grid(row=1, column=0, padx=10, pady=10, sticky="nwes")
    project_work_packages_label_frame.grid_rowconfigure(0, weight=1)
    project_work_packages_label_frame.grid_columnconfigure(0, weight=1)
    canvas = tk.Canvas(project_work_packages_label_frame, highlightthickness=0,
background='white')
    canvas.grid(row=0, column=0, sticky="nwes")
    scrollbar = ttk.Scrollbar(project_work_packages_label_frame, orient="vertical",
command=canvas.yview)
    scrollbar.grid(row=0, column=2, sticky="ns")
    canvas.configure(yscrollcommand=scrollbar.set)

    frame = ttk.Frame(canvas)
    canvas_frame = canvas.create_window((0, 0), window=frame, anchor="nw")

    labels = ["Work Package ID", "Work Package Name", "Budget At Completion ($)", "Duration
(days)", "Start Date (yyyy/mm/dd)", "Finish Date (yyyy/mm/dd)"]

    new_work_package_entries = []
    # Function to create widgets for entering a new work package
    def create_new_widgets_for_work_package():
        nrow = len(new_work_package_entries) * 2
        wp_frame = tk.Frame(frame, background=bg_color)
        wp_frame.grid(row=nrow, column=0, columnspan=len(labels), sticky="ew", padx=5, pady=2)
        entries = []
        for i, label_text in enumerate(labels):
            label = ttk.Label(wp_frame, text=label_text, font=("Arial", 11, ),
background='white', borderwidth=1)
            label.grid(row=0, column=i, sticky="w", padx=19, pady=2)
            entry = ttk.Entry(wp_frame, width=20, font=("Arial", 11, ))
            entry.grid(row=1, column=i, sticky="ew", padx=19, pady=2,)
            entries.append(entry)

        new_work_package_entries.append(entries)
        frame.update_idletasks()
        canvas.configure(scrollregion=canvas.bbox("all"))

    # Function to save work packages into project_info
def save_new_project():
        global project_info, new_work_package_entries, work_package_names_entries
        project_id = project_entries_list[1].get()
        if project_entries_list and original_project_entry_list:
            project_info[project_id] = {
                'Project Name': project_entries_list[0].get(),
                'Business Unit': project_entries_list[2].get(),
                'Portfolio Code': project_entries_list[3].get(),
                'Program Code': project_entries_list[4].get(),
                'Contract Type': project_entries_list[5].get(),
                'Owner Internal Code': project_entries_list[6].get(),
                'Project Type': project_entries_list[7].get(),
                'Location': project_entries_list[8].get(),
                'IB-Budget At Completion ($)': original_project_entry_list[0].get(),
                'IB-Duration (days)': original_project_entry_list[1].get(),
                'IB-Start Date': original_project_entry_list[2].get(),
                'IB-Finish Date': original_project_entry_list[3].get(),
                'Work Packages':{} }

            for entries in new_work_package_entries:
                wp_name = entries[1].get()
                project_info[project_id]["Work Packages"][wp_name.upper()] = {
```

```
                    "Work Package ID": entries[0].get(),
                    "IB-Budget At Completion ($)": entries[2].get(),
                    "IB-Duration (days)": entries[3].get(),
                    "IB-Start Date": entries[4].get(),
                    "IB-Finish Date": entries[5].get(),}
            messagebox.showinfo("Info Saved", "New Project saved successfully.")
            save_project_info_to_file()

        else:
            tk.messagebox.showerror("Missing Information", "Please provide all required
information.")
        add_window.destroy()
    Cancel_button = ttk.Button(add_window, text="Cancel", command = lambda:
destroy_window(add_window), padding=20, style='Custom3.TButton')
    Cancel_button.grid(row=3, column=0, padx=200, pady=10, sticky="e")
    Cancel_button.configure(padding=(10,10))

    # Button to save all entered work packages
    save_button = ttk.Button(add_window, text="Save New Project", command=save_new_project,
padding=20, style='Custom2.TButton')
    save_button.grid(row=3, column=0, sticky="e", padx=10, pady=10)
    save_button.configure(padding=(10,10))
```

5. Updating projects information into the GUI

```
def project_updating():
    update_window = tk.Toplevel(root, background='white')
    update_window.title("Update Project")
    project_entry_for_update_frame = ttk.Labelframe(update_window, text="Project Entry for
Updating", style='Custom5.TLabelframe')
    project_entry_for_update_frame.grid(row=0, column=0, padx=10, pady=10, sticky="nwes")
    projectid_label = ttk.Label(project_entry_for_update_frame, text="Select Project ID:",
font='Arial 11 bold', background='white')
    projectid_comb = ttk.Combobox(project_entry_for_update_frame, values=
list(project_info.keys()), state="readonly", font='Arial 11 bold',)
    projectid_comb.set("Select Project")
    projectid_label.grid(row=0, column=0, padx=10, pady=10, sticky="w")
    projectid_comb.grid(row=0, column=1, padx=10, pady=10, sticky="w")
    projectid_button = ttk.Button(project_entry_for_update_frame, text="Display Project
Information", command=lambda: populate_update_frame(projectid_comb.get()),
style='Custom1.TButton')
    projectid_button.grid(row=0, column=2, padx=10, pady=10, sticky="w")

    project_info_label_frame = ttk.Labelframe(update_window, text="At Project Level",
style='Custom5.TLabelframe')
    project_info_label_frame.grid(row=1, column=0, padx=10, pady=10, sticky="nwes")

    def populate_update_frame(selected_project):
        global work_package_list, current_project_entry_list, current_work_package_entries

    ### PROJECT LEVEL
        project_info_frame = ttk.Labelframe(project_info_label_frame, text="Project
Information", style='Custom4.TLabelframe', padding= (10,20))
        project_info_frame.grid(row=0, column=0, columnspan=4, padx=10, pady=10,
sticky="nwes")
        width_label = 19
        width_entries = 12
        name_label = ttk.Label(project_info_frame, text="Project Name", font='Arial 11 bold',
background='white', width=width_label)
        name_label.grid(row=0, column=0, padx=10, pady=0, sticky="w")
```

```
        name_text = ttk.Label(project_info_frame, text=
project_info[selected_project]["Project Name"], font='Arial 11', background='white', )
        name_text.grid(row=0, column=1, columnspan=5, padx=10, pady=0, sticky="w")

        pid_label = ttk.Label(project_info_frame, text="Project ID", font='Arial 11 bold',
background='white', width=width_label)
        pid_label.grid(row=1, column=0, padx=10, pady=0, sticky="w")
        pid_text = ttk.Label(project_info_frame, text=selected_project, font='Arial 11 ',
background='white', width=width_entries)
        pid_text.grid(row=1, column=1, padx=10, pady=0, sticky="w")

        bu_label = ttk.Label(project_info_frame, text="Business Unit Code", font='Arial 11
bold', background='white', width=width_label)
        bu_label.grid(row=1, column=2, padx=10, pady=0, sticky="w")
        bu_text = ttk.Label(project_info_frame, text=project_info[selected_project]["Business
Unit"], font='Arial 11 ', background='white', width=width_entries)
        bu_text.grid(row=1, column=3, padx=10, pady=0, sticky="w")

        pf_label = ttk.Label(project_info_frame, text="Portfolio Code", font='Arial 11 bold',
background='white', width=width_label)
        pf_label.grid(row=1, column=4, padx=10, pady=0, sticky="w")
        pf_text = ttk.Label(project_info_frame, text=project_info[selected_project]["Portfolio
Code"], font='Arial 11 ', background='white', width=width_entries)
        pf_text.grid(row=1, column=5, padx=10, pady=0, sticky="w")

        pg_label = ttk.Label(project_info_frame, text="Program Code", font='Arial 11 bold',
background='white', width=width_label)
        pg_label.grid(row=1, column=6, padx=10, pady=0, sticky="w")
        pg_text = ttk.Label(project_info_frame, text=project_info[selected_project]["Program
Code"], font='Arial 11 ', background='white', width=width_entries)
        pg_text.grid(row=1, column=7, padx=10, pady=0, sticky="w")

        contract_type_label = ttk.Label(project_info_frame, text="Contract Type", font='Arial
11 bold', background='white', width=width_label)
        contract_type_label.grid(row=2, column=0, padx=10, pady=0, sticky="w")
        contract_type_label = ttk.Label(project_info_frame,
text=project_info[selected_project]["Contract Type"], font='Arial 11 ', background='white',
width=width_entries)
        contract_type_label.grid(row=2, column=1, padx=10, pady=0, sticky="w")

        Owner_label = ttk.Label(project_info_frame, text="Owner Internal Code", font='Arial 11
bold', background='white', width=width_label)
        Owner_label.grid(row=2, column=2, padx=10, pady=0, sticky="w")
        Owner_text = ttk.Label(project_info_frame, text=project_info[selected_project]["Owner
Internal Code"], font='Arial 11 ', background='white', width=width_entries)
        Owner_text.grid(row=2, column=3, padx=10, pady=0, sticky="w")

        project_type_label = ttk.Label(project_info_frame, text="Project Type", font='Arial 11
bold', background='white', width=width_label)
        project_type_label.grid(row=2, column=4, padx=10, pady=0, sticky="w")
        project_type_text = ttk.Label(project_info_frame,
text=project_info[selected_project]["Owner Internal Code"], font='Arial 11 ',
background='white', width=width_entries)
        project_type_text.grid(row=2, column=5, padx=10, pady=0, sticky="w")

        location_label = ttk.Label(project_info_frame, text="Location", font='Arial 11 bold',
background='white', width=width_label)
        location_label.grid(row=2, column=6, padx=10, pady=0, sticky="w")
        location_text = ttk.Label(project_info_frame,
text=project_info[selected_project]["Location"], font='Arial 11 ', background='white',
width=width_entries)
```

```
        location_text.grid(row=2, column=7, padx=10, pady=0, sticky="w")

        project_baselines_frame = ttk.Labelframe(project_info_label_frame, text="Project
Baselines", style='Custom4.TLabelframe', padding= (10,10))
        project_baselines_frame.grid(row=1, column=0, padx=10, pady=10, sticky="nwes")

        original_label = ttk.Label(project_baselines_frame, text="Initial Baseline",
font=("Arial", 11, 'bold'), background='white', width=20,)
        original_label.grid(row=1, column=0, padx=10 )
        current_label = ttk.Label(project_baselines_frame, text="Current Baseline",
font=("Arial", 11, 'bold'), background='white', width=20)
        current_label.grid(row=2, column=0, padx=10)

        label_values = ["Budget At Completion ($)", "Duration (days)", "Start Date
(yyyy/mm/dd)", "Finish Date (yyyy/mm/dd)" ]
        text_values = [project_info[selected_project]["IB-Budget At Completion ($)"],
                       project_info[selected_project]["IB-Duration (days)"],
                       project_info[selected_project]["IB-Start Date"],
                       project_info[selected_project]["IB-Finish Date"] ]

        current_project_entry_list = []

        for i, (label, text) in enumerate(zip(label_values, text_values), start=1):
            label = ttk.Label(project_baselines_frame, text=f"{label}", font=("Arial", 11,
"bold"), background='white', width=25)
            label.grid(row=0, column=i, sticky='nesw', padx=20)
            text_label = ttk.Label(project_baselines_frame, text=f"{text}", font=("Arial",
11), background='white')
            text_label.grid(row=1, column=i,sticky='nesw' , padx=20)

            entry = ttk.Entry(project_baselines_frame, font=("Arial", 11),)
            entry.grid(row=2, column=i,sticky='nesw', padx=20 )

            current_project_entry_list.append(entry)

        ### WORK PACKAGES
        work_package_frame = ttk.Labelframe(update_window, text="At Work Package Level",
style='Custom5.TLabelframe', padding= (10,20))
        work_package_frame.grid(row=2, column=0, padx=10, sticky="nwes")
        work_package_frame.grid_rowconfigure(0, weight=1)
        work_package_frame.grid_columnconfigure(0, weight=1)
        canvas = tk.Canvas(work_package_frame, highlightthickness=0, background='white',
height=220)
        canvas.grid(row=0, column=0, sticky="nwes")
        scrollable_frame = ttk.Frame(canvas, style='Custom.TFrame')
        canvas.create_window((0, 0), window=scrollable_frame, anchor='nw')
        scrollbar = ttk.Scrollbar(work_package_frame, orient="vertical", command=canvas.yview)
        scrollbar.grid(row=0, column=2, sticky="ns")
        canvas.configure(yscrollcommand=scrollbar.set)

        def on_configure(event):
            canvas.configure(scrollregion=canvas.bbox("all"))

        # Bind the on_configure function to the canvas's configure event
        canvas.bind("<Configure>", on_configure)

        labels = ["Work Package", "Budget At Completion ($)", "Duration (days)", "Start Date
(yyyy/mm/dd)", "Finish Date (yyyy/mm/dd)"]


        for i, label in enumerate(labels):
```

```
            label_widget = ttk.Label(scrollable_frame, text=label, font=("Arial", 11, "bold"),
background='white', width=25, anchor='center')
            label_widget.grid(row=0, column=i, sticky="w", padx=20)

        work_package_list = list(project_info[selected_project]["Work Packages"].keys())
        current_work_package_entries = []

        row=1
        for column, work_package in enumerate(work_package_list):
            work_package_name_label = ttk.Label(scrollable_frame, text=
project_info[selected_project]["Work Packages"][work_package.upper()]["Work Package ID"] + f"-
{work_package}", font=("Arial", 11,'bold'), background='white', )
            work_package_name_label.grid(row=row, column=0, sticky="nwse")

            original_wp_label = ttk.Label(scrollable_frame, text="Initial Baseline",
font=("Arial", 11), background='white')
            original_wp_label.grid(row=row+1, column=0, padx=20, sticky="e")
            current_wp_label = ttk.Label(scrollable_frame, text="Current Baseline",
font=("Arial", 11), background='white')
            current_wp_label.grid(row=row+2, column=0, padx=20, sticky="e")
            original_cost_text = ttk.Label(scrollable_frame,
text=project_info[selected_project]["Work Packages"][work_package.upper()]["IB-Budget At
Completion ($)"], font=("Arial", 11), background='white')
            original_cost_text.grid(row=row+1, column=1, )
            original_duration_text = ttk.Label(scrollable_frame,
text=project_info[selected_project]["Work Packages"][work_package.upper()]["IB-Duration
(days)"], font=("Arial", 11), background='white')
            original_duration_text.grid(row=row+1, column=2, )
            original_startdate_text = ttk.Label(scrollable_frame,
text=project_info[selected_project]["Work Packages"][work_package.upper()]["IB-Start Date"],
font=("Arial", 11), background='white')
            original_startdate_text.grid(row=row+1, column=3, )
            original_finishdate_text = ttk.Label(scrollable_frame, text=
project_info[selected_project]["Work Packages"][work_package.upper()]["IB-Finish Date"],
font=("Arial", 11), background='white')
            original_finishdate_text.grid(row=row+1, column=4, )

            current_cost_text = ttk.Entry(scrollable_frame, width=20,font=("Arial", 11) )
            current_cost_text.grid(row=row+2, column=1,)
            current_duration_text = ttk.Entry(scrollable_frame, width=20 ,font=("Arial", 11) )
            current_duration_text.grid(row=row+2, column=2, )
            current_startdate_text = ttk.Entry(scrollable_frame, width=20,font=("Arial", 11) )
            current_startdate_text.grid(row=row+2, column=3,  )
            current_finishdate_text = ttk.Entry(scrollable_frame, width=20,font=("Arial", 11)
)
            current_finishdate_text.grid(row=row+2, column=4,  )

            current_work_package_entries.append([current_cost_text, current_duration_text,
current_startdate_text, current_finishdate_text])
            row += 3

        update_window.update_idletasks()
        canvas.config(scrollregion=canvas.bbox("all"))

    def save_updated_project_info(selected_project):
        global project_info
        # Ensure there is a selected project, and the necessary entries are filled
        if not current_project_entry_list or not all(entry.get() for entry in
current_project_entry_list):
            tk.messagebox.showwarning("Missing Information", "Please provide all required
information for the Project.")
```

```
            return

        # Update project-level information
        attributes = ['CB-Budget At Completion ($)', 'CB-Duration (days)', 'CB-Start Date',
'CB-Finish Date']
        project_info[selected_project].update({attr: current_project_entry_list[i].get() for
i, attr in enumerate(attributes)})

        # Update work package-level information
        for pos, wp_att in enumerate(current_work_package_entries):
            wp_name = work_package_list[pos].upper()
            project_info[selected_project]["Work Packages"][wp_name].update({
                attributes[i]: wp_att[i].get() for i in range(4)
            })

        # Clear entry widgets after successful update
        for entry in current_project_entry_list + [item for sublist in
current_work_package_entries for item in sublist]:
            entry.delete(0, "end")

        tk.messagebox.showinfo("Info Saved", "Information saved successfully.")
        save_project_info_to_file()
        update_window.destroy()

    def on_save_click(project_id):
        save_updated_project_info(project_id)
        current_datetime = datetime.now()
        save_project_info_to_file()
        project_info[project_id]["CB_Storage_date"] = current_datetime

    Cancel_button = ttk.Button(update_window, text="Cancel", command = lambda:
destroy_window(update_window), padding=20, style='Custom3.TButton')
    Cancel_button.grid(row=3, column=0, padx=250, pady=10, sticky="e")
    Cancel_button.configure(padding=(10,10))

    update_project_button = ttk.Button(update_window, text="Update Project Information",
command=lambda: on_save_click(projectid_comb.get()), padding=20, style='Custom2.TButton')
    update_project_button.grid(row=3, column=0, padx=10, pady=10, sticky="e")
    update_project_button.configure(padding=(10,10))
```

6. Inputs for the Forecasting model from the Project Tracking

```
def enter_WP_reports():
    global project_id_combobox_frame0, report_combobox_frame0, report_date_entry,
project_selected_progress

    for widget in progress_frame.winfo_children():
        widget.destroy()

    frame0 = ttk.Labelframe(progress_frame, text='Progress Period Data',
style='Custom2.TLabelframe')
    frame0.grid(row=0, column=0, padx=0, pady=0,  sticky="nwes" )

    project_list = [f"Project {code}: {list(name.values())[0]}" for code, name in
project_info.items()]
    project_id_combobox_frame0 = ttk.Combobox(frame0, values=project_list, font=('Arial', 13),
width=45, justify='left')
    project_id_combobox_frame0.grid(row=0, column=0, padx=(10,100), pady=10, sticky="w")
    project_id_combobox_frame0.set('Please select a project')

    def on_project_selected_progress(event):
```

```
        global project_selected_progress
        selection = project_id_combobox_frame0.get()
        match = re.search(r"Project ([\w-]+):", selection)
        if match:
            project_selected_progress = match.group(1)
project_id_combobox_frame0.bind('<<ComboboxSelected>>', on_project_selected_progress)
    report_date_label = ttk.Label(frame0, text="Ending Date:", font=('Arial', 13),
background='white')
    report_date_label.grid(row=0, column=3, padx=10, pady=10)

  def open_calendar():

        def on_date_selected():

            report_date_entry.set(calendar.selection_get().strftime("%Y/%m/%d"))
            calendar_window.destroy()
        calendar_window = tk.Toplevel(root)
        calendar_window.title("Select Date")
        calendar_window.geometry(f"+{report_date_field.winfo_rootx()}+{report_date_field.winfo
_rooty() + report_date_field.winfo_height()}")
        calendar_window.grab_set()
        calendar_window.transient(root)

        calendar = Calendar(calendar_window, selectmode='day', date_pattern='y-mm-dd')
        calendar.pack(pady=10, padx=10)

        ok_button = ttk.Button(calendar_window, text="OK", command=on_date_selected)
        ok_button.pack()

    report_date_entry = tk.StringVar()
    report_date_field = ttk.Entry(frame0, font=('Arial', 13), textvariable=report_date_entry)
    report_date_field.grid(row=0, column=4, padx=(10,0), pady=10, sticky='n')
    deploy_button = ttk.Button(frame0, text="▼", command=open_calendar)
    deploy_button.grid(row=0, column=5, padx=(0,0))

    set_data_button = ttk.Button(frame0, text="Enter Data...", command=lambda:
validation_and_entries_creation(project_selected_progress), style='Custom.TButton')
    set_data_button.grid(row=0, column=7, padx=10, pady=0, sticky="nwse")

    def validation_and_entries_creation(selected_project):
        global report_number_key
        report_date = report_date_entry.get()
        project_reports_info = project_info.setdefault(selected_project,
{}).setdefault("Project Reports Info", {})
        dates_exist = any("Date" in report_info for report_info in
project_reports_info.values())
        if dates_exist:
            max_report_number = max(int(k.split()[-1]) for k in project_reports_info.keys())
            last_report_date = project_reports_info[f'Project Report Number
{max_report_number}']['Date']
            if datetime.datetime.strptime(report_date, "%Y/%m/%d") <=
datetime.datetime.strptime(last_report_date, "%Y/%m/%d"):
                messagebox.showerror("Date Error", "Reporting Date must be later than the
previous reporting dates entered.")
            else:
                report_number_key=f'Project Report Number {max_report_number+1}'
                project_reports_info[report_number_key] = {'Date': report_date}
                create_work_package_entries(selected_project)

        else:
            report_number_key=f'Project Report Number 1'
```

```
            project_reports_info[report_number_key] = {'Date': report_date}
            create_work_package_entries(selected_project)
    def create_work_package_entries(selected_project):
        global frame1, save_data_button, go_to_step_2_button, entries, headers_list,
work_packages_in_selected_project

        frame1 = ttk.Labelframe(progress_frame, text='Inputs for Machine Learning Model
Forecasting', padding=(10,10), style='Custom9.TLabelframe')
        frame1.grid(row=1, column=0, padx=0, pady=20, sticky="nwse")

        project_reports_info = project_info[project_selected_progress]["Project Reports Info"]
        max_report_number = max(int(k.split()[-1]) for k in project_reports_info.keys())
        next_reporting_period = f"Reporting Period {max_report_number}: Ending
{report_date_entry.get()}"
        ttk.Label(frame1, text=next_reporting_period, font=("Arial", 11, )).grid(row=0,
column=0, columnspan=5, pady=10,sticky='nwes')
        work_packages_in_selected_project = list(project_info[selected_project]["Work
Packages"].keys())

        #Row labels-----------------------------------------------------:
        for i, work_package in enumerate(work_packages_in_selected_project, start=2):
            work_package_id_label = ttk.Label(frame1,
text=project_info[selected_project]["Work Packages"][work_package]["Work Package ID"],
font=("Arial", 10, ), width= 10)
            work_package_id_label.grid(row=i, column=0, sticky='nwes')
            work_package_name_label = ttk.Label(frame1, text=work_package, font=("Arial", 10,
), justify='left')
            work_package_name_label.grid(row=i, column=1, )

        #Headers labels-------------------------------------------------:
        headers_list = ['ID', 'WP Name', 'Status', 'Period Number', 'AD to date',
                        'Cum EV ($)', 'Cum ES (weeks)', 'TPI' ,
                        'Actual Start Date', 'Actual Finish Date']
        for i, header in enumerate(headers_list):
            if header == 'Status' or header == 'ID':
                ttk.Label(frame1, text=header, font=("Arial", 11, ),
anchor='center').grid(row=1, column=i, pady=20,sticky='nwes')

            else:
                ttk.Label(frame1, text=header, font=("Arial", 11, ),
anchor='center').grid(row=1, column=i, pady=20,)

            frame1.grid_columnconfigure(i, weight=1)

        ## Entries-------------------------------------------------------:
        entries = []
        for j in range(2, len(work_packages_in_selected_project)+2):   ### "j" rows
            row_entries = []
            for i in range(2,len(headers_list)):  ### "i" is column
                if i==2:
                    att_entry = ttk.Combobox(frame1, values=["Non Started", "In Progress",
"Finished"], font=("Arial", 11,), width=10)
                else:
                    att_entry = ttk.Entry(frame1,  font=("Arial", 11,), width=15)
                att_entry.grid(row=j, column=i)
                row_entries.append(att_entry)
            entries.append(row_entries)

        save_data_button = ttk.Button(progress_frame, text="Save", command=lambda:
store_data(project_selected_progress), style='Custom2.TButton')
        save_data_button.grid(row=3, column=0, padx=20, pady=20, sticky="e" )
```

```
        def cancel_function():
            del project_info[selected_project]["Project Reports Info"][report_number_key]
            cancel_button.destroy()
            save_data_button.destroy()
            frame1.grid_forget()
            show_frame(project_hub_frame)

        cancel_button = ttk.Button(progress_frame, text="Cancel", command=cancel_function,
style='Custom3.TButton')
        cancel_button.grid(row=3, column=0, padx=150, pady=20, sticky="e" )
        save_data_button.configure(padding=(10, 10))
        cancel_button.configure(padding=(10, 10))

def store_data(selected_project):
    if 'Work Packages' not in project_info[selected_project]["Project Reports
Info"][report_number_key]:
        project_info[selected_project]["Project Reports Info"][report_number_key]['Work
Packages'] = {}

    for i, work_package in enumerate(work_packages_in_selected_project):
        project_info[selected_project]["Project Reports Info"][report_number_key]['Work
Packages'][work_package] = {
            'ID': project_info[selected_project]["Work Packages"][work_package]["Work Package
ID"],
            'Status': entries[i][0].get(),
            'Period Number': entries[i][1].get(),
            'Actual Duration To Date (days)': entries[i][2].get(),
            'Cumulative EV': entries[i][3].get(),
            'Cumulative ES': entries[i][4].get(),
            'TPI': entries[i][5].get(),
            'Start Date Actual': entries[i][6].get(),
            'Finish Date Actual': entries[i][7].get(),
        }

    save_project_info_to_file()

    for work_package in work_packages_in_selected_project:
        try:
            Budget_item = project_info[selected_project]['Work Packages'][work_package]['CB-
Budget At Completion ($)']
        except KeyError:
            Budget_item = project_info[selected_project]['Work Packages'][work_package]['IB-
Budget At Completion ($)']

        data_for_dict = {
        'Period Number': [project_info[selected_project]["Project Reports
Info"][report_number_key]['Work Packages'][work_package]['Period Number']],
        'Actual Duration To Date (days)':[project_info[selected_project]["Project Reports
Info"][report_number_key]['Work Packages'][work_package]['Actual Duration To Date (days)']],
        'Cumulative Earned Value ($)': [project_info[selected_project]["Project Reports
Info"][report_number_key]['Work Packages'][work_package]['Cumulative EV']],
        'Cumulative Earned Schedule (weeks)': [project_info[selected_project]["Project Reports
Info"][report_number_key]['Work Packages'][work_package]['Cumulative ES']],
        'Time Performance Index': [project_info[selected_project]["Project Reports
Info"][report_number_key]['Work Packages'][work_package]['TPI'] ],
        'Budget': [Budget_item],
        'Duration to Complete (days)': None}

        df = pd.DataFrame(data_for_dict)
        df.astype(float)
```

```
        if os.path.exists(f'{selected_project}_{work_package}_data.csv'):
            df_1 = pd.read_csv(f'{selected_project}_{work_package}_data.csv')
            df3 = pd.concat([df_1, df], ignore_index=True)
            df3.to_csv(f'{selected_project}_{work_package}_data.csv', index=False)
        else:
            df.to_csv(f'{selected_project}_{work_package}_data.csv', index=False)

    clear_entries()
    messagebox.showinfo("Success", "Data Stored Successfully")
```

7. The forecasting of Work Packages: Integrating forecasting model and incorporating Gantt and line charts and, a summary table.

```
##### Filtering data for forecasting model ###
def forecasting_file(selected_project, selected_package, period):
    df = pd.read_csv(f'{selected_project}_{selected_package}_data.csv')
    df = df.head(period)
    df_reduced = df[df['Period Number'] != 0].reset_index(drop=True)
    df_predic_1 = df_reduced.drop(['Period Number'],axis=1)
    return df_predic_1, df

##### VARIABLES FOR NORMALIZATION ###
def normalization(df):
    values = df.values.astype('float32')
    scaler = MinMaxScaler()
    values_scaled = scaler.fit_transform(values)
    df_scaled = pd.DataFrame(values_scaled)
    return df_scaled
##### GROUPING FOR MODELLING ###
def df_to_X_y(df, past):
  df_as_np = df.to_numpy()
  X = []
  y = []
  for i in range(len(df_as_np)-past):
    row = [r for r in df_as_np[i:i+past]]
    X.append(row)
    label = df_as_np[i+past][df.shape[1]-1]
    y.append(label)
  return np.array(X), np.array(y)
##### VARIABLES FOR DENORMALIZATION ###
def factors_for_denormalization(df):
    Max_orig = df.iloc[:,5].max()
    Min_orig = df.iloc[:,5].min()
    Delta = Max_orig-Min_orig
    return Delta, Min_orig

def Prediction(selected_project, selected_package, period):
    predict_value = None
    df_predic, df_in_csv_file = forecasting_file(selected_project, selected_package, period)

    try:
        bl_duration = float(project_info[selected_project]['Work
Packages'][selected_package]['CB-Duration (days)'])
    except KeyError:
        bl_duration = float(project_info[selected_project]['Work
Packages'][selected_package]['IB-Duration (days)'])

    if len(df_predic) >= 4:
        df=df_predic.tail(4)
```

```
        df = df_predic
        to_complete = float(bl_duration) - float(df.iloc[-1,0])
        df.iloc[-1,-1] = to_complete
        if df.iloc[:, -1].isnull().any():
            messagebox.showwarning("Warning", "Ensure that previous Duration to Completion
(DTC) were filled")
        else:
            Delta_1p, Min_orig_1p = factors_for_denormalization(df)
            df_predict_scaled = normalization(df)
            model = load_model(f'{selected_package}_model.h5')
            X1p, y1p = df_to_X_y(df_predict_scaled, 3)
            Xpredict=model.predict(X1p).flatten()
            predicted_list = Xpredict*Delta_1p+Min_orig_1p
            predict_value = predicted_list[-1]
    else:
        predict_value = float(bl_duration) - float(df_predic.iloc[-1,0])

    period_value = df_predic.index[-1] + 1
    df1 = pd.read_csv(f'{selected_project}_{selected_package}_data.csv')
    df1.loc[df1['Period Number'] == period_value, 'Duration to Complete (days)'] =
predict_value
    df1.to_csv(f'{selected_project}_{selected_package}_data.csv', index=False)

    return predict_value

def work_package_forecasting():

    for widget in forecasting_work_package_frame.winfo_children():
        widget.destroy()
    wp_prediction_frame = ttk.Labelframe(forecasting_work_package_frame, text='Forecasting per
Work Package', style='Custom2.TLabelframe')
    wp_prediction_frame.grid(row=0, column=0, padx=0, pady=0, sticky='nwes')

    buttons_frame = ttk.Labelframe(forecasting_work_package_frame, text="Work Packages",
style='Custom7.TLabelframe')
    buttons_frame.grid(row=1, column=0, sticky="nsew", padx=0, pady=5)

    ### Getting Predictions----------------------###
    def getting_predictions(project, period):
        dic = project_info[project]["Project Reports Info"][f'Project Report Number {period}']
        if 'Predictions' not in dic:
            wpackages = list(project_info[project]["Work Packages"].keys())
            predicted_values_dict = {key: None for key in wpackages}
            for wpackage in wpackages:
                if dic['Work Packages'][wpackage]["Status"] == 'In Progress':
                    predicted_value = Prediction(project, wpackage, period)
                    predicted_values_dict[wpackage] = float(math.ceil(predicted_value))
                else:
                    predicted_values_dict[wpackage] = 0
            dic['Predictions'] = predicted_values_dict

            save_project_info_to_file()
  def show_chart(proj, package_name, period_selected):
        fig_gantt.clf()
        fig_line.clf()

        ax = fig_gantt.add_subplot(111)
        ax1 = fig_line.add_subplot(111)

        dic_temp_2 = project_info[proj]["Project Reports Info"][f'Project Report Number
{period_selected}']['Work Packages'][package_name]
```

```
        df1=pd.read_csv(f'database {package_name}.csv')

        df1 = df1[df1['Period'] == period_selected]
        df1=df1.reset_index()

        df=pd.DataFrame(columns=['Item', 'Start', 'Finish'])
        Item_list = ['Current BL', 'Actual', 'Forecast']
        if df1.at[0,'Status']== 'In Progress':
            Start_list = [df1.at[0,'BL Start Date'], df1.at[0,'Start date'],
df1.at[0,'Reporting Date']]
            Finish_list = [df1.at[0,'BL Finish Date'], df1.at[0,'Reporting Date'],
df1.at[0,'Forecasted Finish Date']]
        elif df1.at[0,'Status'] == 'Non Started':
            Start_list = [df1.at[0,'BL Start Date'], df1.at[0,'Reporting Date'],
df1.at[0,'Reporting Date']]
            Finish_list = [df1.at[0,'BL Finish Date'], df1.at[0,'Reporting Date'],
df1.at[0,'Reporting Date']]
        elif df1.at[0,'Status'] == 'Finished':
            Start_list = [df1.at[0,'BL Start Date'], df1.at[0,'Start date'],
df1.at[0,'Reporting Date']]
            Finish_list = [df1.at[0,'BL Finish Date'], df1.at[0,'Finish Date'],
df1.at[0,'Reporting Date']]

        df['Item'] = Item_list
        df['Start'] = pd.to_datetime(Start_list, )
        df['Finish'] = pd.to_datetime(Finish_list, )
        df['Duration'] = (df['Finish']-df['Start']).dt.days
        df['Start'] = mdates.date2num(df['Start'])
        df['Finish'] = mdates.date2num(df['Finish'])

        # Create a Gantt chart
        color_dict = {'Current BL': '#EAEE1B','Actual': '#271BE6',
        'Forecast': '#9F2CEA'}
        labeled_items = set()

        for index, row in df.iterrows():
            item_color = color_dict[row['Item']]
            # Only add a label if this item hasn't been labeled before
            if row['Item'] not in labeled_items:
                ax.barh(row['Item'], row['Finish'] - row['Start'], left=row['Start'],
height=0.5, color=item_color, edgecolor='black', label=row['Item'])
                labeled_items.add(row['Item'])
            else:
                ax.barh(row['Item'], row['Finish'] - row['Start'], left=row['Start'],
height=0.5, color=item_color, edgecolor='black')


        ax.xaxis_date()
        ax.xaxis.set_major_locator(mdates.MonthLocator())
        ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%b-%d'))
        ax.set_xlabel('Date', fontdict={'family': 'Arial', 'size': 11})
        ax.set_yticks(range(len(df['Item'])))
        ax.set_yticklabels(df['Item'], fontdict={'family': 'Arial', 'size': 11})
        ax.yaxis.set_major_locator(FixedLocator(range(len(df['Item']))))
        for label in ax.get_xticklabels():
            label.set_fontname('Arial')
            label.set_fontsize(10)

        today1 = df1.at[0,'Reporting Date']
        today1 = datetime.datetime.strptime(today1, '%Y-%m-%d')
```

```
        ax.axvline(x=today1, color='red', linestyle='-')
        ax.text(today1, 0.5, 'Data Date', color='red', fontsize=10, ha='left', family='Arial')
        ax.legend(fontsize=11, loc='upper center', frameon=False, bbox_to_anchor=(0.5, 1.3),
prop={'family': 'Arial', 'size': 10}, ncol=len(labeled_items))
        ax.invert_yaxis()
        ax.grid(axis='x', zorder=0)
        ax.set_axisbelow(True)
        fig_gantt.autofmt_xdate()
        fig_gantt.subplots_adjust(left=0.05, right=0.95, top=0.95, bottom=0.05)
        fig_gantt.tight_layout()
        chart_canvas.draw()

        ### Line Chart
        if dic_temp_2["Status"] == 'In Progress':
            df2=pd.read_csv(f'database {package_name}.csv')
            df2 = df2[df2['Tracking Period'] != 0]
            x_values = df2['Tracking Period']
            y_values = df2['Finish Date Variance']
            ax1.plot(x_values, y_values, marker='o', linestyle='-', color='blue')
            ax1.set_xlim(left=x_values.min(), right=x_values.max())
            ax1.set_xticks(x_values)
            ax1.set_xticklabels(x_values, fontdict={'family': 'Arial', 'size': 10} )

            min_y = y_values.min()
            max_y = y_values.max()
            y_ticks = np.arange(min_y, max_y+1, 10)
            ax1.set_yticks(y_ticks)
            ax1.set_yticklabels(y_ticks, fontdict={'family': 'Arial', 'size': 10} )
           # Formatting the date axis
            ax1.set_xlabel('Work Package Execution Period', fontdict={'family': 'Arial',
'size': 9})
            ax1.set_ylabel('Deviation (days)', fontdict={'family': 'Arial', 'size': 11})
            fig_line.tight_layout()
            ax1.grid(True)
            ax1.set_axisbelow(True)
            line_chart_canvas.draw()

        else:
            ax1.text(0.5, 0.5, 'No chart to display as \nthe work package has not \nstarted or
has already finished.',
                     horizontalalignment='center', verticalalignment='center',
                     transform=ax1.transAxes,fontdict={'family': 'Arial', 'size': 12,
'weight': 'normal'})
        line_chart_canvas.draw()

    def show_data_and_charts(proj, package_name, period_selected):

        dic_temp = project_info[proj]['Work Packages'][package_name]
        data_chart_frame = ttk.Labelframe(forecasting_work_package_frame,
text=f"{dic_temp['Work Package ID']} {package_name}", style='Custom7.TLabelframe')
        data_chart_frame.grid(row=2, column=0, sticky="nsew", padx=0, pady=5)
        data_chart_frame.grid_columnconfigure(2, weight=1)

        chart_frame = ttk.Labelframe(data_chart_frame, text="Gantt Chart",
style='Custom11.TLabelframe')
        chart_frame.grid(row=1, column=0, sticky="nsew", padx=5, pady=5)
        line_chart_frame = ttk.Labelframe(data_chart_frame, text="Finish Date Deviation per
Period", style='Custom11.TLabelframe')
        line_chart_frame.grid(row=1, column=1, sticky="nsew", padx=5, pady=5)
        chart_frame.grid_rowconfigure(1, weight=1)
        chart_frame.grid_columnconfigure(1, weight=1)
```

```
        line_chart_frame.grid_rowconfigure(0, weight=1)
        line_chart_frame.grid_columnconfigure(0, weight=1)
        fig_gantt = Figure(figsize=(6, 4), dpi=100)
        chart_canvas = FigureCanvasTkAgg(fig_gantt, master=chart_frame)
        chart_canvas.get_tk_widget().grid(row=0, column=0, sticky="nsew")
        fig_line = Figure(figsize=(4, 4), dpi=100)
        line_chart_canvas = FigureCanvasTkAgg(fig_line, master=line_chart_frame)
        line_chart_canvas.get_tk_widget().grid(row=0, column=0, sticky="nsew")
        df = pd.DataFrame({'Period': range(1, period_selected + 1)})
        df['ID'] = dic_temp['Work Package ID']
        df['WP Name'] = package_name

        def get_status(period):
            return project_info[proj]["Project Reports Info"][f'Project Report Number
{period}']['Work Packages'][package_name]['Status']
        df['Status'] = df['Period'].apply(get_status)
        df['Tracking Period'] = 0
        df.loc[df['Status'] == 'In Progress', 'Tracking Period'] = (df['Status'] == 'In
Progress').cumsum()

        def get_reporting_date(period):
            return project_info[proj]["Project Reports Info"][f'Project Report Number
{period}']['Date']
        df['Reporting Date'] = df['Period'].apply(get_reporting_date)
        df['Reporting Date'] = pd.to_datetime(df['Reporting Date'], format='%Y/%m/%d')

        def get_actual_start(period):
            return project_info[proj]["Project Reports Info"][f'Project Report Number
{period}']['Work Packages'][package_name]['Start Date Actual']
        df['Start date'] = df['Period'].apply(get_actual_start)
        df['Start date'] = pd.to_datetime(df['Start date'], format='%Y/%m/%d',
errors='coerce')

        def get_actual_finish(period):
            return project_info[proj]["Project Reports Info"][f'Project Report Number
{period}']['Work Packages'][package_name]['Finish Date Actual']
        df['Finish Date'] = df['Period'].apply(get_actual_finish)
        df['Finish Date'] = pd.to_datetime(df['Finish Date'], format='%Y/%m/%d',
errors='coerce')

        #### BL Duration
        df['BL Duration'] = None
        for i, row in df.iterrows():
            if row['Status'] == 'In Progress':
                try:
                    given_value1 =
datetime.datetime.strptime(project_info[proj]["CB_Storage_date"], "%Y/%m/%d")
                    value3 = dic_temp["IB-Duration (days)"]
                    value4 = dic_temp["CB-Duration (days)"]
                    df.at[i, 'BL Duration'] = value3 if row['Reporting Date'] < given_value1
else value4
                except KeyError:
                    df.at[i, 'BL Duration'] = dic_temp["IB-Duration (days)"]
            elif row['Status'] == 'Non Started':
                try:
                    df.at[i, 'BL Duration'] = dic_temp["CB-Duration (days)"]
                except KeyError:
                    df.at[i, 'BL Duration'] = dic_temp["IB-Duration (days)"]
            elif row['Status'] == 'Finished':
                try:
                    df.at[i, 'BL Duration'] = dic_temp["CB-Duration (days)"]
```

```
                except KeyError:
                    df.at[i, 'BL Duration'] = dic_temp["IB-Duration (days)"]

        df['BL Duration'] = df['BL Duration'].astype(float)

        ### Forecasted Duration
        def get_ADdate(period):
            return project_info[proj]["Project Reports Info"][f'Project Report Number
{period}']['Work Packages'][package_name]['Actual Duration To Date (days)']
        df['AD_to_date'] = df['Period'].apply(get_ADdate)
        df['AD_to_date'] = df['AD_to_date'].astype(int)

        def get_predicted(period):
            try:
                return project_info[proj]["Project Reports Info"][f'Project Report Number
{period}']['Predictions'][package_name]
            except KeyError:
                return 0
        df['Predicted_value'] = df['Period'].apply(get_predicted)
        df['Predicted_value']=np.ceil(df['Predicted_value'])

        ### Then:
        df['Forecasted Duration'] = df['AD_to_date']+df['Predicted_value']

        for i, row in df.iterrows():
            if row['Status'] == 'In Progress' and row['Predicted_value'] == 0:
                df.at[i, 'Forecasted Duration'] = df.at[i, 'BL Duration']
            elif row['Status'] == 'Non Started':
                df.at[i, 'Forecasted Duration'] = df.at[i, 'BL Duration']
            elif row['Status'] == 'Finished':
                df.at[i, 'Forecasted Duration'] = (df.at[i,'Finish Date'] - df.at[i,'Start
date']).days

        ### BL Finish Date
        df['BL Finish Date'] = None
        df['BL Start Date'] = None

        for i, row in df.iterrows():
            if row['Status'] == 'In Progress':
                try:
                    given_value1 = datetime.datetime.strptime(project_info[proj]
["CB_Storage_date"], "%Y/%m/%d")
                    value1=dic_temp["IB-Finish Date"]
                    value2=dic_temp["CB-Finish Date"]
                    value10=dic_temp["IB-Start Date"]
                    value20=dic_temp["CB-Start Date"]

                    df.at[i,'BL Finish Date'] = value1 if row['Reporting Date'] < given_value1
else value2
                    df.at[i,'BL Start Date'] = value10 if row['Reporting Date'] < given_value1
else value20

                except KeyError:
                    df.at[i,'BL Finish Date'] = dic_temp["IB-Finish Date"]
                    df['BL Finish Date'] = pd.to_datetime(df['BL Finish Date'],
format='%Y/%m/%d')
                    df.at[i,'BL Start Date'] = dic_temp["IB-Start Date"]
                    df['BL Start Date'] = pd.to_datetime(df['BL Start Date'],
format='%Y/%m/%d')

            elif row['Status'] == 'Non Started':
```

```
                try:
                    df.at[i,'BL Finish Date'] = dic_temp["CB-Finish Date"]
                    df.at[i,'BL Start Date'] = dic_temp["CB-Start Date"]
                except KeyError:
                    df.at[i,'BL Finish Date'] = dic_temp["IB-Finish Date"]
                    df.at[i,'BL Start Date'] = dic_temp["IB-Start Date"]

            elif row['Status'] == 'Finished':
                try:
                    df.at[i,'BL Finish Date'] = dic_temp["CB-Finish Date"]
                    df.at[i,'BL Start Date'] = dic_temp["CB-Start Date"]
                except KeyError:
                    df.at[i,'BL Finish Date'] = dic_temp["IB-Finish Date"]
                    df.at[i,'BL Start Date'] = dic_temp["IB-Start Date"]

        df['BL Finish Date'] = pd.to_datetime(df['BL Finish Date'], format='%Y/%m/%d')
        df['BL Start Date'] = pd.to_datetime(df['BL Start Date'], format='%Y/%m/%d')

        ### Forecasted Finish Date
        df['Forecasted Finish Date'] = None
        for i, row in df.iterrows():
            if row['Status'] == 'In Progress':
                duration_days = Timedelta(days=row['Forecasted Duration'])
                try:
                    df.at[i,'Forecasted Finish Date'] = row['Start date'] + duration_days
                except KeyError:
                    df.at[i,'Forecasted Finish Date'] = row['BL Start Date'] + duration_days

            elif row['Status'] == 'Non Started':
                if row['Reporting Date'] <= row['BL Start Date']:
                    df.at[i, 'Forecasted Finish Date'] = row['BL Finish Date']
                else:
                    df.at[i, 'Forecasted Finish Date'] = row['Reporting Date'] +
pd.to_timedelta(row['BL Duration'], unit='D')

            elif row['Status'] == 'Finished':
                df.at[i, 'Forecasted Finish Date'] = row['Finish Date']

        df['Forecasted Finish Date'] = pd.to_datetime(df['Forecasted Finish Date'],
format='%Y/%m/%d')
        df['Duration Variance (days)'] = df['BL Duration'] - df['Forecasted Duration']
        df['Finish Date Variance']= df['Forecasted Finish Date'] - df['BL Finish Date']
        df['Finish Date Variance']=df['Finish Date Variance'].dt.days
        df.to_csv(f"database {package_name}.csv")
        df_for_tree = df
        df_for_tree = df_for_tree.drop(['Tracking Period', 'AD_to_date',
'Predicted_value',  'Finish Date','Forecasted Duration','Duration Variance (days)','BL
Duration',], axis=1)

        df_for_tree['BL Finish Date'] = df_for_tree['BL Finish Date'] - pd.Timedelta(days=1)
### Subtractring one day to match P6 dates.
        datetim_cols = ['Reporting Date', 'BL Finish Date', 'Forecasted Finish Date', 'BL
Start Date','Start date']
        for col in datetim_cols:
            df_for_tree[col] = df_for_tree[col].dt.date
        df_for_tree = df_for_tree[['Period', 'Reporting Date', 'Status', 'BL Start Date', 'BL
Finish Date', 'Start date','Forecasted Finish Date', 'Finish Date Variance' ]]
        df_for_tree.rename(columns={'Start date': 'Actual Start Date','Finish Date
Variance':'Schedule Deviation'}, inplace=True)
        tree = ttk.Treeview(data_chart_frame, height=7, style='Custom2.Treeview')
```

```
        tree['columns'] = list(df_for_tree.columns)
        tree.column("#0", width=0, stretch=tk.NO)
        for col in df_for_tree.columns:
            if col in ('Period'):
                tree.column(col, anchor=tk.CENTER, width=8)
            elif col in ('ID'):
                tree.column(col, anchor=tk.CENTER, width=14)
            elif col in ( 'WP Name', 'Status'):
                tree.column(col, anchor=tk.CENTER, width=35)
            else:
                tree.column(col, anchor=tk.CENTER, width=80)
        tree.heading("#0", text="", anchor=tk.CENTER)
        for col in df_for_tree.columns:
            tree.heading(col, text=col, anchor=tk.CENTER )

        # Insert data from DataFrame into Treeview
        for i, row in df_for_tree.iterrows():
            tree.insert("", tk.END, iid=i, values=list(row), tags=('myTag',))

        tree.tag_configure('highlight', background='yellow')
        children = tree.get_children()
        if children:
            last_child_id = children[-1]
            # Apply the 'highlight' tag to the last row
            tree.item(last_child_id, tags=('highlight',))

        tree.grid(row=2, columnspan=2, padx=10, pady=0, sticky="ew" )
        scrollbar = ttk.Scrollbar(data_chart_frame, orient="vertical", command=tree.yview)
        tree.configure(yscrollcommand=scrollbar.set)
        scrollbar.grid(row=2, column=2, padx=0, pady=0, sticky="sn" )

        tree.tag_configure('myTag', font=("Arial", 10))

        show_chart(proj, package_name, period_selected)

    def create_buttons(frame, project, period_number):
        work_packages = list(project_info[project]["Work Packages"].keys())
        for widget in frame.winfo_children():
            widget.destroy()
        for i, package_name in enumerate(work_packages):
            button = ttk.Button(frame, text=package_name,
                                command=lambda name=package_name, proj=project,
period=period_number: show_data_and_charts(proj, name, period),
                                style='Custom4.TButton')
            button.grid(row=0, column=i, pady=10, padx=15, sticky="ew")

    def update_packages_combobox(project_code):
        project_reports_info = project_info[project_code]["Project Reports Info"]
        formatted_list = [f"Reporting Period {k.split()[-1]}: Ending {v['Date']}" for k, v in
project_reports_info.items()]
        period_combobox['values'] = formatted_list
        period_combobox.set(formatted_list[-1] if formatted_list else '')

    def clear_widgets():
        for widget in buttons_frame.winfo_children():
            widget.destroy()
        try:
            for widget in data_chart_frame.winfo_children():
                widget.destroy()
        except tk.TclError:
            pass
```

```
    def on_project_select(event):
        selection = project_combobox.get()
        match = re.search(r"Project ([\w-]+):", selection)
        if match:
            project_code = match.group(1)
            update_packages_combobox(project_code)
            clear_widgets()


    project_list = [f"Project {code}: {list(name.values())[0]}" for code, name in
project_info.items()]
    project_combobox = ttk.Combobox(wp_prediction_frame, values=project_list, font=('Arial',
13), width=45, justify='left')
    project_combobox.grid(row=0, column=0, padx=(10,180), pady=10, sticky="w")
    project_combobox.set('Please select a project')
    project_combobox.bind('<<ComboboxSelected>>', on_project_select)
    period_combobox = ttk.Combobox(wp_prediction_frame, values=['Please select a period'],
font=('Arial', 13), width=35, justify='left')
    period_combobox.grid(row=0, column=2, padx=10, pady=10, sticky="e")
    period_combobox.set('Please select a period')
    def on_package_select(event):
        project_selection = project_combobox.get()
        period_selection = period_combobox.get()

        # Extract project code from selection
        project_match = re.search(r"Project ([\w-]+):", project_selection)
        period_match = re.search(r"Reporting Period (\d+):", period_selection)
        if project_match:
            project_code = project_match.group(1)
        if period_match:
            period_number = int(period_match.group(1))
            create_buttons(buttons_frame, project_code, period_number)
            getting_predictions(project_code, period_number)
            wp_slected_project = list(project_info[project_code]["Work Packages"].keys())
            first_package_show = wp_slected_project[0]
            show_data_and_charts(project_code, first_package_show, period_number)

    period_combobox.bind('<<ComboboxSelected>>', on_package_select)
```

8. The Overall Project Forecasting: Capturing information from P6 Schedule and development CPM and PDM methodologies computerized to consolidate individual work package predictions.

```
## PROJECT FORECASTING
def project_forecasting():
    for widget in forecasting_project_frame.winfo_children():
        widget.destroy()

    def browse_file():
        global file_path
        file_path = filedialog.askopenfilename()
            file_path_var.set(file_path)
    def update_packages_combobox(project):
        project_reports_info = project_info[project]["Project Reports Info"]
        formatted_list = [f"Reporting Period {k.split()[-1]}: Ending {v['Date']}" for k, v in
project_reports_info.items()]
        period_combobox_overall['values'] = formatted_list
        period_combobox_overall.set(formatted_list[-1] if formatted_list else '')
```

```
    def clear_widgets():
        try:
            for widget in forecasting_project_frame.winfo_children():
                widget.destroy()
        except tk.TclError:
            pass

    heading_frame = ttk.Labelframe(forecasting_project_frame, text='Project Duration
Forecasting', style='Custom2.TLabelframe')
    heading_frame.grid(row=0, column=0, padx=0, pady=0, sticky='nwes')
    primavera_frame = ttk.Labelframe(forecasting_project_frame, text='Primavera P6 data',
style='Custom7.TLabelframe')
    primavera_frame.grid(row=1, column=0, sticky="nsew", padx=0, pady=5)
    ttk.Label(primavera_frame, text='Select project schedule at work package level: ',
font=("Arial", 11, ), background='white', ).grid(row=0, column=0, padx=10, pady=0, sticky="w")

    project_list = [f"Project {code}: {list(name.values())[0]}" for code, name in
project_info.items()]
    project_combobox_overall = ttk.Combobox(heading_frame, values=project_list, font=('Arial',
13), width=45, justify='left')
    project_combobox_overall.grid(row=0, column=0, padx=(10,180), pady=10, sticky="w")
    project_combobox_overall.set('Please select a project')
    period_combobox_overall = ttk.Combobox(heading_frame, values=['Please select a period'],
font=('Arial', 13), width=35, justify='left')
    period_combobox_overall.grid(row=0, column=2, padx=10, pady=10, sticky="e")
    period_combobox_overall.set('Please select a period')
    def on_project_select_overall(event):
        selection = project_combobox_overall.get()
        match = re.search(r"Project ([\w-]+):", selection)
        if match:
            project_code = match.group(1)
            update_packages_combobox(project_code)

    project_combobox_overall.bind('<<ComboboxSelected>>', on_project_select_overall)

## 1. CALCULATION BY CPM
    def get_primavera_p6(project_code_0, period_number_0):
        global project_id_combobox_frame0, report_combobox_frame0, data
        ttk.Button(primavera_frame, text="Display Duration at Completion Forecasting",
command=lambda: duration_at_completion(project_code_0), style='Custom.TButton').grid(row=2,
column=0,  columnspan= 5, padx=10, pady=5, sticky='nwes')

        dict_predictions = project_info[project_code_0]["Project Reports Info"][f'Project
Report Number {period_number_0}']['Predictions']
        dfPred_WP = pd.DataFrame(list(dict_predictions.items()), columns=['Work_Package_Name',
'WP_Predictions'])
        WPnames = list(dict_predictions.keys())

        #### Parsing the XML file
        tree = Xet.parse(file_path)
        root = tree.getroot()
        ns = {'' : root.tag[1:root.tag.index('}')]}
        for i in root.findall('*/Activity', ns):
            col_01 = []
            for i in root.findall('*/Activity/WBSObjectId', ns):
                col_01.append(i.text)
            col_02 = []
            for i in root.findall('*/Activity/ObjectId', ns):
                col_02.append(i.text)
            col_07 = []
            for i in root.findall('*/Activity/PlannedDuration', ns):
```

```
            col_07.append(i.text)
        col_08 = []
        for i in root.findall('*/Activity/ActualDuration', ns):
            col_08.append(i.text)
        col_09 = []
        for i in root.findall('*/Activity/PlannedStartDate', ns):
            col_09.append(i.text)
        col_10 = []
        for i in root.findall('*/Activity/PlannedFinishDate', ns):
            col_10.append(i.text)
        col_11 = []
        for i in root.findall('*/Activity/ActualStartDate', ns):
            col_11.append(i.text)
        col_12 = []
        for i in root.findall('*/Activity/ActualFinishDate', ns):
            col_12.append(i.text)
        col_23 = []
        for i in root.findall('*/Activity/StartDate', ns):
            col_23.append(i.text)
        col_24 = []
        for i in root.findall('*/Activity/FinishDate', ns):
            col_24.append(i.text)

    rows_1=zip(col_01, col_02, col_03, col_04, col_05, col_06, col_07, col_08, col_09,
col_10, col_11, col_12, col_23, col_24)
    cols_1 = ["WBSObjectId", "Activity_ObjectId", "ActivityID", "Activity_Name", "Type",
"Physical_Percent_Complete", "Planned_duration", "Actual_duration", "Planned_Start_Date",
"Planned_finish_date", "Actual_start_date", "Actual_finish_date", "Start_date", "Finish_date"]
    df1 = pd.DataFrame(rows_1, columns = cols_1)

    for i in root.findall('*/Relationship', ns):
        col_13 = []
        for i in root.findall('*/Relationship/ObjectId', ns):
            col_13.append(i.text)
        col_14 = []
        for i in root.findall('*/Relationship/PredecessorActivityObjectId', ns):
            col_14.append(i.text)
        col_15 = []
        for i in root.findall('*/Relationship/SuccessorActivityObjectId', ns):
            col_15.append(i.text)
        col_16 = []
        for i in root.findall('*/Relationship/Type', ns):
            col_16.append(i.text)
        col_17 = []
        for i in root.findall('*/Relationship/Lag', ns):
            col_17.append(i.text)

    rows_2=zip(col_13, col_14, col_15, col_16, col_17)
    cols_2 = ["Relationship_ObjectId", "PredecessorActivityObjectId", "Activity_ObjectId",
"Type", "Lags"]
    df2 = pd.DataFrame(rows_2, columns = cols_2)
    ################################################################
    for i in root.findall('*/WBS', ns):

        col_19 = []
        for i in root.findall('*/WBS/Name', ns):
            col_19.append(i.text)
        col_20 = []
        for i in root.findall('*/WBS/ObjectId', ns):
            col_20.append(i.text)
```

```
        rows_3=zip(col_19, col_20)
        df2333 = pd.DataFrame(rows_3, columns = ["Work_Package_Name", "WBSObjectId"])
        df2333["Work_Package_Name"] = df2333["Work_Package_Name"].str.upper()
        df2333.to_csv('output3.csv')

        ### Joining Predictions to respective WBS codes
        df_incl_predictions = pd.merge(df2333, dfPred_WP, on ='Work_Package_Name', how
='left')

        ###Calculating Physical Percent Complete per Work Package, using the average of
Physical Percent Complete of their activities.
        df1['Physical_Percent_Complete'] = pd.to_numeric(df1['Physical_Percent_Complete'])
        df3 = df1.groupby('WBSObjectId')['Physical_Percent_Complete'].mean()
        df3 = pd.DataFrame(df3)
        ### Putting Physical Percent Completes (df3) to every WBSObjectID of df1,
        df4 = pd.merge(df1, df3, on ='WBSObjectId', how ='left')
        df4[["Planned_Start_Date", "Planned_finish_date", "Actual_start_date",
"Actual_finish_date", "Start_date", "Finish_date"]] = df4[["Planned_Start_Date",
"Planned_finish_date", "Actual_start_date","Actual_finish_date", "Start_date",
"Finish_date"]].apply(pd.to_datetime)
        df4['WBSObjectId'] = pd.to_numeric(df4['WBSObjectId'])
        df4['Activity_ObjectId'] = pd.to_numeric(df4['Activity_ObjectId'])
        ### Putting min and max activities of each Work Package depending on % progress.
        ### Actual dates stand for finished, Planned dates for non started and dates (alone),
in progress. (Primavera nomenclature)

        Start_date_of_WP = []
        Finish_date_of_WP = []
        Activity_ID_start_date = []
        Activity_ID_finish_date = []

        for i in range(len(df3.index)):
            df5= df4.groupby('WBSObjectId').get_group(int(df3.index[i]))
            if df5['Physical_Percent_Complete_y'].mean() == 1 :
                    Start_date_of_WP.append(df5['Actual_start_date'].min())
                    Finish_date_of_WP.append(df5['Actual_finish_date'].max())
                    if  'Start Milestone' in df5['Type'].values:
                        Activity_ID_start_date.append(df5.at[df5.loc[df5['Type'] == 'Start
Milestone'].index[0],'Activity_ObjectId'])
                    else:
                        Activity_ID_start_date.append(df5.at[df5['Actual_start_date'].idxmin()
,'Activity_ObjectId'])

                    if  'Finish Milestone' in df5['Type'].values:
                        Activity_ID_finish_date.append(df5.at[df5.loc[df5['Type'] == 'Finish
Milestone'].index[0],'Activity_ObjectId'])
                    else:
                        Activity_ID_finish_date.append(df5.at[df5['Actual_finish_date'].idxmax
(),'Activity_ObjectId'])

            elif df5['Physical_Percent_Complete_y'].mean() == 0 :
                    Start_date_of_WP.append(df5['Planned_Start_Date'].min())
                    Finish_date_of_WP.append(df5['Planned_finish_date'].max())

                    if  'Start Milestone' in df5['Type'].values:
                        Activity_ID_start_date.append(df5.at[df5.loc[df5['Type'] == 'Start
Milestone'].index[0],'Activity_ObjectId'])
                    else:
                        Activity_ID_start_date.append(df5.at[df5['Planned_Start_Date'].idxmin(
),'Activity_ObjectId'])
```

```
                    if  'Finish Milestone' in df5['Type'].values:
                        Activity_ID_finish_date.append(df5.at[df5.loc[df5['Type'] == 'Finish
Milestone'].index[0],'Activity_ObjectId'])
                    else:
                        Activity_ID_finish_date.append(df5.at[df5['Planned_finish_date'].idxma
x(),'Activity_ObjectId'])

            else:   ### In-Progress Work Packages
                    Start_date_of_WP.append(df5['Start_date'].min())
                    Finish_date_of_WP.append(df5['Finish_date'].max())

                    if  'Start Milestone' in df5['Type'].values:
                        Activity_ID_start_date.append(df5.at[df5.loc[df5['Type'] == 'Start
Milestone'].index[0],'Activity_ObjectId'])
                    else:
                        Activity_ID_start_date.append(df5.at[df5['Start_date'].idxmin(),'Activ
ity_ObjectId'])

                    if  'Finish Milestone' in df5['Type'].values:
                        Activity_ID_finish_date.append(df5.at[df5.loc[df5['Type'] == 'Finish
Milestone'].index[0],'Activity_ObjectId'])
                    else:
                        Activity_ID_finish_date.append(df5.at[df5['Finish_date'].idxmax(),'Act
ivity_ObjectId'])
        df3['Start_date_of_WP'] = Start_date_of_WP
        df3['Finish_date_of_WP'] = Finish_date_of_WP
        df3['Activity_ID_start_date'] = Activity_ID_start_date
        df3['Activity_ID_finish_date'] = Activity_ID_finish_date

        df3 = df3.reset_index()
        ### Adding Activity name and ActivityID (df3_2)
        df3_1 = df3.merge(df4[['Activity_ObjectId','ActivityID','Activity_Name']], how
='left', left_on ='Activity_ID_start_date', right_on ='Activity_ObjectId')
        df3_2 = df3_1.merge(df4[['Activity_ObjectId','ActivityID','Activity_Name']], how
='left', left_on ='Activity_ID_finish_date', right_on ='Activity_ObjectId')
        df3_2.drop(['Activity_ObjectId_x', 'Activity_ObjectId_y'], axis=1, inplace=True)

        ###Sorting WP Start Dates'
        df3_2[["Start_date_of_WP"]] = df3_2[["Start_date_of_WP"]].apply(pd.to_datetime)
        df3_2 = df3_2.set_index('Start_date_of_WP')
        df3_2 = df3_2.sort_index()
        df3_2 = df3_2.reset_index()
        df3_2.columns.tolist()
        list1 = df3_2.columns.tolist()[0:3]
        list2 = df3_2.columns.tolist()[3:11]
        reorder_list1=[list1[1], list1[2], list1[0] ]
        new_column_order = reorder_list1 + list2
        df3_2 = df3_2[new_column_order]
        ##### Calculating the S-S lags -----------------
        df_for_CPM = df3_2.copy()
        df_with_predictions = pd.merge(df_for_CPM, df_incl_predictions, on='WBSObjectId',
how='left')

        df_with_predictions['CurrentDurations_P6'] = (df_with_predictions['Finish_date_of_WP']
- df_with_predictions['Start_date_of_WP']).dt.days +1
        date_match_1 = re.search(r"Ending (\d{4}/\d{2}/\d{2})", period_combobox_overall.get())
        report_date = date_match_1.group(1)
        report_date_datetime = datetime.datetime.strptime(report_date, '%Y/%m/%d')
        df_with_predictions['ElapsedDays'] = (report_date_datetime -
df_with_predictions['Start_date_of_WP']).dt.days
        df_with_predictions['ElapsedDays'] = df_with_predictions['ElapsedDays'].clip(lower=0)
```

```
        df_with_predictions['Duration'] = np.where(
            df_with_predictions['WP_Predictions'] == 0,
            df_with_predictions['CurrentDurations_P6'],
            df_with_predictions['ElapsedDays'] + df_with_predictions['WP_Predictions'])
df_with_predictions['Start_date_of_WP'].dtype)

        FS_lag_CPM=[]

        if df_with_predictions.shape[0]>1:
            for i in range(df_with_predictions.shape[0]-1):
                vari1=df_with_predictions.loc[i,'Start_date_of_WP']
                vari2=df_with_predictions.loc[i+1,'Start_date_of_WP']
                vari3=df_with_predictions.loc[i,'Finish_date_of_WP']
                vari4=df_with_predictions.loc[i+1,'Finish_date_of_WP']

                FS_lag_CPM.append((vari3-vari2).days + 1)
        else:
            None
        FS_lag_CPM_array = np.array([timedelta(days=days) for days in FS_lag_CPM])
        FS_lag_CPM.append(0)  ### to equal shape of df_with_predictions.
        df_with_predictions['FS_lag_CPM'] = FS_lag_CPM
        df_with_predictions['WP_SS_lags'] = (df_with_predictions['Duration'] -
df_with_predictions['FS_lag_CPM'])

        def generate_ac_list(number):
            return [chr(ord('A') + i) for i in range((number*2)-1)]

        ac_list = generate_ac_list(len(WPnames))

        letters_alternate = [ac_list[i] for i in range(0, len(ac_list)-1, 2)]
        pr_list = [item for item in letters_alternate for _ in range(2)]
        pr_list.insert(0, '-')

        du_list =  [val for pair in zip(df_with_predictions['WP_SS_lags'],
df_with_predictions['FS_lag_CPM']) for val in pair]
        du_list = du_list[:-2]
        du_list.append(df_with_predictions.loc[df_with_predictions.shape[0] - 1, 'Duration'])
        # Including Predecessors and Successors:

        df_2 = df2[["PredecessorActivityObjectId", "Activity_ObjectId", "Type", 'Lags']]
        df_2[["PredecessorActivityObjectId","Activity_ObjectId"]] =
df_2[["PredecessorActivityObjectId","Activity_ObjectId"]].astype(int)

        df_1 = df_with_predictions[['Work_Package_Name','WBSObjectId',
'Activity_ID_start_date', 'Activity_ID_finish_date']]
        df_1[['WBSObjectId','Activity_ID_start_date','Activity_ID_finish_date']] =
df_1[['WBSObjectId','Activity_ID_start_date','Activity_ID_finish_date']].astype(int)
        df_1.loc[:, 'Successors'] = None
        df_1.loc[:, 'Predecessors'] = None
        df_1.loc[:, 'RelationshipsS'] = None
        df_1.loc[:, 'RelationshipsP'] = None
        df_1.loc[:, 'lagS'] = None
        df_1.loc[:, 'lagP'] = None

        for i, row in df_2.iterrows():
            if row['Type'] == 'Finish to Start':
                first_index = df_1[df_1['Activity_ID_finish_date'] ==
row['PredecessorActivityObjectId']].index[0]
                second_index = df_1[df_1['Activity_ID_start_date'] ==
row['Activity_ObjectId']].index[0]
```

```
                df_1.at[first_index,'Successors'] = df_1.at[second_index, 'WBSObjectId']
                df_1.at[first_index,'RelationshipsS'] = row['Type']
                df_1.at[first_index,'lagS'] = int(row['Lags'])/8
                df_1.at[second_index,'RelationshipsP'] = row['Type']
                df_1.at[second_index,'lagP'] = int(row['Lags'])/8
            elif row['Type'] == 'Start to Start':
                first_index = df_1[df_1['Activity_ID_start_date'] ==
row['PredecessorActivityObjectId']].index[0]
                second_index = df_1[df_1['Activity_ID_start_date'] ==
row['Activity_ObjectId']].index[0]
                df_1.at[first_index,'Successors'] = df_1.at[second_index, 'WBSObjectId']
                df_1.at[first_index,'RelationshipsS'] = row['Type']
                df_1.at[first_index,'lagS'] = int(row['Lags'])/8
                df_1.at[second_index,'RelationshipsP'] = row['Type']
                df_1.at[second_index,'lagP'] = int(row['Lags'])/8
            elif row['Type'] == 'Finish to Finish':
                first_index = df_1[df_1['Activity_ID_finish_date'] ==
row['PredecessorActivityObjectId']].index[0]
                second_index = df_1[df_1['Activity_ID_finish_date'] ==
row['Activity_ObjectId']].index[0]
                df_1.at[first_index,'Successors'] = df_1.at[second_index, 'WBSObjectId']
                df_1.at[first_index,'RelationshipsS'] = row['Type']
                df_1.at[first_index,'lagS'] = int(row['Lags'])/8
                df_1.at[second_index,'RelationshipsP'] = row['Type']
                df_1.at[second_index,'lagP'] = int(row['Lags'])/8
            elif row['Type'] == 'Start to Finish':
                first_index = df_1[df_1['Activity_ID_start_date'] ==
row['PredecessorActivityObjectId']].index[0]
                second_index = df_1[df_1['Activity_ID_finish_date'] ==
row['Activity_ObjectId']].index[0]
                df_1.at[first_index,'Successors'] = df_1.at[second_index, 'WBSObjectId']
                df_1.at[first_index,'RelationshipsS'] = row['Type']
                df_1.at[first_index,'lagS'] = int(row['Lags'])/8
                df_1.at[second_index,'RelationshipsP'] = row['Type']
                df_1.at[second_index,'lagP'] = int(row['Lags'])/8

        ## Making Predecessor column:
      for i, row in df_1.iterrows():
            if row['Successors'] is not None:
                index1= df_1[df_1['WBSObjectId'] == int(row['Successors'])].index[0]
                df_1.at[index1,'Predecessors'] = row['WBSObjectId']
        ## Getting WP codes:
        work_packages = project_info[project_code_0]["Work Packages"]
        work_packages_list = [{"Work_Package_Name": wp.upper(), "Work Package ID":
details["Work Package ID"]}
                      for wp, details in work_packages.items()]
        df_3 = pd.DataFrame(work_packages_list)

        df_1 = pd.merge(df_1, df_3, on='Work_Package_Name', how='left')
        ### Preparing for tree:
        WBSObjectId_to_code = df_1.set_index('WBSObjectId')['Work Package ID'].to_dict()
        df_1['Successors'] = df_1['Successors'].map(WBSObjectId_to_code)
        df_1['Predecessors'] = df_1['Predecessors'].map(WBSObjectId_to_code)

        df_1 = df_1.drop(['WBSObjectId', 'Activity_ID_start_date', 'Activity_ID_finish_date'],
axis=1)

        df_1 = df_1[['Work Package ID', 'Work_Package_Name', 'Predecessors', 'RelationshipsP',
'lagP', 'Successors','RelationshipsS','lagS' ]]
        df_1.rename(columns={'Work_Package_Name': 'Work Package Name',
```

```
                            'RelationshipsP': 'Predecessor Relationship Type', 'lagP':
'Predecessor Lag',
                            'RelationshipsS': 'Successor Relationship Type', 'lagS':
'Successor Lag'}, inplace=True)
        df_1['Work Package Name'] = df_1['Work Package Name'].str.title()

        df_1.fillna('---', inplace=True)
        ## Making the Tree
        tree = ttk.Treeview(primavera_frame, height=3, style='Custom2.Treeview')
        tree['columns'] = list(df_1.columns)

        tree.column("#0", width=0, stretch=tk.NO)
            if col in ('Work Package ID'):
                tree.column(col, anchor=tk.CENTER, width=80)
            elif col in ('Work Package Name'):
                tree.column(col, anchor=tk.CENTER, width=110)
            elif col in ('Predecessors'):
                tree.column(col, anchor=tk.CENTER, width=65)
            elif col in ('Predecessor Relationship Type'):
                tree.column(col, anchor=tk.CENTER, width=160)
            elif col in ('Predecessor Lag'):
                tree.column(col, anchor=tk.CENTER, width=90)
            elif col in ('Successors'):
                tree.column(col, anchor=tk.CENTER, width=60)
            elif col in ('Successor Relationship Type'):
                tree.column(col, anchor=tk.CENTER, width=160)
            elif col in ('Successor Lag'):
                tree.column(col, anchor=tk.CENTER, width=80)


        tree.heading("#0", text="", anchor=tk.CENTER)  # Invisible column for IDs
        for col in df_1.columns:
            tree.heading(col, text=col, anchor=tk.CENTER )

        tree.tag_configure('myTag', font=("Arial", 10))
        for i, row in df_1.iterrows():
            tree.insert("", tk.END, iid=i, values=list(row), tags=('myTag',))

        tree.grid(row=1, padx=10, columnspan=8, pady=10, sticky="ew" )
        scrollbar = ttk.Scrollbar(primavera_frame, orient="vertical", command=tree.yview)
        tree.configure(yscrollcommand=scrollbar.set)
        scrollbar.grid(row=1, column=8, padx=0, pady=0, sticky="sn" )

        Removed_WP_list=[]
        Number_times = []

        for i in range(df3_2.shape[0]):

            for j in range(df3_2.shape[0]):

                var1=df3_2.iloc[i,[2]].item()
                var2=df3_2.iloc[j,[2]].item()
                var3=df3_2.iloc[i,[3]].item()
                var4=df3_2.iloc[j,[3]].item()

                if (var1 > var2) and (var3 < var4) and (i != j):
                    Removed_WP_list.append(df3_2['WBSObjectId'][i])
                elif (var1 > var2) and (var3 == var4) and (i != j):
                    Removed_WP_list.append(df3_2['WBSObjectId'][i])
                elif (var1 == var2) and (var3 < var4) and (i != j):
                    Removed_WP_list.append(df3_2['WBSObjectId'][i])
```

```
                elif (var1 == var2) and (var3 == var4) and (i != j):
                    Removed_WP_list.append(df3_2['WBSObjectId'][i])
                    Number_times.append(df3_2['WBSObjectId'][i])

        df3_4 = pd.DataFrame(zip(list(pd.unique(Number_times))), columns = ['WBSObjectId'])

        df3_4 = df3_4.merge(df3_2, how ='left', left_on ='WBSObjectId', right_on
='WBSObjectId')

        df3_4 = df3_4.groupby('Start_date_of_WP')['WBSObjectId'].min()
        df3_4 = pd.DataFrame(zip(df3_4.values.tolist()), columns = ['WBSObjectId'])
        df3_4 = df3_4.merge(df3_2, how ='right', left_on ='WBSObjectId', right_on
='WBSObjectId')

        df_Critical_WP = pd.DataFrame(zip(list(set(df3_2['WBSObjectId'].tolist()) -
set(list(pd.unique(Removed_WP_list)))) + df3_4['WBSObjectId'].values.tolist()), columns =
['WBSObjectId'])
        df_Critical_WP = df_Critical_WP.merge(df3_2, how ='left', left_on ='WBSObjectId',
right_on ='WBSObjectId')

        ###Evaluating Critical WP identified by including repeated WP with same start and
finish dates
        Removed_WP_list_1=[]
        Number_times_1 = []
        for i in range(df_Critical_WP.shape[0]):
            for j in range(df_Critical_WP.shape[0]):
                if (df_Critical_WP.iloc[i,[2]].item() > df_Critical_WP.iloc[j,[2]].item()) and
(df_Critical_WP.iloc[i,[3]].item() < df_Critical_WP.iloc[j,[3]].item()) and (i != j):
                    Removed_WP_list_1.append(df_Critical_WP['WBSObjectId'][i])
                elif (df_Critical_WP.iloc[i,[2]].item() > df_Critical_WP.iloc[j,[2]].item())
and (df_Critical_WP.iloc[i,[3]].item() == df_Critical_WP.iloc[j,[3]].item()) and (i != j):
                    Removed_WP_list_1.append(df_Critical_WP['WBSObjectId'][i])
                elif (df_Critical_WP.iloc[i,[2]].item() == df_Critical_WP.iloc[j,[2]].item())
and (df_Critical_WP.iloc[i,[3]].item() < df_Critical_WP.iloc[j,[3]].item()) and (i != j):
                    Removed_WP_list_1.append(df_Critical_WP['WBSObjectId'][i])
                elif (df_Critical_WP.iloc[i,[2]].item() == df_Critical_WP.iloc[j,[2]].item())
and (df_Critical_WP.iloc[i,[3]].item() == df_Critical_WP.iloc[j,[3]].item()) and (i != j):
                    Removed_WP_list_1.append(df_Critical_WP['WBSObjectId'][i])
                    Number_times_1.append(df_Critical_WP['WBSObjectId'][i])

        df3_5 = pd.DataFrame(zip(list(pd.unique(Number_times_1))), columns = ['WBSObjectId'])
        df3_5 = df3_5.merge(df3_2, how ='right', left_on ='WBSObjectId', right_on
='WBSObjectId')
        df3_5 = df3_5.groupby('Start_date_of_WP')['WBSObjectId'].min()
        df3_5 = pd.DataFrame(zip(df3_5.values.tolist()), columns = ['WBSObjectId'])
        df3_5 = df3_5.merge(df3_2, how ='right', left_on ='WBSObjectId', right_on
='WBSObjectId')

        ### Ensambling Critical WP table removing repeated WP with same starts and finish
dates (if they are)
        new_df_Critical_WP = pd.DataFrame(zip(list(set(df_Critical_WP['WBSObjectId'].tolist())
- set(list(pd.unique(Removed_WP_list_1))))+ df3_5['WBSObjectId'].values.tolist()), columns =
['WBSObjectId'])
        new_df_Critical_WP = new_df_Critical_WP.merge(df3_2, how ='left', left_on
='WBSObjectId', right_on ='WBSObjectId')

        ### df_WP_ensambled:
        Physical_progress_list = []
        start_date_list = []
        finish_date_list = []
        actual_elapsed_duration = []
```

```
        for i, work_package in enumerate(list(project_info[project_code_0]["Work
Packages"].keys())):
            dict3_temp= project_info[project_code_0]["Work Packages"][work_package]
            dict4_temp= project_info[project_code_0]["Project Reports Info"][f'Project Report
Number {period_number_0}']['Work Packages'][work_package]
            if dict4_temp['Status'] == 'In Progress':
                Physical_progress_list.append(float(0.5))
                actual_elapsed_duration.append(float(dict4_temp["Actual Duration To Date
(days)"]))
                if dict4_temp['Start Date Actual']:
                    start_date_list.append(dict4_temp['Start Date Actual'])
                else:
                    try:
                        start_date_list.append(dict3_temp["CB-Start Date"])
                    except KeyError:
                        start_date_list.append(dict3_temp["IB-Start Date"])
                try:
                    finish_date_list.append(dict3_temp["CB-Finish Date"])
                except KeyError:
                    finish_date_list.append(dict3_temp["IB-Finish Date"])

            elif dict4_temp['Status'] == 'Non Started':
                Physical_progress_list.append(float(0))
                actual_elapsed_duration.append(float(0))
                try:
                    start_date_list.append(dict3_temp["CB-Start Date"])
                    finish_date_list.append(dict3_temp["CB-Finish Date"])
                except KeyError:
                    start_date_list.append(dict3_temp["IB-Start Date"])
                    finish_date_list.append(dict3_temp["IB-Finish Date"])

            elif dict4_temp['Status'] == 'Finished':
                Physical_progress_list.append(float(1))
                actual_elapsed_duration.append(float(0))
                start_date_list.append(dict4_temp['Start Date Actual'])
                finish_date_list.append(dict4_temp['Finish Date Actual'])

        #Finding WP codes associated to WP by names
        WBSObjectId_list= []
        for WPname in WPnames:
            matching_items = df_incl_predictions.loc[df_incl_predictions['Work_Package_Name']
== WPname, 'WBSObjectId'].values.tolist()
            WBSObjectId_list.extend(matching_items)

        Physical_df = pd.DataFrame({'WBSObjectId':WBSObjectId_list,
'Physical_progress':Physical_progress_list, 'Start Date':start_date_list, 'Finish Date':
finish_date_list, 'Actual Duration to Date': actual_elapsed_duration})
        new_df_Critical_WP = pd.merge(new_df_Critical_WP, Physical_df, on='WBSObjectId',
how='left' )

        new_physical_list = new_df_Critical_WP['Physical_progress'].tolist()
        new_starts_date_list = new_df_Critical_WP['Start Date'].tolist()
        new_finish_date_list = new_df_Critical_WP['Finish Date'].tolist()
        new_actual_duration_list = new_df_Critical_WP['Actual Duration to Date'].tolist()

        new_df_Critical_WP['Physical_Percent_Complete'] = new_physical_list
        new_df_Critical_WP['Start_date_of_WP'] = new_starts_date_list
        new_df_Critical_WP['Finish_date_of_WP'] = new_finish_date_list
        new_df_Critical_WP['Actual Duration to Date'] = new_actual_duration_list
```

```
        new_df_Critical_WP = new_df_Critical_WP.drop(['Physical_progress','Start Date',
'Finish Date'],axis=1)
        ##Converting to datetime:
        new_df_Critical_WP['Start_date_of_WP'] =
pd.to_datetime(new_df_Critical_WP['Start_date_of_WP'])
        new_df_Critical_WP['Finish_date_of_WP'] =
pd.to_datetime(new_df_Critical_WP['Finish_date_of_WP'])
        ## Converting to numeric:
        new_df_Critical_WP['Physical_Percent_Complete'] =
pd.to_numeric(new_df_Critical_WP['Physical_Percent_Complete'])

        #### Overlap - WP relationships: List of numbers representing overlaps among Critical
WP:
        Overlap_list=[]
        if new_df_Critical_WP.shape[0]>1:
            for i in range(new_df_Critical_WP.shape[0]-1):
                if (new_df_Critical_WP.iloc[i+1,[2]].item() <
new_df_Critical_WP.iloc[i,[2]].item()) and (new_df_Critical_WP.iloc[i+1,[3]].item() <
new_df_Critical_WP.iloc[i,[2]].item()):
                    Overlap_list.append(new_df_Critical_WP.iloc[i+1,[3]].item()-
new_df_Critical_WP.iloc[i,[2]].item())
                elif (new_df_Critical_WP.iloc[i+1,[2]].item() <
new_df_Critical_WP.iloc[i,[2]].item()) and (new_df_Critical_WP.iloc[i+1,[3]].item() >
new_df_Critical_WP.iloc[i,[2]].item()):
                    Overlap_list.append(new_df_Critical_WP.iloc[i+1,[3]].item()-
new_df_Critical_WP.iloc[i,[2]].item())
                elif (new_df_Critical_WP.iloc[i+1,[2]].item() >
new_df_Critical_WP.iloc[i,[2]].item()):
                    Overlap_list.append(new_df_Critical_WP.iloc[i+1,[2]].item()-
new_df_Critical_WP.iloc[i,[3]].item())
        else:
            None
        Overlap_arr=np.array(Overlap_list)
        total_days = [td.total_seconds()/ (24 * 60 * 60) for td in Overlap_arr]

        total_sum_in_days = np.sum(total_days)
```

9. Inserting independent Prediction Results

```
      new_df_Critical_WP=new_df_Critical_WP.merge(df_incl_predictions, how ='left', left_on
='WBSObjectId', right_on ='WBSObjectId')
        new_df_Critical_WP['Actual Duration to Date'].astype(int)
        new_df_Critical_WP['WP_Duration']=np.where(
            (new_df_Critical_WP['Physical_Percent_Complete']>0)&(new_df_Critical_WP['Physical_
Percent_Complete']<1),
            new_df_Critical_WP['Actual Duration to Date'] +
new_df_Critical_WP['WP_Predictions'],
            (new_df_Critical_WP['Finish_date_of_WP'] -
new_df_Critical_WP['Start_date_of_WP']).dt.days-1)
        data = pd.DataFrame({'ac': ac_list, 'pr': pr_list, 'du': du_list })
        data['du'] = data['du'].round(1)

'Precedence Diagramming Method PDM - CPM'
    def open_cpm():
        global overall_project_frame, photo
        cpm_window = tk.Toplevel(overall_project_frame, background='white')
        cpm_window.title("Project Critical Path")
        ttk.Label(cpm_window, text="Project Duration calculated by the Critical Path Method
(CPM)",font=("Arial", 12, 'bold'), background='white').grid(row=0, column=0, padx=10,
pady=(5,0), sticky='w')
```

```
        ttk.Label(cpm_window, text="This network is the result of transforming existent work
package relationships (with lags) into\nFinish - Start (FS) relationship (without lags). Thus,
this depicts the Critical Path (in red).",font=("Arial", 10, ),
background='white').grid(row=1, column=0, padx=10, sticky='nw')

        label = ttk.Label(cpm_window, image=photo)
        label.photo = photo
        label.grid(row=2, column=0, padx=10, sticky="nsew")

        ttk.Label(cpm_window, text='Note:\n1.The Precedence Diagraming Method (PDM) is
transformed to AON network, which does not contain lags. \n2.From the AON Network above:\n -
"A" and "B" denote the CONCRETE Work Package\n -"C" and "D" denote the EXCAVATION Work
Package\n -"E" represents the BACKFILL Work Package',font=("Arial", 10, ),
background='white').grid(row=3, column=0, padx=10, sticky='nw')


    def duration_at_completion(project_code_0):
        global overall_project_frame, photo, data

        overall_project_frame = ttk.Labelframe(forecasting_project_frame, text='Overall
Project', style='Custom7.TLabelframe')
        overall_project_frame.grid(row=2, column=0, padx=0, pady=5, sticky="nsew")

        ttk.Button(overall_project_frame, text="Show PDM-CPM calculation detail",
command=open_cpm, style='Custom.TButton').grid(row=0, column=0, padx=10, pady=0, sticky='e')

        for q in range(1, 2):
            start = []
            graph = []
            atts = []
            path = []
            new = []
            st = ""

            last = data.iloc[-1, 0]
            last = chr(ord(last)+1)
            for j in range(len(data)):
                for k in range(len(data.iloc[j, 1])):
                    if data.iloc[j, 1][k] != '-':
                        new.append(data.iloc[j, 1][k])
            for j in range(len(data)):
                if not data.iloc[j, 0] in new:
                    st = st+data.iloc[j, 0]
            if data.shape[1] == 3:
                df = pd.DataFrame([[last, st, 0]], columns=["ac", "pr", "du"])
            else:
                df = pd.DataFrame([[last, st, 0, 0, 0]], columns=["ac", "pr", "b", "m", "a"])
            data = data.append(df)
            for i in range(len(data)):
                graph.append([])
                atts.append({})
            for j in range(len(data)):
                atts[j]["Name"] = data.iloc[j, 0]
                if data.shape[1] == 3:
                    atts[j]["DU"] = data.iloc[j, 2]
                else:
                    atts[j]["DU"] = (data.iloc[j, 4] + 4 *
                                     data.iloc[j, 3] + data.iloc[j, 2]) / 6
                if(data.iloc[j, 1] == "-"):
                    start.append(ord(data.iloc[j, 0])-65)
                    continue
```

```
                for k in range(len(data.iloc[j, 1])):
                    graph[ord(data.iloc[j, 1][k]) -
                        65].append(ord(data.iloc[j, 0])-65)

            level = [None] * (len(graph))
            def BFS(s, graph):
                visited = [False] * (len(graph))
                queue = []
                for i in s:
                    queue.append(i)
                    level[i] = 0
                    visited[i] = True
                while queue:
                    s = queue.pop(0)
                    path.append(s)
                    for i in graph[s]:
                        if visited[i] == False:
                            queue.append(i)
                            level[i] = level[s] + 1
                            visited[i] = True
                        else:
                            level[i] = max(level[s]+1, level[i])
            BFS(start, graph)

            levels = [None] * len(path)
            for i in range(len(path)):
                levels[i] = level[path[i]]
            path = [x for y, x in sorted(zip(levels, path))]

            for i in path:
            for s in path:
                if(data.iloc[s, 1] == "-"):
                    atts[s]["ES"] = 0
                else:
                    ls = []
                    for k in range(len(data.iloc[s, 1])):
                        ls.append(atts[ord(data.iloc[s, 1][k]) - 65]["EF"])
                    atts[s]["ES"] = max(ls)
                atts[s]["EF"] = atts[s]["DU"] + atts[s]["ES"]
            for i in range(len(graph)):
                if(graph[i] == []):
                    atts[i]["LF"] = atts[i]["EF"]
                    atts[i]["LS"] = atts[i]["ES"]
            path.reverse()
            for i in path:
                if(data.iloc[i, 1] != "-"):
                    for k in range(len(data.iloc[i, 1])):
                        if "LF" in atts[ord(data.iloc[i, 1][k]) - 65].keys():
                            atts[ord(data.iloc[i, 1][k]) - 65]["LF"] = min(atts[i]
                                                                           ["LS"],
atts[ord(data.iloc[i, 1][k]) - 65]["LF"])
                        else:
                            atts[ord(data.iloc[i, 1][k]) -
                                65]["LF"] = atts[i]["LS"]
                        atts[ord(data.iloc[i, 1][k]) - 65]["LS"] = atts[ord(data.iloc[i,
1][k]) - 65]["LF"] - atts[ord(data.iloc[i, 1][k]) - 65]["DU"]
                atts[i]["SK"] = atts[i]["LF"] - atts[i]["EF"]
            atts[-1]["Name"] = "End"
            for j in range(len(graph)):

            G2 = nx.DiGraph()
```

```
            for i in range(len(graph)):
                for j in graph[i]:
                    G2.add_edge(atts[i]["Name"], atts[j]["Name"])
            temp = []
            for i in range(len(atts)):
                temp.append(atts[i]["Name"])
            temp = dict(zip(temp, atts))
            nx.set_node_attributes(G2, temp)
            fig, ax = plt.subplots(figsize=(15, 15))
            pos = nx.nx_agraph.graphviz_layout(G2, prog='dot')
            nx.draw(G2, pos=pos, ax=ax, with_labels=True, font_weight='bold')
            nx.draw_networkx_edges(G2, pos, edge_color='olive', width=1, arrowstyle='simple',
arrowsize=20, min_source_margin=25, min_target_margin=25)
            crt = []
            notcrt = []
            for j, i in temp.items():
                if(i["LF"] == i["EF"]):
                    crt.append(j)
                else:
                    notcrt.append(j)
            nx.draw_networkx_nodes(G2, pos, node_size=5000,
                                   node_color='red', ax=ax, nodelist=crt)
            nx.draw_networkx_nodes(G2, pos, node_size=2500,
                                   node_color='black', ax=ax, nodelist=notcrt)
            nx.draw_networkx_labels(G2, pos, ax=ax, font_weight="bold",
                                    font_color="white", font_size=26)

            def without(d, keys={"Name"}):
                return {x: d[x] for x in d if x not in keys}

            for node in G2.nodes:
                xy = pos[node]
                node_attr = G2.nodes[node]
                d = G2.nodes[node]
                d = without(d)
                text = '\n'.join(f'{k}: {round(v,0)}' for k, v in d.items())
                ax.annotate(text, xy=xy, xytext=(70, 5), textcoords="offset points",
fontsize=20, bbox=dict(boxstyle="round, pad=0.3", fc="lightgrey"),
arrowprops=dict(arrowstyle="wedge"))
            ax.axis('off')
            plt.savefig('fig'+str(q)+".png")

        image = Image.open("fig1.png")
        percentage=38
        width, height = image.size
        new_width = int(width * (percentage / 100))
        new_height = int(height * (percentage / 100))
        resized_image = image.resize((new_width, new_height))
        photo = ImageTk.PhotoImage(resized_image)
        # Display the Gantt chart
        # Define tasks, their start and end dates
        try:
            start_planned_date_total_project = project_info[project_code_0]["CB-Start Date"]
        except KeyError:
            start_planned_date_total_project = project_info[project_code_0]["IB-Start Date"]

        try:
            finish_planned_date_total_project = project_info[project_code_0]["CB-Finish Date"]
        except KeyError:
            finish_planned_date_total_project = project_info[project_code_0]["IB-Finish Date"]
```

```
        start_planned_date_total_project =
datetime.datetime.strptime(start_planned_date_total_project, '%Y/%m/%d')
        finish_planned_date_total_project =
datetime.datetime.strptime(finish_planned_date_total_project, '%Y/%m/%d')

        wp_dict = project_info[project_code_0]["Work Packages"]
        wp_list = list(wp_dict.keys())
        date_columns = []
        for package_name in wp_list:
            file_path1 = f"database {package_name}.csv"
            df_file = pd.read_csv(file_path1)
            df_file['Start date'] = pd.to_datetime(df_file['Start date'])
            date_columns.append(df_file['Start date'])
        all_dates = pd.concat(date_columns)
        actual_start_date = all_dates.min()

        date_match = re.search(r"Ending (\d{4}/\d{2}/\d{2})", period_combobox_overall.get())
        today_input = date_match.group(1)
        today = datetime.datetime.strptime(today_input, '%Y/%m/%d')
        finish_dates_forecasted = actual_start_date + timedelta(days=int(atts[-1]['ES']))
        tasks = ['Forecasted', 'Actual', 'Current BL' ]
        start_dates = [today, actual_start_date, start_planned_date_total_project ]
        end_dates = [finish_dates_forecasted, today, finish_planned_date_total_project ]
        start_dates_num = [mdates.date2num(date) for date in start_dates]
        end_dates_num = [mdates.date2num(date) for date in end_dates]
        # Create a Gantt chart
        fig, ax = plt.subplots(figsize=(10, 4))
        # Create horizontal bars for tasks
        for i, task in enumerate(tasks):
            if task in 'Current BL':
                color = '#EAEE1B'
            elif task in 'Forecasted':
                color = '#9F2CEA'
            else:
                color = '#271BE6'

            bar = ax.barh(task, width=end_dates_num[i] - start_dates_num[i],
left=start_dates_num[i],height=0.5, color=color, edgecolor='black', label=task)

            text_position = round(((start_dates_num[i] + end_dates_num[i]) / 2),0)
            text_date = mdates.num2date(text_position)
            ax.text(text_position, i, f'{end_dates_num[i] - start_dates_num[i]} days',
ha='center', va='center', color='black', fontsize=11, fontfamily='Arial', )

        end_date = mdates.num2date(end_dates_num[0])
        ax.annotate(f'Finish Date: {end_date.strftime("%Y-%m-%d")}', (end_dates_num[0], 0.5),
                    textcoords="offset points", xytext=(5,0),
                    ha='left', va='center', fontfamily='Arial', fontsize=12)

        ax.set_yticks(range(len(tasks)))
        ax.set_yticklabels(tasks, rotation=0, fontdict={'family': 'Arial', 'size': 12})

        date_range = np.arange(min(start_dates_num), max(end_dates_num) + 1, 14)
        ax.set_xticks(date_range)

        ax.xaxis_date()
        ax.set_xticklabels(ax.get_xticklabels(), fontdict={'fontname': 'Arial', 'size': 8})
        fig.autofmt_xdate()
        plt.xlim(min(start_dates) - timedelta(days=7), max(end_dates) + timedelta(days=15))
        # Calculate the difference in days between "Planned" and "Forecasted"
        difference_in_days = (end_dates[2] - end_dates[0]).days
```

```
        difference_in_days -= datetime.timedelta(days=1).days
        # Determine the y-coordinate for the arrow (lower finish date)
        arrow_y = 0 if end_dates[0] < end_dates[2] else 2

        # Add a horizontal double-headed arrow with the difference in days
        arrow_start_x = end_dates[0]
        arrow_end_x = end_dates[2]

        if end_dates[0] < end_dates[2]:
            text_x = end_dates[0] + (end_dates[2] - end_dates[0]) / 2
            text_y = -0.25
        else:
            text_x = end_dates[2] + (end_dates[0] - end_dates[2]) / 2
            text_y = 1.75

        ax.text(text_x, text_y, arrow_text,
                fontdict={'family': 'Arial', 'fontsize': 11, 'color': 'black'}, ha='center',
va='center',
                bbox=dict(facecolor='white', edgecolor='black', boxstyle='square, pad=0.3'))

        ax.annotate('', xy=(arrow_start_x, arrow_y),
                    xytext=(arrow_end_x, arrow_y),
                    arrowprops=dict(arrowstyle='<->', color='black', lw=1.5, ls='--'))
        ax.axvline(x=today, color='red', linestyle='--')
        ax.text(today, 0.5, 'Data Date', color='red', fontsize=10, ha='left', family='Arial')
        ax.legend(fontsize=11, loc='upper center', ncol=3, frameon=False, bbox_to_anchor=(0.5,
1.3), prop={'family': 'Arial', 'size': 10})

        # Customize date formatting on the x-axis
        ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%b-%d'))

        # Show the Gantt chart
        plt.grid(axis='x', zorder=0)
        plt.gca().set_axisbelow(True)
        plt.tight_layout()
        plt.show()

        chart_canvas = FigureCanvasTkAgg(fig, master=overall_project_frame)
        chart_canvas.get_tk_widget().grid(row=1, padx=0, pady=0, column=0, sticky='new' )

        ### RESULT BOX:
        # Create Label widgets for the text values in the second row and onwards
        text_values_2 = int(atts[-1]['ES'])
        try:
            text_values_3 = datetime.datetime.strptime(project_info[project_code_0]['CB-Start
Date'], "%Y/%m/%d") + timedelta(days=atts[-1]['ES'])
            text_values_3 = text_values_3.strftime("%Y/%m/%d")
        except KeyError:
            text_values_3 = datetime.datetime.strptime(project_info[project_code_0]['IB-Start
Date'], "%Y/%m/%d") + timedelta(days=atts[-1]['ES'])
            text_values_3 = text_values_3.strftime("%Y/%m/%d")

        try:
            text_values_4 = datetime.datetime.strptime(project_info[project_code_0]['CB-Finish
Date'], "%Y/%m/%d")
            text_values_4 = text_values_4.strftime("%Y/%m/%d")
        except KeyError:
            text_values_4 = datetime.datetime.strptime(project_info[project_code_0]['IB-Finish
Date'], "%Y/%m/%d")
            text_values_4 = text_values_4.strftime("%Y/%m/%d")
```

```
        text_values_5 = (datetime.datetime.strptime(text_values_4, "%Y/%m/%d") -
datetime.datetime.strptime(text_values_3, "%Y/%m/%d")).total_seconds() / (24 * 60 * 60)

        text_values = [text_values_2, text_values_3, text_values_4, int(text_values_5)-1]
        text_labels = ["Prediction (days)", "Predicted Finish Date", "Planned Finish Date",
"Deviation (days)"]

        tree1 = ttk.Treeview(overall_project_frame, height=1, style='Custom2.Treeview')
        tree1['columns'] = text_labels
        tree1.column("#0", width=0, stretch=tk.NO)
        for heading in text_labels:
            tree1.column(heading, width=120, anchor=tk.CENTER)

        tree1.heading("#0", text="", anchor=tk.CENTER)
        for heading in text_labels:
            tree1.heading(heading, text=heading, anchor=tk.CENTER)

        tree1.tag_configure('myTag', font=("Arial", 11))
        tree1.insert('', tk.END, values=text_values, tags=('myTag',))

        tree1.grid(row=2, column=0, padx=10, pady=0, sticky="ew" )

    file_path_var = tk.StringVar()
    file_path_entry = ttk.Entry(primavera_frame, textvariable=file_path_var, state='readonly',
width=50, font=('Arial', 11))
    file_path_entry.grid(row=0, column=1, columnspan=5, padx=10, pady=2, sticky='nesw')

    def get_primavera_p6_wrapper():
        # Fetch project code from the combobox string
        match = re.search(r"Project ([\w-]+):", project_combobox_overall.get())
        if match:
            project_code_overall = match.group(1)
        else:
            messagebox.showerror("Error", "Invalid project selection.")
            return

        # Fetch period number from the combobox string
        period_match = re.search(r"Reporting Period (\d+):", period_combobox_overall.get())
        if period_match:
            period_number_overall = int(period_match.group(1))
        else:
            messagebox.showerror("Error", "Invalid reporting period selection.")
            return

        # Now call the original function with the extracted values
        get_primavera_p6(project_code_overall, period_number_overall)

    ttk.Button(primavera_frame, text="Browse File", command=browse_file,
style='Custom.TButton').grid(row=0, column=6,  padx=10, pady=2, sticky='nwes')
    ttk.Button(primavera_frame, text="Get P6 Data...", command=get_primavera_p6_wrapper,
style='Custom.TButton').grid(row=0, column=7,  padx=10, pady=2, sticky='nwes')
```

10. Consolidation of Frames

```
frames_names = [project_hub_frame, progress_frame, forecasting_work_package_frame,
forecasting_project_frame]
frames_functions  = [project_hub, enter_WP_reports, work_package_forecasting,
project_forecasting ]
```

```
def show_frame(frame_name):
    for frame, functions in zip(frames_names, frames_functions):
        if frame == frame_name:
            frame.grid(row=0, column=2, padx=10, pady=10, sticky="nsew")
            functions()
        else:
            frame.grid_forget()

task_bar = ttk.Labelframe(root, text="Main Menu", padding=(1, 10), style='Custom.TLabelframe')
project_frame = ttk.Labelframe(task_bar, text='Projects Setup', style='Custom1.TLabelframe')
home_button = ttk.Button(project_frame, text="Home", command = lambda:
show_frame(project_hub_frame), style='Custom.TButton')
track_frame = ttk.Labelframe(task_bar, text='Deep Learning Forecasting Data',
style='Custom1.TLabelframe')
progress_button = ttk.Button(track_frame, text="Enter Tracking Data", command=lambda:
show_frame(progress_frame), style='Custom.TButton' )
forecasting_frame = ttk.Labelframe(task_bar, text='Deep Learning Forecasting',
style='Custom1.TLabelframe')
forecasting_WP_results_button = ttk.Button(forecasting_frame, text="Step 01: Work Package
Level", command=lambda: show_frame(forecasting_work_package_frame), style='Custom.TButton')
forecasting_project_results_button = ttk.Button(forecasting_frame, text="Step 02: Project
Level", command=lambda: show_frame(forecasting_project_frame), style='Custom.TButton')

home_button.configure(padding=(10, 10))
progress_button.configure(padding=(10, 10))
forecasting_WP_results_button.configure(padding=(10, 10))
forecasting_project_results_button.configure(padding=(10, 10))

task_bar.grid(row=0, column=0, padx=5, pady=2, sticky="news",)
project_frame.grid(row=0, column=0, padx=10, pady=(10,20), sticky="nwes")
home_button.grid(row=0, column=0, padx=10, pady=10, sticky="w")
track_frame.grid(row=1, column=0, padx=10, pady=(20,20), sticky="nwes")
progress_button.grid(row=1, column=0, padx=10, pady=10, sticky= 'w')
forecasting_frame.grid(row=2, column=0, padx=10, pady=(20,20), sticky="nwes")
forecasting_WP_results_button.grid(row=3, column=0, padx=10, pady=10, sticky= 'w')
forecasting_project_results_button.grid(row=4, column=0, padx=10, pady=10, sticky= 'w')

show_frame(project_hub_frame)
root.mainloop()
```