Advancing Regression Based Analytics for Steel Fabrication Productivity Modeling

by

Arash Mohsenijam

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

Doctor of Philosophy
in
Construction Engineering and Management

Department of Civil and Environmental Engineering
University of Alberta

ABSTRACT

Accounting for seven percent of the Gross Domestic Product (GDP), the construction industry is the fifth largest contributor to the Canadian economy (Statistics Canada 2019). Structural steel is one of the primary materials used in the construction industry for providing structural stability in residential and commercial buildings, as well as critical infrastructure and industrial facilities such as bridges, oil and gas pipe racks. With emerging economic diversification efforts in Canada, it is expected that the construction industry, and the utilization of steel products therein, will continue to grow over the coming years. To respond to increasing demand, the construction industry is relying more on off-site prefabrication to shorten project delivery time, reduce construction cost, and improve overall quality. Off-site fabrication shops provide a safer work environment for labourers, conducive to higher productivity, while also removing uncertainties and risks associated with site conditions and environmental factors.

Because the construction industry relies on labour-intensive activities, predicting labour productivity is critical to project estimating and production planning, as well as scheduling and assessing the costs of different design alternatives. In particular, with the fast-growing market needs for structural steel, there is an urgent call for data-driven productivity models to support the industry practice in project estimating, scheduling and control. Based on a literature review, this research has identified a need for an analytical methodology that is effective, scalable and data-driven to model and predict labour productivity. This model is needed to improve the current industry practice of relying on irreplaceable and experienced personnel for project estimating and planning.

Despite this pressing need, the construction industry faces several challenges when shifting to more data-driven productivity modelling systems. First, because of the complexity, variability, and uncertainty of construction conditions and activities, productivity-influencing factors cannot be exhaustively identified and quantified, making it practically impossible to account for every relevant detail. As a response to this challenge, prefabrication facilities isolate environmental factors and implement manufacturing-like methodologies that minimize productivity-influencing factors. Second, in order for practitioners to trust and apply developed productivity models, the generated models need to be transparent, easy to use and adaptable. Third, with limited resources available in the construction industry, implementing and maintaining data-driven models need to minimize overhead costs and take advantage of readily available information as much as practically possible to optimize data collection efforts. Therefore, a systematic, transparent, and quantitative approach to determine labour productivity, based on historical project data, is optimal to support project cost estimating, resource scheduling and productivity analysis. Furthermore, an innovative approach is highly desirable to account for sufficient project details describing product uniqueness, complexity, and uncertainty involved in steel fabrication processes.

This research proposes a new methodology that correlates labour productivity data with project design features. This methodology essentially utilizes efficient data-driven methods to capture implicit patterns in historical data and steel structure design details to produce labour productivity models. The novelty of the present research lies in its simple-to-understand and easy-to-implement analytical approach in selecting model input parameters and classifying steel fabrication projects based on work content and design features. The focus of this research is on applications of Multiple Linear Regression (MLR) and proposes enhanced methods to cater to application needs, first by selecting a proper set of input variables through a proposed method called Modified Stepwise

Regression (MSR), then by splitting the feature domain by Model Trees (MT) into different branches to predict non-linearity using piecewise linear models. Compared to other predictive methods, this approach would satisfy the construction industry application need for transparency and ease of use in modelling productivity, while maintaining minimal data collection efforts and achieving high prediction accuracy.

The contributions of this study include: (1) proposing an application framework based on Modified Stepwise Regression (MSR) for selecting relevant input variables and streamlining a predictive model without losing the model's predictive power; the MSR method leverage a simple but different method to select input variables while also verifying MLR underlying assumptions; (2) developing and validating a steel fabrication labour productivity model and identifying effects of work content factors; (3) developing an analytical methodology to generate a system of Multiple Linear Regression (MLR) equations by coupling the power of MSR and Model Tree (MT); (4) formalizing a quantitative approach to analyze the trade-off between model fit quality, prediction accuracy, and model complexity; and (5) providing an analytical means to elucidate productivity data structure and influencing factors by classifying the data and identifying significant variables for each class.

Although this research focusses on applying the proposed methodologies and framework to steel fabrication productivity modelling, the proposed data-driven methodologies and application framework can be implemented wherever there is a need for a transparent, accurate and generalized predictive model to quantify input-output relationships. Concrete slump and viaduct installation time-predictive models are just a few examples of the generic applicability of the methodologies proposed and demonstrated in this thesis.

This thesis is organized to first introduce the problem domain and discuss this research significance. The problem statement and research objective are then elaborated for the scope of the proposed research. Then the MSR methodology, as an essential step in selecting relevant variables for MLR-based prediction modelling is presented. Next, the MSR application in steel fabrication productivity analysis is investigated in depth based on a BIM dataset from a partner company in Alberta. In order to achieve higher prediction accuracy, higher model generalization, and more insight into the data structure, the integration of MSR and MT is attempted and formalized. This integration significantly improves the model's prediction accuracy and generalization ability while maintaining a straightforward and explainable model form.

PREFACE

This thesis is an original work by Arash Mohsenijam, organized in a paper-based format, and is based on the following research papers:

Chapter 2 of the thesis has been accepted for publication as part of a special issue and journal paper: (1) Arash Mohsenijam, Ming Lu (2016). "Achieving Sustainable Structural Steel Design by Estimating Fabrication Labour Cost Based on BIM Data." Procedia Engineering, 145, 654-661. (2) Arash Mohsenijam, Francis M. Siu, Ming Lu (2017). "Modified Stepwise Regression Approach to Streamlining Predictive Analytics for Construction Engineering Applications". Journal of Computing in Civil Engineering, 31(3), 04016066. Dr Ming Lu was involved in problem identification, concept formation, research verification. Dr. Francis M. Siu was then the Postdoctoral Fellow who provided general comment on paper structure and cross-validated calculations presented in the paper.

and manuscript composition. Chapter 3 of the thesis is in the press for publication as a journal paper: Arash Mohsenijam and Ming Lu. “Framework for Developing Labour-Hour Prediction Models from Project Design Features: Case Study in Structural Steel Fabrication", Canadian Journal of Civil Engineering, Accepted Feb 2019. Dr Ming Lu was involved in problem identification, concept formation, research verification.

Chapter 4 of the thesis has been submitted for publication as a journal paper and is under review: Arash Mohsenijam and Ming Lu. “Integrating Model Trees and Modified Stepwise Regression for

Modeling Productivity of Offsite Fabrication". Journal of Computing in Civil Engineering, Feb 2019. Dr Ming Lu was involved in problem identification, concept formation, research verification.

ACKNOWLEDGEMENT

The presented work was substantially funded by the Natural Sciences and Engineering Research Council (NSERC) of Canada through a Discovery Grant (RGPIN-2016-04687).

First, I would like to express my sincere gratitude to my advisor Prof. Ming Lu for his continuous support, patience, inspiration, and immense knowledge. Appreciations are sincerely extended to the examination committee members Dr. Simman Abourizk, Dr. Carlos Caldas, Dr. Carlos C. Noguezfor, Dr. Yuxiang Chen, and Dr. Zhigang Tian for providing their critiques to advance this research work.

I would especially like to thank my industrial mentors David Fritz, Dalip Parsad, Paul Zubik, Ross Fraser and Todd Collister for sharing their valuable knowledge and experience and helping me define a practical research problem worth solving.

Finally, I would like to thank my lovely wife, Nassim Sedaghat, for her understanding, kindness, love, and support throughout the years; without her, this work would not have been possible. Thanks and respect to my parents, Amir Nasser and Kianoush for setting me on the right path, teaching me the value of hard work, and giving me all the opportunities any child could ever ask for. I dedicate this work to my beloved wife and parents.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

TABLE OF ABBREVIATIONS

AIC	Akaike Information Criteria
AID	Automatic Interaction Detection
ANN	Artificial Neural Network
BE	Backward Elimination
BIC	Bayesian Information Criteria
BIM	Building Information Model
BLUE	Best Linear Unbiased Estimators
CART	Classification and Regression Tree
EC	Environment Conditions
FPE	Final Prediction Error
FS	Forward Selection
GDP	Gross Domestic Product
GLS	Generalized Least Squares
MLE	Maximum Likelihood Estimators
MLR	Multiple Linear Regression
MSR	Modified Stepwise Regression
MT	Model Tree
NLMLR	Non-Linear Multiple Linear Regression
OLS	Ordinary Least Square
RMSE	Root-Mean Square Error
PRESS	Predicted Residual Error Sum of Squares
RT	Regression Trees
SSE	Sum of squared error
UCI	University of California, Irvine
VIF	Variance Inflation Factor
WC	Work Content
WE	Work Environment
WLS	Weighted Least Square

CHAPTER 1 : INTRODUCTION

This chapter presents an introduction to steel fabrication, the research background, predictive analytics in construction engineering and management, problem statements, research objectives, research methodologies, and overall thesis organization.

1.1 INTRODUCTION TO STEEL FABRICATION

Structural steel is one of the primary materials used to provide structural stability in projects ranging from residential and commercial buildings to oil and gas pipe racks (Warrian 2010). In addition to being adaptable across construction disciplines, structural steel is also a highly recyclable material, which improves projects' sustainability throughout the project life cycle. Within the context of prefabrication, fabricating structural steel elements in the controlled environment of an offsite fabrication shop leads to a higher fabrication quality, as well as time and cost benefits (Liddy and Cross 2002). Ultimately, off-site prefabrication improves the efficiency of the fabrication process and erection of structural steel, allowing the associated tasks to be scheduled earlier in the construction schedule, and making the engineered materials available at the right time to feed in to other construction trades on site.

The fabrication shop environment, at first glance, resembles a manufacturing setting (Figure 1-1); however, steel fabrication is significantly different from other types of manufacturing which produce large quantities of identical products in an automated or semi-automated environment with less uncertainty and fewer changes (Song and AbouRizk 2003). In contrast, steel fabrication

features labour-intensive processes which are performed on bespoke project designs that experience frequent design and shop layout changes.

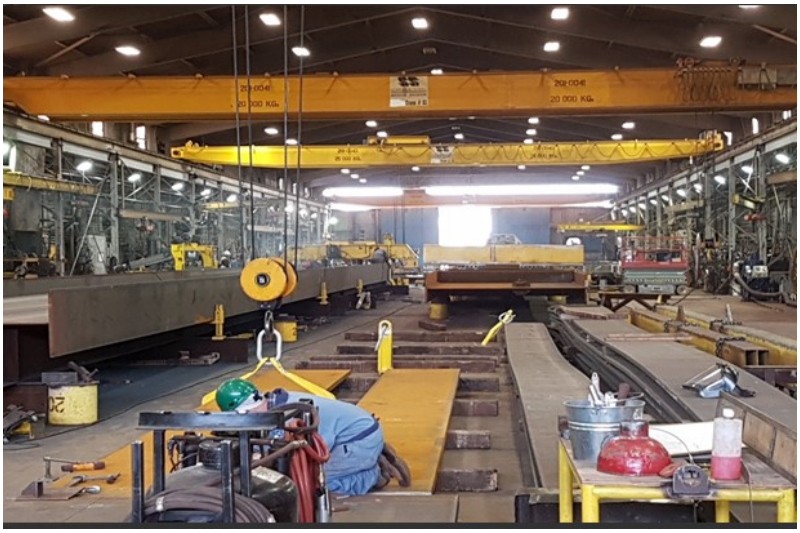

Figure 1-1: Steel girder fabrication with custom design and specifications

Another major difference between conventional manufacturing and steel fabrication lies in the client's role and relationship with the contractor. In conventional manufacturing, the customer and manufacturer relationship is built based on market needs and product advertisement, which usually takes place after production is complete. On the other hand, steel fabrication does not start without a request from the client. Typically, the first interaction between the project owner and steel fabricators occurs in the project bidding stage. As a result, a confident estimate of steel fabrication labour-hours is essential for assessing fabrication costs and planning production resources and processes (Dozzi and AbouRizk 1993). In the project bidding stage, the project is handed over to the steel fabricator's estimation team who must account for project details in order to justify a competitive price.

The structural steel price is the sum of materials, drafting and engineering, fabrication, and overhead costs, plus profit margins. The material costs are straight-forward as the purchasing department has exact quotations for different steel sections based on their weights. The raw material cost is then adjusted by the material wastage ratio (between 0-10%) which is affected by member length and size compared to what is available to purchase. In the estimating procedure, the commonly-applied unit of measure to account for direct cost is labour-hours required for each task or a project. The drafting and engineering expenses of a project are assigned through close collaboration between these departments and the estimators in order to adequately reflect project complexity and repetitiveness, and identify the labour-hours required to perform relevant tasks.

The fabrication of structural steel elements involves multiple specialized trades carrying out a series of operations such as handling, cutting, fitting, welding, and surface processing (e.g. sandblasting and painting). Therefore, in the estimating procedure, the labour-hours required to perform different operations are considered as the main unit to measure the direct cost. Thus, the estimators' knowledge of the engineering design and fabrication process is critical to transform design specifications into the associated labour-hour requirements for a certain project. On a construction task that is performed by manual labour, productivity is expressed as the labour production rate (labour-hours per installed unit), which measures a key dimension of project performance and is critical to estimating, scheduling and controlling the project (Alfeld, 1988). Owing to the labour-intensive nature of steel fabrication, the costs of tools, equipment, consumables, and shop managerial team are generally treated as indirect or overhead costs in practice, which are correlated with the direct cost in terms of labour-hours (Dozzi and AbouRizk 1993).

Despite advances made in information and construction technology in the past two decades, estimating methods in practice have remained much unchanged. In current practice, practitioners still rely on the rule of thumb for rough guesstimates, instead of utilizing data-driven, quantitative analytics. A systematic and quantitative approach to determining labour productivity based on readily available project data is much desired to support project cost estimating, resource scheduling and productivity performance tracking and improvement. Such an approach needs to account for sufficient project details such as product uniqueness in design, complexity, and uncertainty involved in steel fabrication processes. This research proposes and demonstrates integrated data-driven methods which support steel fabrication cost estimating. These proposed methods utilize innovative yet practical data-mining methods to capture the estimators' knowhow, represent steel structure design details and produce predictive models of productivity.

1.2 RESEARCH BACKGROUND

In this section, a review of related literature is presented to provide background for the thesis research, focussed on three main pillars: (I) labour productivity studies, (II) take-off project design features, and (III) linking the two aspects with predictive analytics.

1.2.1 Labour Productivity

"Productivity is a measure of the overall effectiveness of an operating system in utilizing labor, equipment and capital to convert labor efforts into useful output" (Hendrickson 2008). It is a widely accepted fact that steel fabrication is a complex process that features labour-intensive processes, custom-made pieces and activities involving a variety of equipment and labour disciplines (Song and Abourizk 2003). Unlike typical manufacturing, steel fabrication is labour-intensive, less automated, and undergoes frequent change orders and shop layout changes. These features make tracking the daily utilization of the workforce, and thus labour costs and productivity, a difficult task. These characteristics differentiate steel fabricators from other manufacturing companies, which produce identical products in high quantities and in an optimized environment (Song and AbouRizk 2003).

Labour productivity is critical information for managing projects and performing tasks such as estimating, scheduling, and project control (Song and Abourizk 2006). Having access to historical labour cost and productivity data is critical for cost estimating and shop production scheduling. In compiling the unit rates in a tender, estimators usually utilize a set of norms or standard productivity outputs to assess labour unit costs (Davison 2008). The industry productivity norms

are either obtained from published books [e.g. RS means (Gordian 2016)] or compiled by a steel fabricator based on historical records of past projects.

Productivity modelling has been extensively addressed in construction literature. Randolph et al. (1990) researched different definitions for productivity and proposed two methods to model productivity: a factor-based model and an expectancy model. Rifat and Rowings (1998) developed productivity models considering factors like temperature, quantity and crew size using neural networks and regression analysis. Dawood (1998) proposed Monte Carlo simulation to generate more reliable duration estimates, considering variations in quality of material, weather, and labour productivity. Knowles (1997) presented a feedforward backpropagation neural network model in predicting pipe-installation labour productivity. Building on Knowles work, Lu et. al. (2000) utilized probability inference neural network (PINN) models to predict pipe spool fabrication labour productivity using historical data. Song and AbouRizk (2008) used ANN and discrete-event simulation to analyze historical project data and develop labour productivity models. The neural network approach in productivity modelling is capable of mapping high dimensional input-output relationships; however, a productivity model should not be deemed as "black box" by keeping implicit critical information on factor selection and reasoning logic, which is crucial to produce reliable decision support in estimging productivity and potentially improving productivity.

There is an extensive published literature that provides unit rates and prices including labour-hour content for construction estimation. These publications range from price books for building and civil engineering works to books of norms for industrial and mechanical engineering works. However, such cost norms should not be utilized without rigorously comparing the circumstances of published data against those of the project in hand (Van Vilet 2011). In other words, the

estimators need to apply adjustment factors to the labour productivity norms on a per project basis. In current practice, adjustment factors are decided based on experience alongside rule-of-thumb guidelines (Lu 2001). Choosing adjustment factors relies on estimators' judgment, which might lead to biased and unreliable estimates. Unfortunately, this labour productivity information depends heavily on individual companies' fabrication shop performance and may differ from project to project and from time to time. For this reason, published steel fabrication productivity data only represent industry average values and are often neglected in practice. To arrive at more accurate labour productivity, more customized and systematic data collection is required.

To overcome this challenge, companies invest in custom productivity manuals, which entail substantial costs to establish and maintain. Therefore, they may not be updated regularly and only serve as a guideline for productivity estimating. Eventually, companies resort to relying on irreplaceable experienced personnel for project estimation and scheduling. In short, the absence of a scalable data-driven labour-hour estimating system has led to the current industry practice of relying on irreplaceable experienced personnel for project estimating and planning. It is noteworthy that relying heavily on individuals' experience has also been identified as one of the main causes behind inaccurate or insufficient estimates and project budget overruns on structural steel fabrication projects (Song and AbouRizk 2006).

A reliable source of productivity data already exists in companies' databases including project scope, progress information, and labour expenditures. Historical data serves as a basis for productivity models to form simple equations that predict productivity of future projects (Song and AbouRizk 2008). More specifically, these productivity models evaluate the effect of influencing factors on productivity using simple equations, nonlinear equations, or other advanced

model forms. Once created, these models can then be used as effective decision-support to predict productivity on future projects (Song and Abourizk 2008).

A review of recent literature has resulted in the following observations:

- There is an immediate need for a systematic approach to investigate companies' databases for historical project data that hold predictive productivity information. Available historical data from building information models and labour costing systems would serve as a basis for developing quantitative productivity models.
- Productivity data tracked in labour costing systems in practice are generally not accurate or sufficient to support any meaningful data-driven analysis.
- Limited research has been conducted on the quantitative analysis of the relationship between design inputs and productivity outputs. Therefore, limited contributions have been made to improve estimation accuracy and controlling engineering productivity by connecting to project design.
- In many instances, the fabrication cost is predicted based on the structural weight, which is an oversimplification. Structural weight cannot adequately predict the fabrication cost as it ignores project complexity and connection details.

1.2.2 Project features

The first step in cost estimating is an accurate quantity take-off. There are several common ways to perform quantity take-offs in construction projects. The conventional method is to count different materials from design drawings (i.e. 2D drawings) and transfer the obtained information

into spreadsheets (Hu X. et al. 2014). Manual quantity take-off methods are not able to detect inconsistencies between drawings, which would cause inaccurate estimates (Shen and Issa 2010).

To overcome the limitations of manual quantity take-offs, an enhanced approach is to extract quantities from Building Information Models (BIM) (Monteiro and Poças Martins 2013). BIM proposes significant advantages over previously used CAD systems by integrating the visual enhancements, parametric modelling, while providing a collaborative environment to designers and construction experts (Gu & London, 2010). Vast adoption of BIM by the construction industry has opened new opportunities in design, planning, and execution of construction projects. There are many studies addressing the quantity take-off and estimating capabilities of BIM (Monteiro and Poças Martins 2013; Plebankiewicz et al. 2015; Shen and Issa 2010; Xiaolin Hu et al. 2014). Shen and Issa (2010) studied BIM-based construction estimation and the impact of visualization on estimation accuracy. Monteiro and Pocas Martins (2013) demonstrated the possibility of extracting quantities from a BIM model in order to create a model ready for visualization or drawing applications. Plebankiewicz et al. (2015) investigated BIM-based cost estimating systems, concluding that BIM applications can generate accurate quantity takeoffs, but there is a knowledge gap in relating material quantities to labour costs. Hu et al. (Hu et al. 2014) used extracted data from a BIM model and generated a linear regression model with 56 available BIM features for estimating steel fabrication labour-hours. The reviewed literature indicates a great potential for capturing design features and quantities from BIM. That being said, the design features represented in BIM models of structural steel projects can be numerous; thus, having all of them factored in a model would not be a practical approach.

Moreover, the connection between the BIM features (along with the quantity takeoff for each) and required labour-hours is yet to be established. Also, a reliable source of productivity data can be found in most of the companies' databases, containing well-structured information related to such items as project scope, progress information, and labour expenses. High-quality historical data serves as a firm basis for building labour cost models in the form of simple equations that are instrumental in predicting the productivity of future projects (Song and Abourizk 2008).

Furthermore, the granularity of data as captured in the company's databases is usually limited by the contract requirements for filing billing documents or project earned value reports. In other words, existing productivity data is generally not tracked for creating detailed labour cost models at the operations level that represent workflows on the shop floor, namely how much time particular individual workers dedicate to specific jobs. For instance, limited research is conducted on the quantitative analysis of the design input and labour-hour output relationship due to the difficulty in acquiring well-structured design data (Song and AbouRizk 2006).

The first step of the proposed research methodology for developing data-driven labour cost or productivity estimating models is to thoroughly investigate the current practice and systems. This investigation facilitates an understanding of the practical level of granularity in the historical project data available. Although historical data may be imperfect or incomplete, devising powerful analytics to make the most value of such data is significant when developing decision-support models in the realistic setting of construction engineering and management.

1.3 PREDICTIVE ANALYTICS IN CONSTRUCTION ENGINEERING MANAGEMENT

Predictive analytics has played an essential role in making critical decisions in construction engineering and project management scenarios ranging from project cost and time estimation to cost control and analysis. This section provides comprehensive literature reviews on commonly accepted predictive methodologies in the construction engineering and management domain, including: (1) Artificial Neural Networks, (2) Operations Simulation, and (3) Regression Analysis.

1.3.1 Artificial neural networks

Artificial Neural Networks (ANN) are a computational method inspired by the brain's biological neural networks. This technique can be used to create predictive models by simulating complex input-output relationships. The ANN network is developed using a heuristic process (Graham et al. 2006) which is comprised of interconnected nodes arranged in different layers, where the node connections are characterized as numeric weights that are refined during the training stage. In general, the network structure can be subjectively formulated (e.g. the number of layers and types of transformation function) (El-Sawy et al. 2011). There are numerous variations of ANN; however, ANN in this thesis focuses on Back Propagation Neural Networks (BPNN) which are the most widely applied variation in engineering applications.

The suitability of the network configurations needs to be examined prior to drawing any statistical predictions (Heravi and Eslamdoost 2015). Increasing the number of nodes and hidden layers of an ANN model can reduce the errors between the predicted outputs and the actual outputs based on the training set. However, an ANN with a more elaborated structure is computationally intensive and more likely to produce an over-fitted model (Demuth et al. 2009). As such, some

researchers have suggested appropriate structures for ANN to eliminate under-fitting and over-fitting problems based upon trial-and-error approach and optimization techniques (Kim et al. 2004; Zhou 2010), but the analytical solutions are case-dependent and cannot be universally applied.

Implementations of ANN in construction-related predictive models can be traced back to the early 1980s (El-Sawy et al. 2011). Representative ANN applications relevant to the current research include: estimating the labour productivity of spool fabrication (Lu et al. 2000); modelling a ready-mixed concrete delivery system to predict operation duration, rate of delivery, and utilization rates of concrete plants (Graham et al. 2006); estimating steel fabrication productivity (Song and AbouRizk 2008); predicting the overhead cost of building projects (El-Sawy et al. 2011); forecasting construction labour productivity (Heravi and Eslamdoost 2015); and estimating the construction cost of executing multiple projects (Hyari et al. 2016).

From a practical point of view, the ANN methodology is mainly used as a "black box" by observing the predicted outputs based on the inputs. Approaches such as sensitivity analysis have been proposed to gain insight into the input and output relationships (Lu et al. 2001); while, analytical mechanism still relies on complex neural computing algorithms and built-in heuristics for Monte Carlo simulation. The derived results depend on the "trial and error" process in neural network training and interpretation of the results is not as straightford and explicit as desired. Nonetheless, to make it acceptable and to lend effecitive decision support in the intended application setting in the real world, it is indispensable to provide transparent reseaning logic of an AI model in terms of what role each input parameter plays in deriving the predicted output and how the output is related to the inpout factors in addition to making a point-value prediction.

1.3.2 Operations Simulation

Operations simulation is another strategy to support project estimation and predictions by modelling the construction process and representing activity and resource interactions and practical constraints (Song and AbouRizk 2008). Construction simulation is capable of capturing the uncertainties of activity execution by modelling the activity duration with probability distributions. Multiple simulation runs can be conducted by performing "what-if" scenarios for prediction purposes, for example, predicting project duration with a particular confidence level.

Operations simulation has been conducted across various construction disciplines, with notable studies that span the last two decades. For example, Marzouk and Moselhi (2002) simulated equipment workflows to achieve minimum project cost. Nguyen et al. (2013) simulated the construction of multistory buildings to predict project durations. Akhavian and Behzadan (2013), Alshibani and Moselhi (2012), and Marzouk and Moselhi (2003) simulated earthmoving operations for balancing fleets. These simulation applications can be limited to the in-depth knowledge and practical experience required to transform real-world operations into the representation of a simulation model.

Despite the prevalence of simulation models, there are concerns regarding their sufficiency in making statistically sound decisions. The predictions' accuracy depends on the validity of the simulation model (Nguyen et al. 2013); however, updating the simulation model is a demanding task if the planned operation sequence changes or new data is fed to the model. As a result, redefining fitted distribution and rebuilding the simulation model may be required (Gozalez-Quevedo et al. 1993; Panas and Pantouvakis 2014).

1.3.3 Regression Analysis

Multiple Linear Regression (MLR), as represented by Equation (1-1), has demonstrated usefulness and practicality in construction (El-abbasy et al. 2014; Jafarzadeh et al. 2014, 2015; Silva et al. 2013). MLR formulates a linear equation by relating two or more independent variables with the dependent variable for estimation and prediction. The literature review demonstrates a widespread use of MLR in the construction field, which can be summarized as follows: Smith (1999) investigated earthmoving productivity in association with operating conditions. Lowe et al. (2006) estimated the project cost of constructing buildings. Choi et al. (2013) identified the factors affecting pavement performance. El-abbasy et al. (2014) predicted the conditions of oil and gas pipelines. Jafarzadeh et al. (2015) predicted seismic retrofit cost.

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \varepsilon \quad (1\text{-}1)$$

Where α = intercept, β_i = regression coefficients; ε = error; X_i = independent variables; Y = dependent variable.

MLR is commonly accepted as a transparent predictive method; however, several application limitations need to be addressed. First, the underlying assumption for MLR needs to be verified, so the resulting model is not biased and quantitatively reliable. Second, the input variables chosen for the model need to be analytically justified. Third, MLR assumes a linear relationship between inputs and outputs, which could be a good approximation but would fail if a significant level of nonlinearity exists in the data. The first two limitations are addressed in the second chapter of this thesis by proposing the Modified Stepwise Regression (MSR) approach. MSR analytically selects a proper set of input variables by analyzing the variables' significance and descriptive power, while maintaining validity and MLR assumptions. The third limitation of MLR models is addressed in

Chapter 4 by splitting the complex problem into branches by applying Model Trees and tackling nonlinearity with multiple piecewise linear regressions. This proposed approach leads to the realization of transparency, simplicity and accuracy in productivity predictive modelling. Moreover, the proposed methodologies in this research would elucidate the structure, patterns and relationship between input and output variables. Those functionationalities are generally not provided in the mainstream AI methods at present.

1.4 PROBLEM STATEMENT

The current practice of steel fabrication relies on manual quantity take-offs, outdated productivity data, and experience-based contingency and uncertainty markups (Davison 2008). Consequences of the current practice are time-consuming and error-prone quantity take-offs, rough estimates based on outdated productivity data, and biased uncertainty related markups that do not represent project complexity.

Extensive literature review and communication with industry partners has revealed the need and desire for a quantitative approach that determines labour productivity based on project design features. This quantitive approach is needed to support project cost estimating, resource scheduling and productivity improvement; and must be simple but effective enough to account for sufficient project details such as product uniqueness, complexity, and uncertainty involved in steel fabrication processes. Despite decades of research and technological advancement, construction practice has generally remained the same. The movement towards data-driven productivity modelling has failed to cause any significant change in the construction industry as a result of the following:

(1) Due to the complexity, variability, and uncertainty of construction projects, many productivity-influencing factors are required to be collected and analyzed, making it practically impossible to account for every relevant detail.
(2) Transparency, accuracy, and simplicity of a productivity model are crucial factors in influencing the extent to which practitioners would trust and use it. Also, the fast-paced

nature of the construction industry, and the increasing amount of data collected over time, requires a method that can be updated with ease.

(3) With the constant pressure to control project cost, a proposed data-driven method should utilize readily available information as much as practically possible, instead of incurring extra overhead costs in order to collect data.

Prefabrication facilities have the advantages of isolating environmental factors and implementing standardized processes and methods that reduce the number of productivity-influencing factors. Prefabrication has created a unique situation for productivity modelling and analysis, where many environmental factors can be isolated, thus significantly increasing the chances of success in data-driven productivity modelling.

There is a need for a systematic, transparent, and quantitative approach in order to determine labour productivity based on project historical data and support project cost estimating, resource scheduling and productivity analysis. In the case of steel fabrication processes, an innovative approach is highly desired that can account for sufficient project details such as product uniqueness, complexity, and uncertainty.

1.5 RESEARCH OBJECTIVE

The primary goal of this research is to correlate project content with labour productivity in the off-site prefabrication of structural steel by developing productivity models that take advantage of existing historical data. The novelty lies in its unsupervised yet transparent approach in grouping similar observations, selecting proper input features for each group, and performing regression analysis on each group, while correlating work content of projects with labour requirements. The following sub-objectives are set in the context of achieving the primary objective of this research:

- Investigating practically available historical data in industry practice and setting an achievable level of granularity for data collection.
- Analytically selecting a proper set of design features that can be relevant to steel fabrication labour-hours, and which result in the leanest form of MLR while maintaining prediction accuracy.
- Proposing a framework to develop productivity models and identify the effects of work content factors.
- Developing range estimates around productivity model point estimates to account for prediction uncertainty.
- Analytically grouping projects based on design features and exploring splitting data prior to variable selection. Developing an analytical methodology for generating a system of MLR equations utilizing tree-based splitting algorithms coupled with variable selection methods.
- Performing bias-variance-complexity trade-off between predictive models in model selection.

Results and methodologies derived from this research would directly influence current practice for estimators, project managers, and shop production planners; as they would have the ability to configure labour resources, schedule shop floor production, and recognize any associated errors in estimates based on historical company data.

It is anticipated that the methodologies and framework proposed by this research would apply to construction-related problems beyond steel fabrication productivity modelling. Wherever there is a need to develop a transparent, accurate and generalized predictive model to quantify input-output relationships, the proposed methods can be leveraged. Concrete slump and viaduct installation time predictive models are just a few examples of the generic application of proposed methodologies in this thesis.

1.6 THESIS ORGANIZATION

This thesis consists of the following five chapters:

Chapter 1 provides a brief overview of the research background from practical and academic perspectives. A short literature review is provided to support the need for this research. The problem statement and research objective are later discussed in this Chapter, followed by proposed methodologies to adequately address them.

Chapter 2 proposes a method of variable selection in predictive models. In this Chapter, an analytical framework is proposed for developing MLR-based predictive models by (1) selecting input variables based on a modified stepwise approach, (2) verifying the MLR underlying assumptions, and (3) validating the prediction performance of the regression model. The resulting MLR model only contains the most relevant input variables while also fulfilling the Best Linear Unbiased Estimators (BLUE) assumptions. By utilizing statistical inference techniques, the MLR model also produces reliable range estimates around its point-value prediction according to a particular confidence level. To illustrate the application procedure of the proposed framework, a dataset intended for workability control of ready-mixed concrete from the University of California, Irvine (UCI) machine learning repository is used. A practical case study based on a real-world bridge construction project is provided to further demonstrate the application of the proposed methodology in modelling the precast span installation cycle-time.

Chapter 3 proposes a data-driven approach that uses MLR and available historical data from Building Information Models (BIM) to associate project labour-hours and project design features.

The framework relies on an enhanced version of the stepwise regression technique to select the most relevant predictive factors and generate a predictive model without compromising the achievable accuracy of regression. The framework also encompasses analytical methods for justifying MLR application, validating the resulting model, and establishing range estimates for point-value predictions. In collaboration with an industry partner, the framework application is exemplified by analyzing labour-hours and design features for structural steel fabrication, leading to the creation of a valid MLR model in the simplest form. Finally, the pros and cons of the proposed framework and opportunities for future research are discussed.

Chapter 4 explores the relationships between engineering design features and fabrication productivity in an off-site facility and utilizes existing historical data in the development of predictive productivity models. End users of predictive productivity models in construction demand an understanding of factor selection and reasoning logic of the built model, more so than achieving marginal gains on model prediction accuracy. The novelty of the present research lies in its unsupervised approach in classifying projects based on work content and design features. The contributions of this study include: (1) proposing a framework to develop productivity models and identify effects of work content factors, (2) developing an analytical methodology for generating a system of Multiple Linear Regression equations by coupling the power of Modified Stepwise Regression (MSR) and Model Tree (MT), and (3) analyzing the trade-off between model fit quality, prediction accuracy, and model complexity. The performance of the proposed methodology is benchmarked on the concrete slump dataset from the University of California Irvine machine learning repository. In the steel fabrication productivity modelling application, variables selected for splitting in Model Tree and variables selected for regression modelling are well aligned with industry practitioners' know-how. This degree of transparency in reasoning logic

is generally impossible to attain for non-linear regression models such as Artificial Neural Networks. As a result, compared to commonly-applied linear or nonlinear regression models, the resulting productivity predictive model achieves higher fit quality and generalization ability when predicting unseen cases.

Chapter 5 restates the research contributions and conclusions of this research and proposes future research ideas to further advance productivity modelling.

Appendix (I) provides all of the MATLAB source codes used in this research for the analysis of data.

Appendix (II) demonstrates a table of sample data points and a link to a public repository to access the primary dataset used in this research.

Chapter 1 of this thesis introduces, elaborates, and validates the Modified Stepwise Regression using two small application cases. Later in Chapter 2, the Modified Stepwise Regression is applied to a practical problem of steel fabrication productivity modelling and provides support for generalization of the methodology. Chapter 3 proposes to improve Modified Stepwise Regression by conducting a piecewise breakdown of the problem feature space using Model Trees; M5 tree combined with Modified Stepwise Regression shows considerable improvement over other linear regression methods while maintaining simplicity and transparency. The dataset analyzed in this study is available at https://figshare.com/s/8de57c3a0ca8f8ed37c4.

CHAPTER 2 : MODIFIED STEPWISE REGRESSION

This chapter elaborates on the Modified Stepwise Regression methodology by first reviewing its background and comparable methods, the stepwise approach in variable selection. Later in this chapter, the underlying assumptions of the developed model are verified and validated. A method for Error Estimation is also discussed in this chapter. In this Chapter, the proposed framework is validated using two cases of concrete slump and viaduct datasets.

2.1 INTRODUCTION

Predictive analytics provides essential quantitative decision-support in construction engineering and project management from material preparation to method design and productivity estimating. For instance, in order to achieve acceptable quality and productivity in concreting construction in the field, it is imperative to have a reliable prediction of the workability of concrete (e.g. the slump measure) in addition to the compressive strength of a given concrete mix design; for achieving cost efficiency and productivity in the construction of the viaduct made of the precast segments, it is crucial to have an accurate estimate of precast span installation cycle-time. Making critical decisions in connection with complicated construction engineering problems is primarily based on experiences, supported by meticulous analyses of data and facts available, particularly when manual calculation methods are used (El-Sawy et al. 2011). Estimates made entirely based on estimators' experiences may lead to unreliable results (Song and AbouRizk 2008). Regression provides an analytical method which models complicated real-world systems and predicts their behaviour using a mathematical equation. (Barrett and Gray 1994; Lewis-Beck 1978; Smith 1999).

The regression analysis is mainly used to create predictive or explanatory models, in an attempt to extract knowledge from collected data (Hair et al. 2010; King 1986). Due to ease of use and the flexibility to model a variety of application problems, regression analysis is a widely accepted quantitative technique for performing time and cost estimates in construction engineering (El-abbasy et al. 2014; Jafarzadeh et al. 2014, 2015). On the other hand, there exists a potential loophole of misusing regression techniques if the underlying assumptions and limitations are not thoroughly understood. It is noteworthy that terminologies, theories and quantitative methods in connection with regression analysis were originally established in specialty disciplines such as applied statistics and economics. Hence, regression methods need to be interpreted and articulated in the terms comprehensible and acceptable to professionals in construction engineering. Moreover, regression-based predictive analytics need to be enhanced through integration, simplification, and customization in order to cater for application needs in practice and add to the body of knowledge in the construction engineering domain.

Multiple linear regression (MLR) is one of the regression techniques where two or more independent variables are used to predict a dependent variable. MLR applications in construction engineering literature include predicting building construction cost (Lowe et al. 2006), assessing the service condition of pipelines (El-abbasy et al. 2014) and predicting seismic retrofit construction (Jafarzadeh et al. 2014, 2015). Commonly-applied methods to fit an MLR model to a dataset and estimate regression coefficients include ordinary least square (OLS), generalized least squares (GLS), maximum likelihood estimators (MLE). In general, the straight-forward option for formulating MLR equations is the OLS method unless critical OLS assumptions are violated. The validity of the OLS formulated model needs to be verified by checking the following assumptions:

(1) There is no perfect linear dependence between input variables (no multicollinearity);

(2) The errors' variance is constant (homoscedasticity);

(3) The errors are serially independent (no autocorrelation);

(4) The errors follow a normal distribution.

The first assumption ensures that the regression variables are independent, and one cannot be linearly predicted from the other. If an input variable depends on other input variables, there is no need to include them all simultaneously in a regression model. In Figure 2-1, the area highlighted in grey represents the explanatory power of the group of input variables x in predicting the output Y. Although x_3 and x_2 individually have high explanatory power, they can almost be accounted by the other input variables (i.e. x_1 and x_4). In other words, if x_3 and x_2 are removed, the explanatory strength of the remaining group of input variables would not be affected. Violation of the first assumption is termed *multicollinearity*, which occurs when two or more input variables in an MLR model are highly correlated. The second assumption enforces the variance of error terms to be constant or homoscedastic. Violation of the second assumption is termed *heteroscedasticity*, which occurs when the variance of the error terms differs across observations. Heteroscedasticity can be visually inspected in regression residual plots, as shown in Figure 2-2, or formally examined by using White, Goldfeld-Quandt or Breusch-Pagan test (Kaufman 2013). In the presence of heteroscedasticity, coefficients estimated by OLS and also the error analysis would become inaccurate and unrealistic. The third assumption states that the error of different observations should be independent. *Autocorrelation* refers to the dependence of errors between serial observations over time. Autocorrelation can be visualized if the error of observation n (ε_n), on one axis is plotted against the error of previous observation (ε_{n-1}) on the other axis, which would reflect any correlation between the two (n is the order of the data being collected) (Figure 2-3). The fourth

assumption states that the errors of observations should follow a normal distribution (Figure 2-4). Based on Gauss-Markov theorem, verifying the underlying assumptions is important to justify the use of OLS and prove that OLS estimators (i.e. coefficients) are best linear unbiased estimators (BLUE) (Berry and Feldman 1985).

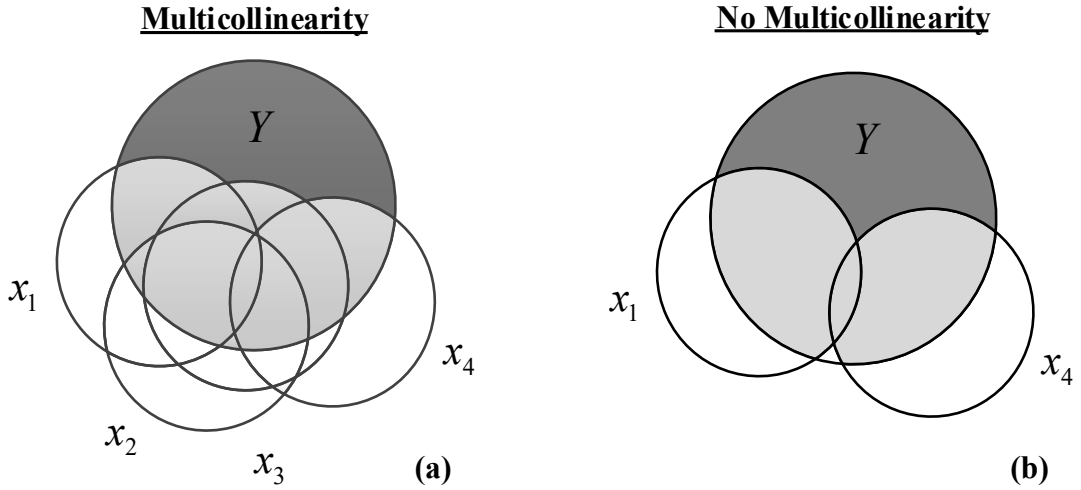

Figure 2-1:Contrasting (a) multicollinearity and (b) no multicollinearity (x3 and x2 are highly correlated to other input variables)

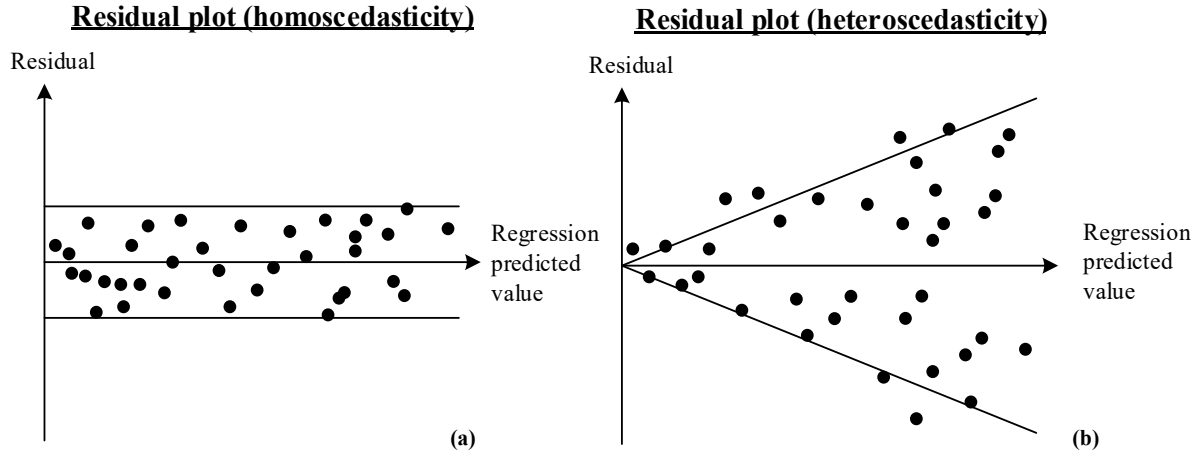

Figure 2-2:Residual plots for contrasting (a) homoscedasticity and (b) heteroscedasticity of errors

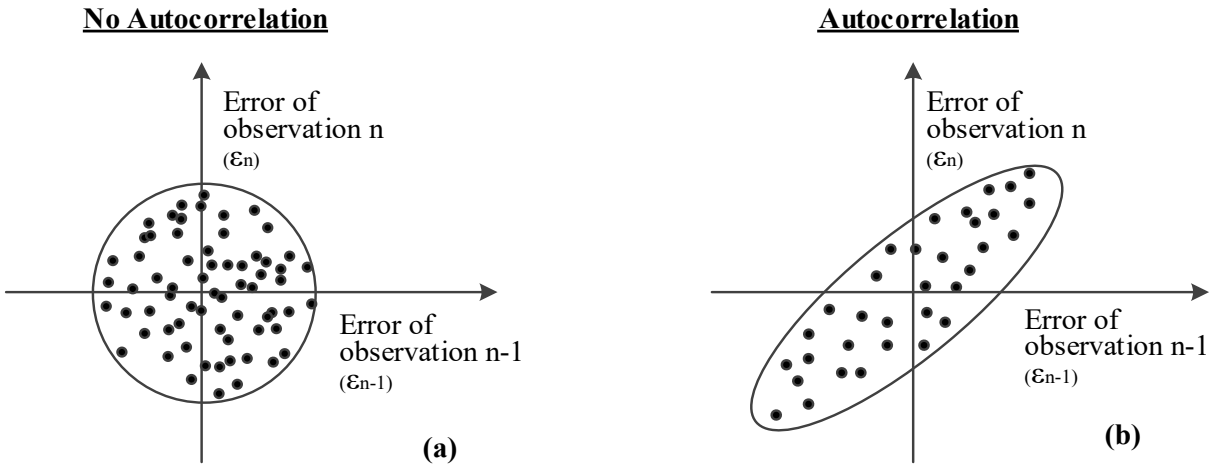

Figure 2-3:Comparing the (a) independence of errors in serial observations and (b) autocorrelation

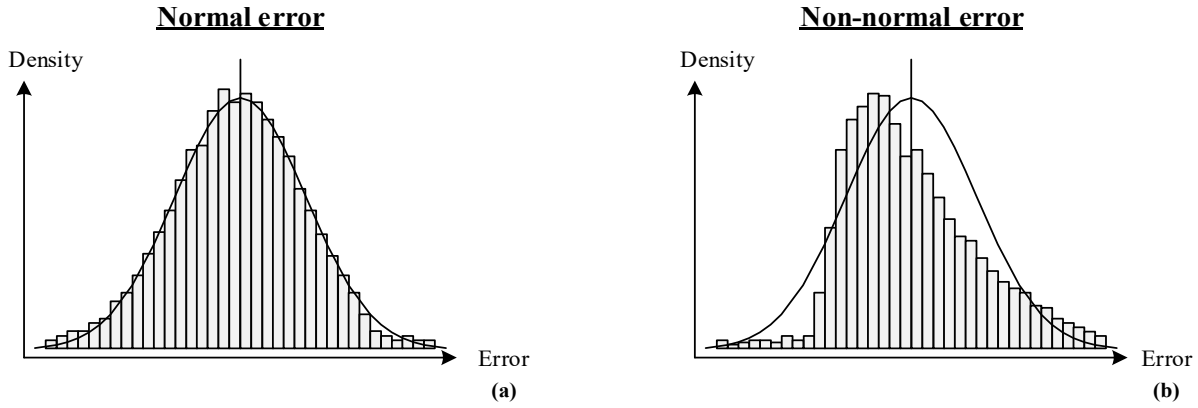

Figure 2-4:Contrasting the (a) normal of errors and (b) non-normal of errors

The primary challenge of applying the MLR technique is how to select a proper set of input variables so as to make accurate and reliable predictions. Appropriately choosing a subset of independent variables, in contrast to including all possible input variables, have both statistical and practical advantages in addition to model simplicity. From the statistical perspective, proper selection of input variables would reduce the chance of over-fitting and transferring noise in data into the regression model (Fox 1991). To elaborate more, if too many variables are used to formulate the regression model, the resulting equation tends to fit onto the noise in the data set (i.e., outliers), leading to over-fitting and unreliable prediction results. From the practical point of

view, being able to make reliable estimates while using fewer input variables would reduce data collection effort to enable the calibration and maintenance of the model. One plausible approach for determining the best subset of variables is by trial-and-error, which essentially enumerates all the potential combinations and compares the analytical results. However, it is not practically feasible in most of the cases, as there are 2^n possible subsets for a model with n input variables. For instance, if there are ten input variables, there would be 1,024 different regression models, which need to be evaluated.

Stepwise regression is a widely accepted method for reducing the number of input variables without sacrificing the prediction accuracy of the MLR model. The objective is to predict an output based on the regression equation formulated by a subset of input variables; while the regression equation retains all or most of the explanatory power as if the full set of variables were used (Barrett and Gray 1994). In stepwise regression, the input variables are selected by testing the significance of each one and its respective correlation with the output variable. It is worth mentioning that the commercial statistical software systems, such as SPSS® and SAS®, provide stepwise regression functionality. Nonetheless, all these systems function like "black box" and do not check the validity of OLS assumptions in a comprehensive and systematic fashion. Construction-related research studies have also been conducted for streamlining input variables by utilizing stepwise regression (Silva et al. 2013; Smith 1999; Wong 2004). However, only a few studies found in the literature have verified the underlying OLS assumptions (e.g., Choi et al. 2015; El-abbasy et al. 2014; Jafarzadeh et al. 2015). Moreover, determining the errors of the MLR's point-value estimate is crucial to regression applications in engineering. Error analysis can be conducted by establishing a confidence interval around the point-value estimate in order to reveal the uncertainty of the MLR prediction (Cheung and Skitmore 2006).

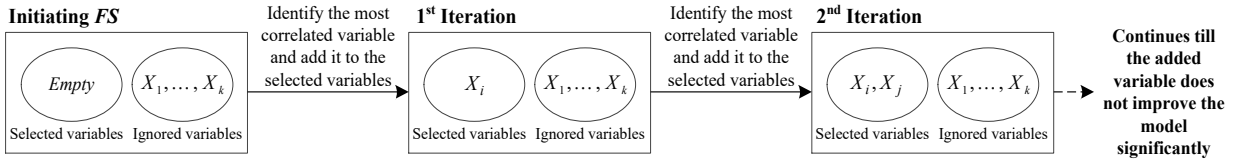

Figure 2-5:Forward selection procedure

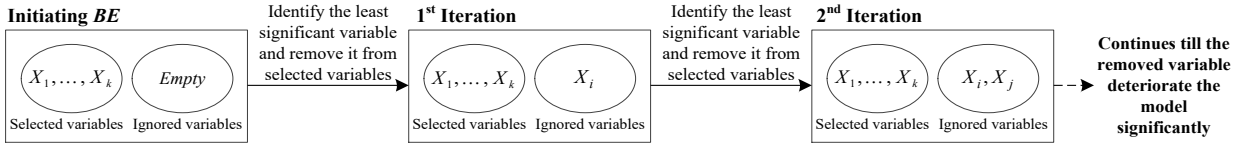

Figure 2-6: Backward elimination procedure

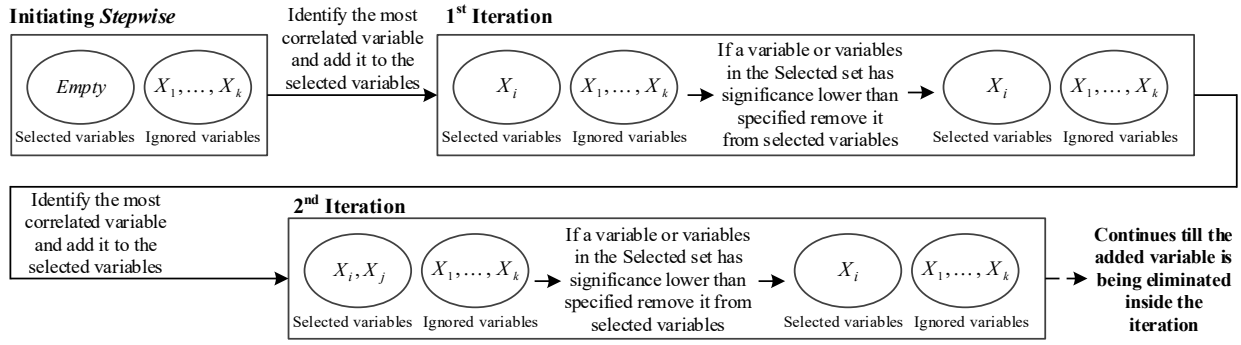

Figure 2-7: Proposed stepwise procedure

To the best of our knowledge, an integrated framework for MLR predictive application in construction engineering is absent in the literature. The proposed framework is intended to assist in formulating MLR models by analytically selecting input variables, verifying and validating the resulting models, and performing predictive error analysis. The objective of this research is to develop a systematic approach to reduce the effort of data collection in the construction field, provide valid point-value estimates based on a streamlined MLR equation, and quantify the error of the point-value estimate in terms of a range associated with a specific level of statistical confidence. In this research, we essentially address the following question: How to select the most

relevant MLR input variables and verify the BLUE assumptions such that reliable point and range estimates can be obtained given particular confidence levels?

In the following sections, literature reviews are provided on commonly used MLR techniques for estimation and prediction in construction engineering applications. As there is no single reference containing all the required procedures in a systematic form, a novel MLR application framework is proposed, consisting of Modified Stepwise Regression procedures for selecting input features and checking the BLUE assumptions. Two case studies are presented for demonstrating the application of the framework step by step and contrast the results against those obtained from commonly-applied MLR methodologies in validation of the framework. Each case study represents a typical application problem in construction engineering: (1) concrete workability control for quality and productivity in concreting construction; and (2) precast span installation cycle-time planning for resource use efficiency in precast construction. It is noteworthy that the dataset for the first case (demonstration case study) was sourced from the University of California, Irvine (UCI) machine learning repository for benchmarking new algorithms; while the dataset for the second case (practical case study) was prepared from a previous study which applied operations' simulation modelling to characterize the construction cycle-time in the field.

2.2 MLR LITERATURE REVIEW

MLR has shown its usefulness and practicality in construction engineering (El-abbasy et al. 2014; Jafarzadeh et al. 2014, 2015; Silva et al. 2013). MLR modelling formulates a linear equation by relating two or more independent variables with a dependent variable for estimation and prediction, as represented by Equation (2-1).

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \varepsilon \quad (2\text{-}1)$$

whereα =intercept,β_i=regression coefficients;ε =error;X_i=input variables; Y =dependent variable.

The literature review demonstrates a widespread use of MLR in the construction field. Smith (1999) investigated the earthmoving productivity in association with the operating conditions. Lowe et al. (2006) estimated the project cost of constructing buildings. Choi et al. (2013) identified the factors affecting pavement performance. El-abbasy et al. (2014) predicted the conditions of oil and gas pipelines. Jafarzadeh et al. (2015) predicted the seismic retrofit cost.

The number of input variables can be reduced by feature selection methods such as forward selection (FS), backward elimination (BE), and stepwise regression. As mentioned, the objective of feature selection methods is to predict an output variable by use of a subset of input variables while retaining all or most of the explanatory power given by the full set of variables. The FS starts with a linear regression model containing no input variables but only a constant term as shown in Figure 2-5 (Seber and Lee 2003). The input variables are added sequentially to the regression model based on correlation and statistical significance. The significance of a particular input variable is measured based on its contribution to explaining the output variable, compared against

other input variables. The iterative process is terminated when no remaining variable can be added to improve the model's predictive performance.

In contrast, BE begins with the full set of variables in formulating the linear regression model and removes insignificant variables to achieve the desired subset as presented in Figure 2-6 (Wang and Jain 2003). A major drawback of FS and BE methods lies in the fact that the iterative processes are susceptible to be trapped in a near-optimum subset of variables (Smith and Draper 1998). During the FS iteration, once an input variable is included; it will never be removed. Similarly, an input variable cannot be reintroduced into the linear regression model once the variable is eliminated in the BE process. As a result, applying FS and BE separately on a dataset may give rise to entirely different subsets of selected features (Thompson 1978). Stepwise regression overcomes this drawback by combining the FS and BE processes as demonstrated in Figure 2-7. It identifies significant input variables and eliminates multicollinearity between variables. Note that *multicollinearity* would incur when the selected input variables are correlated (Leung et al. 2001). Several recent studies have applied stepwise regression as the variable selection technique. For instance, Silva et al. (2013) reduced the input variables from fifteen to four while eliminating multicollinearity. Jafarzadeh et al. (2014) initially selected fourteen variables based on the seismic performance of a building and applied stepwise regression to select seven influential factors. Nevertheless, a major drawback in the previous MLR research related to stepwise regression is identified: the regression-based feature selection is performed by applying OLS functionality available in commercial statistical software without formally validating the BLUE assumptions underlying OLS, thus rendering the results to be less reliable.

In this research study, a modified version of stepwise regression provides the key component of the proposed application framework for selecting predictive input variables, which is coupled with the assessment of BLUE assumptions. In addition, the technique for characterizing the margin of error in the point-value estimate with statistical inference is also integrated into the MLR application framework in order to accommodate uncertainties in the predicted point-value by MLR.

2.3 PROPOSED MLR APPLICATION FRAMEWORK

To harness MLR's predictive power and develop a reliable model, a comprehensive framework is proposed, and steps for formulating, verifying, validating, and evaluating the resulting MLR equation are elaborated in this section.

2.3.1 Initializing regression model

The first step is to select an initial set of input variables which sufficiently explains the model's output. MLR coefficients can be obtained by applying OLS. Note OLS is the preferable technique to solve MLR due to its computational simplicity and well established theoretical foundation; in the case of violation of OLS assumptions, alternative methods that are actually developed based on OLS, such as the weighted least squares (WLS) to address the heteroscedasticity in applying OLS (which is explained further in later sections), can be considered to redefine the MLR model.

2.3.2 Performing variable selection

The stepwise regression is structured by selecting and removing input variables based on correlation and statistical significance analyses, as denoted in Figure 2-8.

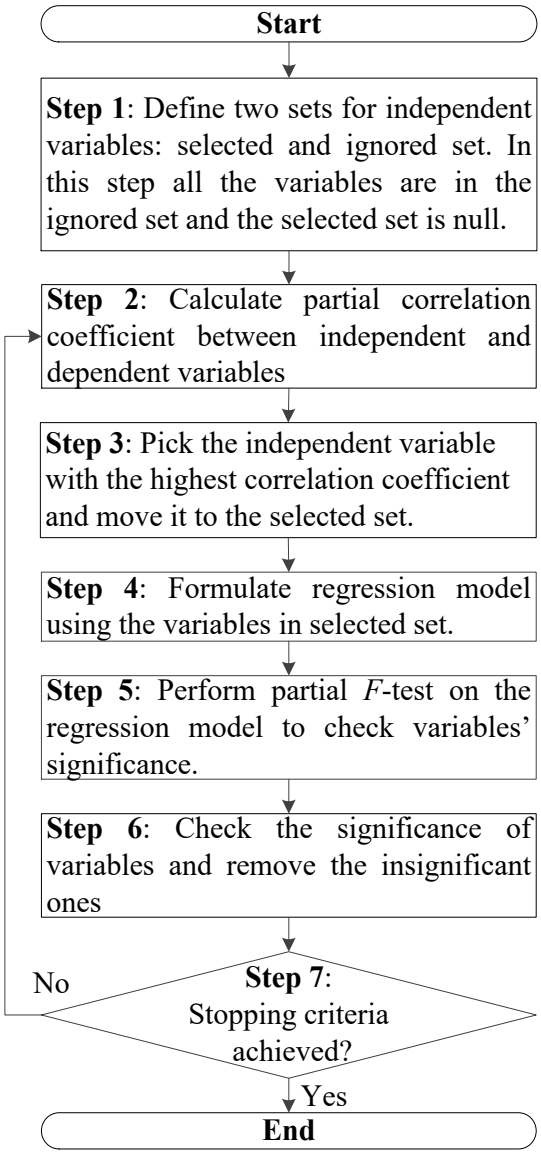

Figure 2-8: Flowchart of variable selection methodology

In **Step 1**, the variables are separated into two sets, namely, (I) *selected set*, denoted by$\{x_{i,sel}\}$ and (II) *ignored set*, denoted by$\{x_{i,ign}\}$. Through all the steps, variables included in the selected set are used to formulate the regression model. The variables in the ignored set are those that have not yet been tested or have been removed from the model because of low significance. All input variables are initialized in the ignored set at the beginning and the selected set is null.

Partial correlation is a measure of the linear relationship between two variables after removing the influence of other control variables. In regression analysis, each input variable contributes to accounting for the dependent variable (i.e., the explaining part of a dependent variable to which an input variable is attributed), as shown in Figure 2-8. In the proposed framework, partial correlation is used to identify a set variable which best describe the unexplained part of the dependent variable (Figure 2-9). Variables in the *selected set* are regarded as control variables when partial correlation is being determined between variables in the ignored set and the dependent variable.

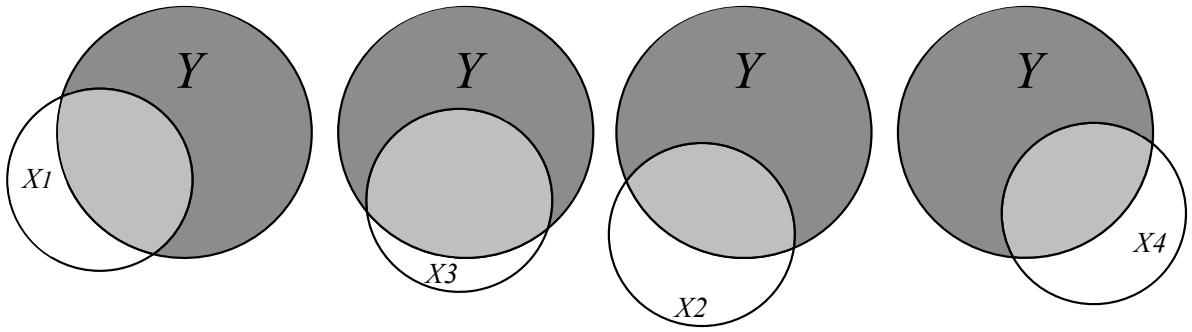

Figure 2-9: Explanatory power of individual input variables

In **Step 2**, the correlation coefficients are determined to measure the degree of association between the output (y) and ignored variables ($\{x_{i,ign}\}$), removing the effect of control variables. If the selected set is null (e.g., first iteration), the correlation coefficient is determined as Pearson's correlation coefficient (Equation 2-2). If there are variables in the selected set, the partial correlation coefficient is calculated as per (Equation 2-3). The correlation coefficient measures the correlation between each input variable x and the response variable y. The numerical boundaries of the coefficient are between [+1, −1]. A positive value indicates that the variables are positively

correlated, and a negative value means that the variables are negatively correlated. The magnitude of the value indicates the strength of correlation.

$$\text{If } \{x_{i,sel}\} = \varphi,\ r_{\{x_{i,ign}\}y} = \frac{\sum_n[(x_n-\bar{x})(y_n-\bar{y})]}{\sqrt{\sum_n(x_n-\bar{x})^2}\sqrt{\sum_n(y_n-\bar{y})^2}},\ \text{for each}\{x_{i,ign}\} \tag{2-2}$$

$$\text{If } \{x_{i,sel}\} \neq \varphi,\ r_{\{x_{i,ign}\}y\cdot\{x_{i,sel}\}} = \frac{r_{\{x_{i,ign}\}y}-r_{\{x_{i,sel}\}y}\times r_{\{x_{i,ign}\}\{x_{i,sel}\}}}{\sqrt{1-r^2_{\{x_{i,ign}\}\{x_{i,sel}\}}}\sqrt{1-r^2_{\{x_{i,sel}\}y}}},\ \text{for each}\{x_{i,ign}\} \tag{2-3}$$

where φ is the symbol for a null set, n is the number of data records, x and y are the values of input and response variables in the dataset, respectively; $\bar{x}$and$\bar{y}$are the mean values of input and response variables, respectively; r is the correlation coefficient.

After calculating the correlation coefficients, the variable with the highest correlation is moved out of the ignored set and added to the selected set in **Step 3**. The selected variable will be included in the regression model and all the variables' significance will be evaluated in the next step.

In **Step 4**, the regression model is formulated using all the variables stored in the selected set by implementing OLS.

Step 5 is to assess the significance of the variables in the regression model resulting from Step 4. To ensure that the selected variables are all significant in explaining the dependent variable, partial *F*-test needs to be performed. Partial *F*-test measures the significance of selected input variables by comparing two regression models, namely: the regression model prior to adding each input variable, and the regression model after adding each input variables. The partial *F*-test (denoted byF_{x_i}) is calculated based on the sum of squared error (*SSE*) (Equation 2-4) and the degree of freedom of the two regression models (Equation 2-5).

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \quad (2\text{-}4)$$

Where $y_i$is the dependent variable (given in the dataset) and$\hat{y}_i$ is the predicted value of the MLR model.

$$F_{x_i} = \frac{(SSE_{k-1,x_i} - SSE_k)}{(SSE_k)/(n-k-1)}, \text{ for each} x_i \in \{x_{i,sel}\} \quad (2\text{-}5)$$

To perform the partial *F*-test, the *SSE* values for the regression models before and after adding an input variable (i.e.,x_i) are computed.SSE_{k-1,x_i}represents the *SSE* before adding the input variable which has (*n*-*k*) degrees of freedom, whileSSE_krepresents the *SSE* after adding the input variable with (*n*-*k*-1) degree of freedom. *n* is the number of datasets. *k* is the number of input variables after adding$x_i$to MLR model.

In **Step 6**, the significance of each variable can be determined such that the variables deemed to be significant (*p-value*<0.05) can be included for formulating the regression model. If the variable is not significant (*p-value*>0.05), it should be removed from the regression model (i.e., removed from the selected set and added to the ignored set). In a particular iteration, if the variable with the highest correlation does not satisfy the required level of significance (*p-value*>0.05), the stepwise procedure ends, and no more iteration is required.

2.3.3 Model verification

The resulting MLR model, in the form of Equation (2-1), must satisfy the BLUE assumptions in order to be valid for estimation and prediction. The model is verified by checking the following six criteria:

(1) $E(\varepsilon_j) = 0$: The expected value of error term is zero. Violation of this assumption would affect the estimation of the intercept.

(2) $VAR(\varepsilon_j) = \sigma^2 I$: The variance of the error term is constant. If the variance of the error term is not constant, the MLR model is associated with heteroscedasticity.

(3) $COV(\varepsilon_j, \varepsilon_h) = 0$: The error terms are uncorrelated. Violation of this assumption causes autocorrelation.

(4) $COV(X_i, \varepsilon) = 0$: Each input variable is uncorrelated with the error term. Violation of this assumption would also result in heteroscedasticity.

(5) There is no perfect collinearity: There is no input variable that is perfectly linearly related to another input variable(s).

(6) $\varepsilon \approx N(0, \sigma^2)$: The error term must be normally distributed. Note, this assumption is only relevant to the tests of statistical significance (e.g., *T*-test and *F*-test); its violation would have no effect on the estimation of the MLR model coefficients. This check is generally required given small sample size (smaller than 30).

The Gauss-Markov theorem (Greene 2008) maintains that the least squares estimators of regression parameters are unbiased and efficient when assumptions 1 to 5 are satisfied; hence, the least square estimators are deemed to be BLUE. Herein, formally established methods in applied statistics to check the discussed BLUE assumptions are briefly described in here:

Heteroscedasticity is statistically detected by use of Breusch-Pagan test (Breusch and Pagan 1979). Breusch-Pagan tests the null hypothesis of constant error variances, against non-constant error variances (i.e., a function of one or more input variables) (Equation 2-6). In the case of rejection of the null hypothesis, the WLS method is opted to formulate the WLS-based regression model (Greene 2008; Gujarati 2004).

$$H_0: Var(\varepsilon|X_1, X_2, \ldots, X_k) = E(\varepsilon^2|X_1, X_2, \ldots, X_k) = \sigma^2$$

$$\varepsilon^2 = \delta_0 + \delta_1 X_1 + \delta_2 X_2 + \cdots + \delta_k X_k + e \quad (2\text{-}6)$$

$$H_1: \delta_1 = \delta_2 = \cdots = \delta_k$$

Multicollinearity is tested by examining the correlation between two input variables. Variance inflation factor (VIF) is a commonly accepted indicator of multicollinearity. To measure the VIF for each input variable$X_{i\in\{1,2,\ldots,k\}}$, an OLS regression is formed with$X_i$as the dependent variable, while all other variables are considered as input variables (Equation 2-7). Multicollinearity exists if the VIF value is higher than 10 (Kutner et al. 2004). There would be no sign of multicollinearity in the MLR model developed by stepwise regression as the significance of added variables are tested before including them in the model.

$$X_i = \alpha_0 + \alpha_1 X_1 + \alpha_2 X_2 + \cdots + \alpha_k X_k + e \quad (2\text{-}7)$$

$$VIF_i = (1 - R_i^2)^{-1}$$

whereR_i^2 =coefficient of determination.

Autocorrelation is detected by use of *d*-statistic test (i.e., Durbin-Watson test) (Durbin and Watson 1951). As per Equation (2-8), the computed value of *d* lies between 0 and 4. Autocorrelation exists if the *d* value is close to 4 or 0. Autocorrelation does not exist if the *d* value is adjacent to 2.

$$d = \frac{\sum_{t=2}^{n}(e_t - e_{t-1})^2}{\sum_{t=1}^{n} e_t^2} \quad (2\text{-}8)$$

where e_i is the regression residual of i^{th} observation.

Normality of the error term can be checked statistically by Anderson-Darling test (Stephens 1974), as shown in Equation 2-9. The hypothesis of normality is rejected if A exceeds a given critical value with presumed significance level [see Table 2-1 in Stephens (1974) for critical values].

$$A^2 = -n - S \quad (2\text{-}9)$$

Where n is the sample size, and S can be obtained from Equation 2-10.

$$S = \sum_{i=1}^{n} \frac{(2i-1)}{n} [ln\, F\,(e_i) + ln(\,1 - F(e_{n+1-i}))] \quad (2\text{-}10)$$

Where n is the sample size; F =cumulative distribution of errors, e_i= regression error of i^{th} observation.

2.3.4 Model validation

Model validation is vitally important to ensure the prediction performance of a calibrated MLR model. The proposed application framework utilizes two cross-validation methods: (I) *k*-fold cross-validation, and (II) *predicted residual error sum of squares (PRESS)* statistic (Holiday et al. 1995). The cross-validation is used for checking the problem of over-fitting, which occurs when a regression model performs reliably on the training set but poorly on new data. The K-fold cross-

validation splits the data into a test set and a training set for *k* times. The training set is used for calibrating the model whereas the test set is used for validation. The *PRESS* statistic is also a widely used method to determine the quality of prediction by each time leaving one observation out for testing the trained model (i.e. K-fold when *k* is equal to the number of observations) (Equation 11) (Choi et al. 2013). By performing cross-validation methods, the sum of squared errors (*SSE*) for each test set can be obtained and compared with the calibrated MLR *SSE*. The test *SSE* cannot be smaller than the MLR *SSE*. If the two *SSE* values are close to each other (e.g. within 10%-15% difference), the MLR model is validated. If the test *SSE* is significantly larger than the model *SSE*, the model is not valid.

$$PRESS = SS(PRESS \text{ residuals}) = \sum_{i=1}^{n} e_i^{*2} = \sum_{i=1}^{n} (y(x_i) - \hat{y}_{PRESS}(x_i))^2 \quad (2\text{-}11)$$

where$y$is the recorded dependent variable;$\hat{y}_{PRESS}$is the predicted value of the MLR model calibrated on the data set excludingx_i.\

2.3.5 Prediction error estimation

The proposed framework further quantifies the uncertainty in MLR predictions, by defining a confidence interval around its point-value estimates. The MLR uncertainty is attributed to both modelling errors and observation errors in the collected data. The confidence interval for a point-value estimate is statistically defined by Equation (2-12) (Liu 2010). The σ_{res} is the residual standard deviation which accounts for the uncertainties in formulating the MLR model.

$$\hat{y}_0 \pm t_{(\alpha/2, n-k-1)} \times s.e. \quad (2\text{-}12)$$

where $\hat{y}_0$ is the predicted point value of the regression model;$t_{(\alpha/2,n-k-1)}$is the *T*-distribution with significance of α (degree of freedom of *n*-*k*-1); and$s.e.$ is the standard error of the estimate determined based on the training set by Equation (2-13).

$$s.e. = \sqrt{\sigma_{res}^2 \ [x_0](X^T X)^{-1}[x_0]^T} \tag{2-13}$$

whereσ_{res} is the residual standard deviation and can be calculated by Equation (2-15);$[x_0]$ in form of $[1 \quad x_{01} \quad \cdots \quad x_{0k}]$ is an array of input variable for which the confidence interval of its MLR model output needs to be established; and *X* with *n* rows (*n*=number of records) and *k*+1 columns (*k*=number of independent variables in MLR) is the matrix of recorded data (Equation 2-14).

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1k} \\ 1 & x_{21} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & \cdots & x_{nk} \end{bmatrix} \tag{2-14}$$

$$\sigma_{res}^2 = \frac{SSE}{n-k-1} \tag{2-15}$$

where SSE is the regression sum of squared errors and *n*-*k*-1 is the degree of freedom of the regression model.

2.4 DEMONSTRATION CASE

To demonstrate calculation procedures of the proposed MLR application framework, a benchmark dataset is selected from UCI machine learning repository (https://archive.ics.uci.edu/ml/index.html) which was collected and utilized by Yeh (2006), (2007), and (2009) for developing new non-linear algorithms. The problem is to model the slump of a concrete mix with different properties. The slump of concrete is not only determined by the water content, but also influenced by other concrete ingredients. The seven attributes in the collected data are cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, and fine aggregate (all in kg/m^3); the dependent variable is the concrete slump in *cm* for each experimented concrete mix (ranging from 0 to 30 *cm*). Yeh (2006) implemented Artificial Neural Network (ANN) and non-linear regression modelling to predict the concrete slump based on its ingredients. In this demonstration case, an MLR model is developed by applying the proposed framework, and the results are compared with predictive models reported in Yeh (2006).

2.4.1 Initializing regression model

The data set includes one hundred and three records. There are seven input variables, and three dependent variables. The data set input features are cement(x_1), blast furnace slag(x_2), fly ash(x_3), water(x_4), superplasticizer(x_5), coarse aggregate(x_6), and fine aggregate(x_7). In the recorded datasets, the dependent variables are slump, flow, and compressive strength; however, in this demonstration case, only slump is selected to be predicted by an MLR model.

2.4.2 Performing variable selection

The following demonstrate the main steps and details in performing stepwise regression for input variable selection as proposed in the framework:

1st Iteration

Step 1: Initializing the selected and ignored variable sets

- Selected set$\{x_{i,sel}\} = \varphi$
- Ignored set$\{x_{i,ign}\} = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$

Step 2: Performing correlation analysis.

- The correlations between the dependent variable (y) and input variables (x_1, x_2, x_3, x_4, x_5, x_6, x_7) are calculated and presented in Table 2-1. Since the variable x_4 has the highest correlation coefficient, it is chosen to be included in the MLR model.

Table 2-1: Correlation coefficients (1st Iteration)

Correlation coefficient	$r_{x_{i,ign}y}$	$(r_{x_{i,ign}y})^2$
r_{x_1y}	0.15	0.02
r_{x_2y}	-0.28	0.08
r_{x_3y}	-0.12	0.01
r_{x_4y}	0.47	0.22
r_{x_5y}	-0.21	0.05
r_{x_6y}	-0.19	0.04
r_{x_7y}	0.20	0.04

Step 3: Refining the selected and ignored variable sets.

- Selected set$\{x_{i,sel}\} = \{x_4\}$

- Ignored set$\{x_{i,ign}\} = \{x_1, x_2, x_3, x_5, x_6, x_7\}$

Step 4: Formulating the regression model with variables in the selected set by OLS.

- The regression is formulated as per Equation 2-16.

$$y = -21.787 + 0.202x_4 \tag{2-16}$$

Step 5: Performing partial *F*-test and testing the variables significance.

- In the 1st iteration, there is no variable in the model prior to adding the input variablex_4. The $SSE_{0,x_i}$is calculated by Equation 2-17. In a regression model with no input variables all the predicted values$(\hat{y}_i)$ are the mean value of the dependent variable$(\bar{y})$. The $SSE_1$is determined as per Equation 2-18. The predicted values$(\hat{y}_i)$ in $SSE_1$are calculated from the regression model formulated in Step 4 $F_{x_4}$is computed using SSE_{0,x_i}andSSE_1 by Equation 2-19 and$p_{partial,x_4}$is determined as per Equation 2-20.

$$SSE_{0,x_4} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \bar{y})^2 = (23 - 18.048)^2 + \ldots + (29 - 18.048)^2 = 7810.882 \tag{2-17}$$

$$SSE_1 = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = (23 - 20.641)^2 + \ldots + (29 - 23.338)^2 = 6110.563 \tag{2-18}$$

$$F_{x_4} = \frac{(SSE_{k,x_4} - SSE_{k+m})/(m)}{(SSE_{k+m})/(n-k-m-1)} = \frac{(SSE_{0,x_4} - SSE_1)/(1)}{(SSE_1)/(103-2)} = \frac{(7810.882 - 6110.563)/(1)}{(6110.563)/(101)} = 28.104 \tag{2-19}$$

$$p_{partial,x_4} = f(F_{x_i}, m, n-k-1) = f(28.104,1,101) = 6.78e^{-7} < 0.05 \quad (2\text{-}20)$$

Step 6: Testing the input variables' significance.

- Since the *p*-value for all the input variables is lower than 0.05 (i.e. accepted significance level), all variables included in the model are significant.

Step 7: Deciding if the stepwise regression should end.

- The last variable entered is significant enough, return to Step 2.

2nd iteration

Step 2: Performing correlation analysis.

The partial correlation coefficients between y and the ignored variables $\{x_1, x_2, x_3, x_5, x_6, x_7\}$ are calculated, given x_4 as the control variable. The correlation coefficients are tabulated in Table 2-2.

Table 2-2: Correlation coefficients (2nd iteration)

Correlation coefficient	$r_{x_{i,ign}y}$	$(r_{x_{i,ign}y})^2$
$r_{x_1y\cdot x_4}$	$4.96e^{-2}$	$2.46e^{-3}$
$r_{x_2y\cdot x_4}$	$-3.07e^{-1}$	$9.43e^{-2}$
$r_{x_3y\cdot x_4}$	$-7.02e^{-3}$	$4.93e^{-5}$
$r_{x_5y\cdot x_4}$	$-1.61e^{-1}$	$2.58e^{-2}$
$r_{x_6y\cdot x_4}$	$1.31e^{-1}$	$1.73e^{-2}$
$r_{x_7y\cdot x_4}$	$1.69e^{-1}$	$2.87e^{-2}$

- Selected set$\{x_{i,sel}\} = \{x_4\}$

- Ignored set$\{x_{i,ign}\} = \{x_1, x_2, x_3, x_5, x_6, x_7\}$

Step 3: Refining the selected and ignored variable sets.

- Selected set$\{x_{i,sel}\} = \{x_4, x_2\}$
- Ignored set$\{x_{i,ign}\} = \{x_1, x_3, x_5, x_6, x_7\}$

Step 4: Formulating the regression model with variables in the selected set by OLS.

- Formulate the regression model with x4 and x2 (Equation 2-21).

$$y = -18.099 + 0.199x_4 - 0.039x_2 \tag{2-21}$$

Step 5: Performing partial F-test and testing the variables' significance.

Perform partial *F*-test (Equations 22–28)

$$SSE_{1,x_2} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = (23 - 17.883)^2 + \ldots + (29 - 21.24)^2 = 7180.726 \tag{2-22}$$

$$SSE_{1,x_4} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = (23 - 20.641)^2 + \ldots + (29 - 23.338)^2 = 6110.563 \tag{2-23}$$

$$SSE_2 = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = (23 - 20.442)^2 + \ldots + (29 - 23.36)^2 = 5534.204 \tag{2-24}$$

$$F_{x_2} = \frac{(SSE_{1,x_2} - SSE_2)/1}{(SSE_2)/27} = \frac{(6110.563 - 5534.204)/1}{(5534.204)/100} = 10.414 \tag{2-25}$$

$$p_{partial,x_2} = f(10.414, m, n-k-1) = f(10.414,1,100) = 1.69e^{-3} < 0.05 \quad (2\text{-}26)$$

$$F_{x_4} = \frac{(SSE_{1,x_4} - SSE_2)/1}{(SSE_2)/100} = \frac{(7180.726 - 5534.204)/1}{(5534.204)/100} = 29.751 \quad (2\text{-}27)$$

$$p_{partial,x_4} = f(29.751, m, n-k-1) = f(29.751,1,100) = 3.56e^{-7} < 0.05 \quad (2\text{-}28)$$

Step 6: Testing the input variables' significance.

- Since the p-value for x_4 and x_2 is lower than 0.05, all the variables included in the model are significant.

Step 7: Deciding if the stepwise regression should end.

- The last variable entered is significant enough, return to Step 2.

3rd Iteration

Step 2: Performing correlation analysis, Table 2-3.

Table 2-3: Correlation coefficients (3rd Iteration)

Correlation coefficient	$r_{x_{i,ign}y}$	$(r_{x_{i,ign}y})^2$
$r_{x_1y\cdot x_4x_2}$	$-2.74e^{-2}$	$7.51e^{-4}$
$r_{x_3y\cdot x_4x_2}$	$-1.24e^{-1}$	$1.54e^{-2}$
$r_{x_5y\cdot x_4x_2}$	$-1.62e^{-1}$	$2.63e^{-2}$
$r_{x_6y\cdot x_4x_2}$	$4.31e^{-2}$	$1.86e^{-3}$
$r_{x_7y\cdot x_4x_2}$	$1.21e^{-1}$	$1.47e^{-2}$

Step 3: Refining the selected and ignored variable sets.

- Selected set $\{x_{i,sel}\} = \{x_4, x_2, x_5\}$
- Ignored set $\{x_{i,ign}\} = \{x_1, x_3, x_6, x_7\}$

Step 4: Formulating the regression model with variables in the selected set (Equation 2-29).

$$y = -15.744 - 0.036x_2 + 0.195x_4 - 0.205x_5 \tag{2-29}$$

Step 5: Performing partial F-test and testing the variables significance (Equations 2-30 to 2-35).

$$F_{x_2} = \frac{(SSE_{2,x_2} - SSE_3)/1}{(SSE_3)/99} = \frac{(5952.849 - 5504.274)/1}{(5504.274)/99} = 8.068 \tag{2-30}$$

$$p_{partial,x_2} = f(8.068, m, n-k-1) = f(38.84,1,99) = 5.47e^{-3} < 0.05 \tag{2-31}$$

$$F_{x_4} = \frac{(SSE_{2,x_4} - SSE_3)/1}{(SSE_3)/99} = \frac{(7044.202 - 5504.274)/1}{(5504.274)/99} = 27.697 \tag{2-32}$$

$$p_{partial,x_4} = f(24.29, m, n-k-1) = f(24.29,1,99) = 8.24e^{-7} < 0.05 \tag{2-33}$$

$$F_{x_5} = \frac{(SSE_{2,x_5} - SSE_3)/1}{(SSE_3)/99} = \frac{(5534.204 - 5504.274)/1}{(5504.274)/99} = 0.538 \tag{2-34}$$

$$p_{partial,x_5} = f(0.538, m, n-k-1) = f(0.538,1,99) = 0.465 > 0.05 \tag{2-35}$$

Step 6: Testing the input variables' significance.

- The p-value for x_5 is higher than 0.05, hence x_5 should return back to the ignored set.

Step 7: Deciding if the stepwise regression should end.

- The last variable entered is not significant, and the stepwise procedure ends here (no more iteration is required). The final MLR only includes water(x_4) and blast furnace slag(x_2) as input variables and is presented by Equation (2-20).

2.4.3 Model verification

The BLUE assumptions are verified for the resulting MLR model (Equation 2-20).

2.4.3.1 Checking heteroscedasticity

The Breusch-Pagan results are shown in Table 2-4. As the *p*-value is smaller than 0.05 (i.e., significant level). Therefore, the null hypothesis of constant variance for the error terms is rejected and the MLR experiences heteroscedasticity. Moreover, a fan-shape pattern can be observed in the OLS-based MLR residual plot which supports the heteroscedasticy identified by Beusch-Pagan test (Figure 2-10).

Table 2-4: Breusch-Pagan test for OLS-based regression model

Variable	Degree of freedom	χ^2	*p*-value
x_2, x_4	100	128.53	0.0288

To adjust the MLR model with heteroscedastic errors, the WLS method is utilized. The MLR model with the input variables x_2 and x_4 is used for WLS coefficient estimation. Equation (2-36) shows the final WLS-based MLR model. The residual plot for WLS-based regression depicts a constant variance of error, shown in Figure 2-10. The Breusch-Pagan test is conducted again, and the test results are shown in Table 2-5. Since the Breusch-Pagan *p*-value is larger than 0.05, the variance of the error terms is constant, and the errors are now homoscedastic.

Table 2-5: Breusch-Pagan test for WLS-based regression model

Variable	Degree of freedom	χ^2	p-value
x_2, x_4	100	101.5364	0.4383

$$y = 4.1742 + 0.0093x_2 + 0.075x_4 \tag{2-36}$$

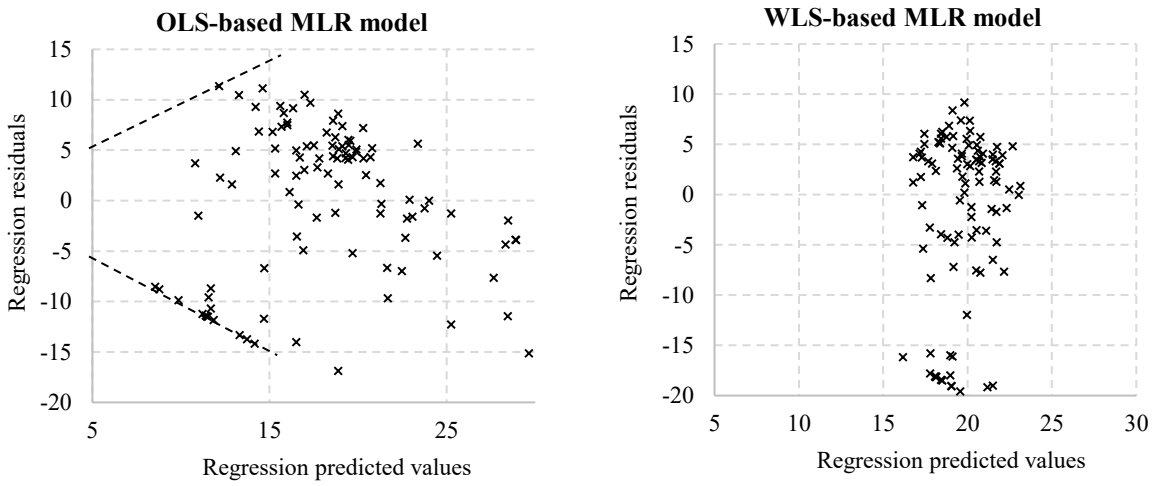

Figure 2-10: Visual comparison WLS-based MLR model and OLS-based MLR model residual plots

2.4.3.2 Checking multicollinearity

The VIF values are computed using Equation (2-7). For both input variables (x_2 and x_4) the VIF is equal to 1 leading to no multicollinearity in the MLR model as expected from stepwise regression, Table 2-6.

Table 2-6: VIF values for checking multicollinearity

Variable	VIF
x_2	1
x_4	1

2.4.3.3 Checking autocorrelation

Durbin-Watson test is conducted to calculate the *d*-statistic value. The result shows that the *d* value is 1.806, which is closed to 2. Thus, autocorrelation does not exist between the error terms.

2.4.3.4 Checking normality of error

The normality assumption is critical when if the sample size is small. In the current case, the number of data set is larger than 100 and checking the normality of error term can be neglected.

2.4.4 MLR model validation

To validate the MLR model's prediction performance, both K-fold cross-validation method and *PRESS* statistics are tested. In the K-fold method, *k* is assumed to be ten (10) and the *SSE* value for each test is given in Table 2-7. The total *SSE* and the *PRESS* values are calculated by Equation (2-37) and Equation (2-38), respectively. In both tests, the resulting *SSE* values are compared with the WLS-based MLR model *SSE* (Equation 2-39) and (Equation 2-40). This model is validated since both *SSE* ratios are in the acceptable range (10%-15%).

Table 2-7: SSE values for k-fold cross-validation

Test No.	1	2	3	4	5	6	7	8	9	10
SSE	659.45	885.39	1148.72	1171.79	164.24	499.29	929.35	350.87	659.45	885.39

$$SSE_{k-fold} = \sum_{i=1}^{k} Test_i(SSE) = 7572.3 \tag{2-37}$$

$$PRESS = SS(PRESS\ \text{residuals}) = \sum_{i=1}^{n} e_i^{*2} = 13.43 + 330.09 + \ldots + 27.92 \tag{2-38}$$

$$= 7609.99$$

$$\frac{SSE_{k-fold}}{SSE_{WLS\ Model}} = \frac{7572.3}{7371.3} = 1.027 \tag{2-39}$$

$$\frac{PRESS}{SSE_{WLS\ Model}} = \frac{7609.99}{7371.3} = 1.032 \tag{2-40}$$

Further, the presented MLR model is comparable with the one resulting from applying MLR by simply including all the input variables into the model.

2.4.5 Error estimation

To estimate the error of point prediction with a significance level of 0.05 (i.e. confidence level of 95%), a data point is postulated (Equation 2-41). To predict the concrete slump (*cm*), the blast furnace slag(x_2) and the water(x_4) are assumed as 100 (kg/m^3) and 196 (kg/m^3), respectively. Equation (2-42) shows the point prediction based on the final MLR model (Equation 36). The predicted value of the slump, given the assumed data point, is 19.8042 *cm*.

$$[x_0] = [1 \quad 100 \quad 196] \tag{2-41}$$

$$\hat{y}_0 = 4.1742 + 0.0093x_2 + 0.075x_4 = 19.8042 \tag{2-42}$$

Equations (2-43) to (2-46) show the calculations of the residual standard deviation(σ_{res}) (Equation 2-15), the matrix of records *X* (Eq. 14), and standard error($s.e.$) (Equation 2-13). Note that the number of observations is 103 (=*n*), and the input variables are 2 (=*k*); therefore, the degree of freedom is 100 (=*n-k-1*). As a result of the error analysis, with 95% confidence level, the interval

of the point prediction is between 18.191 *cm* and 21.417 *cm*. The results are cross-checked with the result of the actual observation, which is equal to 20 *cm*.

$$\sigma_{res}^2 = \frac{SSE}{n-k-1} = \frac{7371.3}{100} = 73.713 \tag{2-43}$$

$$(X^T X)^{-1} = \begin{bmatrix} 9.67e^{-01} & -2.52e^{-04} & -4.75e^{-03} \\ -2.52e^{-04} & 2.68e^{-06} & 2.15e^{-07} \\ -4.75e^{-03} & 2.15e^{-07} & 2.40e^{-05} \end{bmatrix} \tag{2-44}$$

$$s.e. = \sqrt{\sigma_{res}^2 [x_0](X^T X)^{-1}[x_0]^T} = \sqrt{73.713 \times 0.0110} = 0.8132 \tag{2-45}$$

$$\hat{y}_0 \pm t_{(0.025,100)} \times s.e = 19.804 \pm 1.984 \times 0.8132 = 19.804 \pm 1.613 \tag{2-46}$$

2.4.6 Benchmarking against published non-linear regression models

Based on the same dataset, Yeh (2006) produced two predictive models, namely: a non-linear regression model and an ANN model. The non-linear model involved seven inputs and required estimating fifty-six (56) regression coefficients, as shown in Equation (2-47). The ANN model calibration required tedious trial-and-error processes guided by heuristic rules in order to fix the number of hidden layers, the number of hidden nodes, the learning rate and the momentum factor. Additionally, fifty-six (56) ANN transformation weights were estimated. For the current case, the following ANN parameters were reported for the final model in Yeh (2006): 1 hidden layer and seven hidden nodes; the learning rate and momentum factor were set as 0.1 and 0.5 respectively. The *RMSE* values can be calculated based on *SSE* (Equation 48). Yeh (2006) reported the *RMSE* values for the ANN model and the non-linear regression model being 4.03 *cm* and 9.29

cm, respectively. In contrast, the proposed linear regression (Equation 2-36) would require the estimation of only three coefficients, resulting in the *RMSE* value of 8.46 *cm.* Even though the number of independent variables has been significantly reduced (from seven to two), the predictive power of the streamlined MLR model is not compromised. In short, simplicity is retained in the MLR model without losing sophistication of the model in coping with the complexity in the real-world application.

$$y = \sum_{i=1}^{q} \beta_i x_i + \sum_{i<j}^{q} \sum \beta_{ij} x_i x_j \quad (2\text{-}47)$$

where x_i is the i^{th} independent variables, q=total number of independent variables; β_i,$\beta_{ij}$are regression coefficients.

$$RMSE = \sqrt{(SSE/n)} \quad (2\text{-}48)$$

2.5 PRACTICAL CASE STUDY

To demonstrate the usefulness of the proposed framework, a practical case study on predicting one span installation cycle-time on precast viaduct construction is presented. The viaduct is part of an artery linking Hong Kong and Shenzhen, China, and consists of 227 post-tensioned spans. A typical span is made up of fourteen precast segmental box girders (12m×2.5m×2.8m of each). The stepping girder precast installation method was used to accelerate the viaduct construction process (Chan and Lu 2008). The precast segments were fabricated near Shenzhen and hauled to the Hong Kong site for installation. A main field constraint was that the site was too congested to keep all segments in the convenient proximity of the site crew. As an alternative, the precast segments were

partially stocked in a remote storage area and transported to the working span by trailer trucks, without any intermediate storage or buffer.

In order to assist the contractors in estimating the cycle-time for installing the precast concrete segments on one-span viaduct, four input factors relevant to site operations and logistics planning were identified and assessed, namely: (1) the number of trailer trucks rented for hauling segments (the site only considered the options of two trailer trucks or three), (2) one batch or two-batch precast segment delivery modes (fourteen segments can be delivered either in one-batch in the night before installation operation starts or in two batches, which means the first batch of seven segments would be delivered in the night before installation starts and the second batch delivered in the following night), (3) the percentage of the total number of segments on one-span to be placed in the remote storage area, and (4) the haul duration for a trailer truck to transit from the remote storage area to the working span. Table 2-8 shows the thirty cycle-time records used in this research.

Table 2-8: Recorded cycle-time for installing the precast concrete segments

Data record identifiers	No. of trucker (x_1)	Delivery batch (x_2)	Segment at remote storage area (%) (x_3)	Duration to remote storage area (x_4)	Desired install hours (y)
1	2	1	0.00	0.00	103.61
2	3	1	0.50	0.50	104.76
3	3	1	0.29	0.33	104.76
4	3	1	0.50	0.33	104.78
5	3	1	1.00	0.50	105.78
6	3	1	1.00	0.33	105.78
7	3	1	0.29	0.75	108.38
8	3	1	0.50	0.75	109.36
9	2	1	0.50	0.50	111.51
10	3	1	1.00	0.75	112.05
11	3	1	0.71	0.75	112.41
12	2	1	0.29	0.75	112.72
13	2	1	1.00	0.50	114.15
14	2	1	0.50	0.75	115.70
15	2	1	0.71	0.75	116.47
16	2	1	1.00	0.75	116.61
17	2	2	0.29	0.50	116.67
18	2	2	1.00	0.33	116.70
19	2	2	1.00	0.75	116.71
20	2	2	0.00	0.00	116.74
21	2	2	0.57	0.50	116.74
22	2	2	0.57	0.75	116.74
23	3	2	0.57	0.50	116.74
24	3	2	0.57	0.75	116.74
25	3	1	0.29	0.50	104.76
26	3	1	0.71	0.50	104.89
27	3	1	0.71	0.33	104.89
28	3	1	0.00	0.00	105.77
29	2	1	0.71	0.33	106.00
30	2	1	0.29	0.50	108.47

Next, the proposed framework is implemented on the dataset to develop a simple MLR predictive model.

1st iteration

The correlations between the dependent variable (y) and input parameters (x_1, x_2, x_3, x_4) are calculated as shown in Table 2-9. Since x_2 (one-batch or two batches precast segments delivery) has the largest correlation, it will be moved to selected set of variables.

Table 2-9: Correlation coefficient (1st iteration)

Correlation coefficient	$(r_{x_{i,ign}y})^2$
r_{x_1y}	0.370
r_{x_2y}	0.453
r_{x_3y}	0.041
r_{x_4y}	0.130

Selected set$\{x_{i,sel}\} = \{x_2\}$

Ignored set$\{x_{i,ign}\} = \{x_1, x_3, x_4\}$

Formulate the regression model with x_2 (Equation 2-49).

$$y = 100.88 + 7.92x_2 \tag{2-49}$$

Perform partial F-test (Equations 2-50 and 2-51).

$$F_{x_2} = \frac{(SSE_{k,x_2} - SSE_{k+m})/(m)}{(SSE_{k+m})/(n-k-m-1)} = \frac{(SSE_{0,x_2} - SSE_1)/(1)}{(SSE_1)/(30-2)} = \frac{(769.05 - 400.86)/(1)}{(400.86)/(28)} = 25.718 \tag{2-50}$$

$$p_{partial,x_2} = f(F_{x_i}, m, n-k-1) = f(25.718,1,28) = 2.282e^{-5} < 0.05 \tag{2-51}$$

Since the p-value for x_2 is lower than 0.05, x_2 is significant.

2nd iteration

The partial correlation coefficients between y and the ignored variables (x_1, x_3, x_4) are calculated, given x_2 as the control variable (Table 2-10). Since x_4 (haul duration from the remote storage area to the working span) has the largest correlation, it will be moved to selected set of variables.

Table 2-10: Correlation coefficient (2nd iteration)

Correlation coefficient	$(r_{x_{i,ign}y})^2$
$r_{x_1y\cdot x_2}$	0.303
$r_{x_3y\cdot x_2}$	0.099
$r_{x_4y\cdot x_2}$	0.331

Selected set$\{x_{i,sel}\} = \{x_2, x_4\}$

Ignored set$\{x_{i,ign}\} = \{x_1, x_3\}$

Formulate the regression model with x_2 and x_4 (Equation 2-52).

$$y = 96.039 + 7.891x_2 + 9.608x_4 \tag{2-52}$$

Perform partial F-test (Equations 2-53 to 2-56).

$$F_{x_4} = \frac{(SSE_{1,x_4} - v)/1}{(SSE_2)/27} = \frac{(400.866 - 252.327)/1}{(252.327)/27} = 15.894 \tag{2-53}$$

$$p_{partial,x_4} = f(15.894, m, n-k-1) = f(15.894,1,27) = 0.000 < 0.05 \tag{2-54}$$

$$F_{x_2} = \frac{(SSE_{1,x_2} - SSE_2)/1}{(SSE_2)/27} = \frac{(617.663 - 252.327)/1}{(252.327)/27} = 39.092 \tag{2-55}$$

$$p_{partial,x_2} = f(39.092, m, n-k-1) = f(39.092,1,27) = 0.000 < 0.05 \tag{2-56}$$

Since the p-value for x_2 and x_4 is lower than 0.05, both variables are significant.

3rd iteration

The partial correlation coefficients between *y* and the remaining variables (x_1, x_3) are calculated, given x_2 and x_4 as the control variables (Table 2-11). Since x_1 (the number of trailer trucks) has the largest correlation, it will be moved to selected set of variables.

Table 2-11:Correlation coefficient (3rd iteration)

Correlation coefficient	$(r_{x_{i,ign}y})^2$
$r_{x_1y\cdot x_2x_4}$	0.496
$r_{x_3y\cdot x_2x_4}$	0.014

Selected set$\{x_{i,sel}\} = \{x_2, x_4, x_1\}$

Ignored set$\{x_{i,ign}\} = \{x_3\}$

Formulate the regression model (Equation 2-57).

$$y = 107.107 + 6.610x_2 + 9.518x_4 - 3.759x_1 \quad (2\text{-}57)$$

Perform partial *F*-test (Equations 2-58 to2-60).

$$p_{partial,x_2} = f(38.84, m, n-k-1) = f(38.84,1,26) = 0.000 < 0.05 \quad (2\text{-}58)$$

$$p_{partial,x_4} = f(24.29, m, n-k-1) = f(24.29,1,26) = 0.000 < 0.05 \quad (2\text{-}59)$$

$$p_{partial,x_2} = f(16.062, m, n-k-1) = f(16.062,1,26) = 0.000 < 0.05 \quad (2\text{-}60)$$

Since the p-value for all input variables is lower than 0.05, all variables are significant.
4th iteration

The partial correlation coefficients between *y* and the last ignored variable (x_3) is calculated, given x_1, x_2, and x_4 as the control variables (Table 2-12). Since x_3 (the percentage of the total number of

segments on one-span to be placed in the remote storage area) has the largest correlation, it will be moved to selected set of variables.

Table 2-12: Correlation coefficient (4th iteration)

Correlation coefficient	$(r_{x_{i,ign}y})^2$
$r_{x_3y\cdot x_2x_4x_1}$	0.0469

Selected set$\{x_{i,sel}\} = \{x_2, x_4, x_1, x_3\}$

Ignored set$\{x_{i,ign}\} = \varphi$

Formulate the regression model (Equation 2-61).

$$y = 106.928 + 6.603x_2 + 9.0278x_4 - 3.776x_1 + 0.839x_3 \quad (2\text{-}61)$$

Perform partial *F*-test (Equations 2-62 to 2-65).

$$p_{partial,x_2} = f(37.66, m, n-k-1) = f(37.66,1,25) = 0.000 < 0.05 \quad (2\text{-}62)$$

$$p_{partial,x_4} = f(17.17, m, n-k-1) = f(17.17,1,25) = 0.000 < 0.05 \quad (2\text{-}63)$$

$$p_{partial,x_2} = f(15.73, m, n-k-1) = f(15.73,1,25) = 0.000 < 0.05 \quad (2\text{-}64)$$

$$p_{partial,x_3} = f(0.264, m, n-k-1) = f(15.73,1,25) = 0.612 > 0.05 \quad (2\text{-}65)$$

The p-value calculated for x_3 is higher than 0.05, therefore, the x_3 is insignificant and needs to be moved to the ignored set. The final MLR model remains the same as 3rd iteration (Equation 2-57), where y is the desired install hours (hour), $x_1$is the number of tractors, $x_2$is the delivery batch, and$x_4$is the duration to remote storage area.

2.5.1 Model verification

The verifications of BLUE assumptions are required for the final MLR model (Equation 2-57).

2.5.1.1 Checking heteroscedasticity

The Breusch-Pagan results are shown in Table 2-13. As the test *p*-value is larger than 0.05 (i.e., significant level), the variance of the error term is constant, thus, no heteroscedasticity is encountered.

Table 2-13: Breusch-Pagan test for OLS-based regression model

Variable	Degree of freedom	χ^2	*p*-value
x_1, x_2, x_4	26	36.052	0.4662

2.5.1.2 Checking multicollinearity

The VIF test of multicollinearity results are shown in Table 2-14. For all the input variables the VIF value is close to 1 (smaller than 10), the MLR model experiences no multicollinearity.

Table 2-14: VIF values for checking multicollinearity

Variable	VIF
x_1	1.10
x_2	1.10
x_4	1.00

2.5.1.3 Checking autocorrelation

Durbin-Watson test is conducted to calculate the *d*-statistic value. The result shows that the *d* value is 1.748, which is close to 2, therefore, autocorrelation does not exist between the error terms.

2.5.1.4 Checking normality of error

The normality assumption is critical when if the sample size is small (less than thirty recorded data). In the current case, the number of records in data set is thirty (30); thus, the normality of error terms needs to be tested. The Anderson-Darling test for normal distribution has a critical value of 0.7316 (sample size of 30 and percentage level of 0.05). In the current MLR model, the Anderson-Darling value of 0.722 is obtained, which is lower than the critical value suggested (refer to Equation 9). Therefore, the MLR error terms follow a normal distribution.

2.5.2 Cross-checking against trial-and-error approach

Table 2-15 summarizes all the potential MLR models, formulated by considering all the possible subsets of input variables (i.e. x_1, x_2, x_3, and x_4). The evaluation metrics of *RMSE*, R^2, and *Adjusted* R^2 are determined for each MLR model based on Equations (48), (67) and (68). The R^2 measures the correlation between predicted and observed dependent variables (Equation 2-67). The *Adjusted* R^2 is the modified form R^2 which considers the number of input variables in the regression model (Equation 2-68). Note, the *Adjusted* R^2 value would only increase if the added input variable improves the explanatory power of the model. The MLR equation with the least *RMSE* value and highest *Adjusted* R^2 is connected with the subset that includes x_1, x_2, and x_4. The result is identical to the one derived by applying the proposed framework (Equation 2-57). Thus, the proposed approach is cross-validated.

$$R^2 = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2} \tag{2-67}$$

where$y_i$is the observed dependent variable, $\hat{y}_i$ is predicted dependent variable, and $\bar{y}$ is the mean value of actual dependent variable.

$$Adjsuted\ R^2 = 1 - \frac{(1 - R^2)(n - 1)}{(n - k - 1)} \tag{2-68}$$

where n is the number of recorded data sets, and k is the number of predictors in the regression model.

Table 2-15: Regression model formulated by use of trial-and-error approach

Model identifier	Independent variables	Formulated regression model	RMSE	R^2	Adjusted R^2
1	Constant	110.9	5.15	-	-
2	x_1	$124.86 - 5.58x_1$	4.37	0.304	0.279
3	x_2	$100.88 + 7.92x_2$	3.78	0.479	0.460
4	x_3	$108.81 + 3.70x_3$	5.11	0.051	0.017
5	x_4	$105.99 + 9.70x_4$	4.70	0.197	0.168
6	x_1 x_2	$112.06 - 3.81\ x_1 + 6.62\ x_2$	3.34	0.608	0.579
7	x_1 x_3	$122.77 - 5.63\ x_1 + 3.90\ x_3$	4.27	0.360	0.313
8	x_1 x_4	$119.87 - 5.52\ x_1 + 9.54\ x_4$	3.80	0.494	0.457
9	x_2 x_3	$98.81 + 7.91\ x_2 + 3.65\ x_3$	3.67	0.528	0.493
10	x_2 x_4	$96.04 + 7.89\ x_2 + 9.61\ x_4$	3.06	0.672	0.648
11	x_3 x_4	$105.81 + 0.64\ x_3 + 9.32\ x_4$	4.78	0.198	0.139
12	x_1 x_2 x_3	$110.09 - 3.87\ x_1 + 6.58\ x_2 + 3.80\ x_3$	3.16	0.661	0.622
13	x_1 x_2 x_4	$107.11 - 3.76\ x_1 + 6.61\ x_2 + 9.52\ x_4$	2.45	0.797	0.774
14	x_1 x_3 x_4	$119.65 - 5.54\ x_1 + 0.96\ x_3 + 8.98\ x_4$	3.86	0.497	0.439
15	x_2 x_3 x_4	$95.87 + 7.89\ x_2 + 0.62\ x_3 + 9.24\ x_4$	3.11	0.673	0.635
16	x_1 x_2 x_3 x_4	$106.93 - 3.77\ x_1 + 6.60\ x_2 + 0.83\ x_3 + 9.02\ x_4$	2.48	0.799	0.767

2.5.3 Prediction error estimation

To estimate the error of point prediction with a significance level of 0.05, a data point is postulated for illustration. It has been assumed that: there are three trailer trucks rented for hauling segments; the segments are delivered in two batches (the 1st batch of seven segments would be delivered on the night before installation starts, and the 2nd batch delivered on the following night); the haul duration for a trailer truck to transit from the remote storage area to the working span is 45 minutes. As such, the number of trucks (x_1), the delivery batch (x_2), and the duration to remote storage area (x_4) are fixed as 3, 2, and 0.75, respectively. As per Equations (2-69) to (2-75), the range of the cycle-time is predicted between 113.699 hours and 118.679 hours. Note, the observed value for this point estimate is 116.74 (Appendix B, Table 18, date set 24) which lies in the predicted range.

$$[x_0] = [1 \quad x_1 \quad x_2 \quad x_4] = [1 \quad 3 \quad 2 \quad 0.75] \tag{2-69}$$

$$\hat{y}_0 = 107.107 + 6.610x_2 + 9.518x_4 - 3.759x_1 = 116.189 \tag{2-70}$$

$$\sigma_{res}^2 = \frac{SSE}{n-k-1} = \frac{155.9717}{26} = 5.999 \tag{2-71}$$

$$(X^T X)^{-1} = \begin{bmatrix} 1.7357 & -0.4318 & -0.3621 & -0.3234 \\ -0.4318 & 0.1467 & 0.0500 & 0.0035 \\ -0.3621 & 0.0500 & 0.1875 & -0.0008 \\ -0.3234 & 0.0035 & -0.0008 & 0.6216 \end{bmatrix} \tag{2-72}$$

$$s.e. = \sqrt{\sigma_{res}^2 [x_0](X^T X)^{-1}[x_0]^T} = \sqrt{5.999 \times 0.2446} = 1.211 \tag{2-73}$$

$$t_{(\alpha/2, n-k-1)} = t_{(0.025, 26)} = 2.056 \tag{2-74}$$

$$\hat{y}_0 \pm t_{(\alpha/2, n-k-1)} \times s.e. = 116.189 \pm 2.056 \times 1.211 = 116.189 \pm 2.49 \tag{2-75}$$

2.6 MSR Framework Against General Practice of MLR Modelling

The general practice of MLR modelling is to include all the identified input variables and calibrate the model. The OLS optimization method is applied to estimate the coefficients of independent variables. Equation (2-76) shows the OLS-based MLR model.

$$y = -88.525 + 0.010x_1 - 0.013x_2 + 0.006x_3 + 0.259x_4 - 0.184x_5 + 0.030x_6 + 0.039x_7 \quad (2\text{-}76)$$

2.6.1 Verification

2.6.2 Check heteroscedasticity

Table 2-16 shows the results of the Breusch-Pagan test. As the *p*-value is larger than 0.05 (i.e., assumed significant level), the variance of the error term is constant.

Table 2-16: Breusch-Pagan test for OLS-based regression model

Independent variable	χ^2	*p*-value
x_1	0.20	0.6545
x_2	0.59	0.4408
	0.13	0.7202
x_4	1.37	0.2416
x_5	0.07	0.7942
x_6	0.83	0.3627
x_7	0.12	0.7280
Fitted values of y	2.50	0.1142

2.6.3 Check multicollinearity

Except x_5, all the variables have VIF higher than 10 (Table 2-17). The result shows that the multicollinearity exists in the formulated regression model.

Table 2-17: VIF values

Independent variable	VIF
x_6	88.17
x_3	58.65
x_2	55.28
x_7	49.96
x_1	48.57
x_4	31.43
x_5	2.14

2.6.4 Check autocorrelation

The d value is 1.806, which is closed to 2, therefore, no autocorrelation is experienced in MLR.

2.6.5 Check normality of error variance

The assumption is critical when if the sample size is small. In the current scenario, the number of data set is larger than 100 and there is no need to check the normality of error terms.

2.6.6 Model validation

To validate the regression model for prediction, both K-fold cross-validation method and *PRESS* statistics have been tested. In this research study, k is assumed to be ten for applying the k-fold method. The resulting *SSE* values for each test are given in Table 2-18. The total *SSE* and the *PRESS* values are calculated by Equation (2-77) and Equation (2-78), respectively. In both tests,

the resulting *SSE* values are compared with the OLS-based regression model *SSE* (Equation 2-79) and (Equation 2-80). This model can be validated if the *SSE* ratios are both in the acceptable range (10%-15%). Using both the K-fold and *PRESS* validation methods, it can be seen that the *SSE* ratios are slightly higher than the acceptable range. As a result, this regression model might be over-fitted to the data.

Table 2-18: SSE values for k-fold cross-validation

Test	1	2	3	4	5	6	7	8	9	10
SSE	1214.0	1249.7	793.6	369.2	185.2	623.2	436.9	345.8	321.9	703.3

$$SSE_{k-fold} = \sum_{i=1}^{k} Test_i(SSE) = 6242.8 \tag{2-77}$$

$$PRESS = SS(PRESS\ \text{residuals}) = \sum_{i=1}^{n} e_i^{*2} = 79.36 + 5.69 + \ldots + 27.83 = 6253.5 \tag{2-78}$$

$$\frac{SSE_{k-fold}}{SSE_{OLS\ Model}} = \frac{6242.8}{5285.6} = 1.181 \tag{2-79}$$

$$\frac{PRESS}{SSE_{OLS\ Model}} = \frac{6253.5}{5285.6} = 1.183 \tag{2-80}$$

Contrasting the results with proposed regression model

The developed model which includes all the input variables (Equation 2-76) compared to the proposed regression model (Equation 2-36) has several drawbacks. Firstly, Equation (2-76) requires seven input variables compared to two inputs of Equation (2-36). Having more input variables would result in more effort in data collection, and a more complicated application. Secondly, the high multicollinearity between the input variables would lead to the violation of BLUE assumptions and reduce the reliability of the model. Multicollinearity coupled with the validation results, further supports the claim that the model has redundant input variables. On the other hand, Equation (2-36) reduces the chance of over-fitting which would lead to more reliable estimates.

2.7 CONCLUSION

Regression analysis results in simple equations that sufficiently represent real-world systems in civil engineering. Regression methods can be applied to tackle conventional "historical data" as well as emerging "big data" problems. Regression has not been able to catch up with rapid technology advances and practical application needs. The real-world problems can be mind-boggling, and the data often contain noises or missing information. On the other hand, the problem-solving methods are expected to be computationally simple, fast to calibrate, straight-forward to explain, and easy to update as new data become available. To be acceptable and truly appealing to practical applications, user experiences of data-based, analytics-driven decision-support systems in civil engineering must not be perceived as tapping "black box" or requiring too much "trial-and-error".

This Chapter formalizes a generic framework for generating MLR models consisting of variable selection, model verification, model validation, and prediction error estimation. A refined version of stepwise regression is implemented for variable selection; if any of the OLS (ordinary least square) assumptions are violated, the WLS is used for estimating the MLR coefficients. The proposed framework is illustrated and tested on two case studies. The UCI machine learning dataset is widely used for demonstrating calculation procedures and comparison with related non-linear regression models. Previous studies on estimating concrete slump have produced ANN and non-linear regression models for the same dataset with cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, and fine aggregate as model inputs. The non-linear model includes seven inputs and needs fifty-six regression coefficients to be fixed. The ANN model requires fifty-six transformation weights to be fixed. On the contrary, the MLR model resulting

from the proposed framework requires only two input variables (blast furnace slag and water). Its prediction performance is better than the non-linear regression model which was developed in the previous research. Although the ANN model marginally outperformed the MLR model in terms of smaller errors on point-value predictions, the streamlined MLR model was capable of analytically determining a range around the point-value. The practical case study demonstrates the advantages of the proposed framework over the trial-and-error approach in variable selection. With four input variables, the trial-and-error approach would require building and comparing sixteen MLR models. The proposed MLR framework and the trial-and-error approach resulted in the identical model as the final solution; however, the proposed framework is analytical, much simpler and scalable.

It is worth mentioning that both case studies used in this research feature a relatively small number of input variables (the concrete slump dataset has seven inputs, while the precast bridge dataset has four). With a small number of input variables, established factor selection methods (i.e. forward selection, or backward elimination) and the modified stepwise method would end up with similar regression models. Details on a more practical application with larger dataset in connection with many-input-variable applications is given in the next Chapter.

CHAPTER 3 : LABOUR-HOUR PREDICTION MODELS FACTORING IN PROJECT DESIGN FEATURES

This chapter describes a Modified Stepwise Regression application in more practical setting of labour productivity prediction of steel fabrication. This Chapter proposes a data-driven approach that uses Multiple Linear Regression (MLR) and available historical data from Building Information Models (BIM) to associate project labour-hours and project design features. The proposed framework in this chapter also encompasses analytical methods for validating MSR model and establishing range estimates for point-value predictions.

3.1 INTRODUCTION

On labour-driven construction tasks, labour cost is measured by labour-hours and then converted into dollars by multiplying crew specific hourly rates (Alfeld, 1988). Relying on experiences and "gut feels" to estimate labour-hours on steel fabrication work packages has been recognized as one of the main factors that account for insufficient cost budgeting in planning and eventually lead up to budget overrun in project execution (Sawada et al. 2006; Song and AbouRizk 2006). In construction engineering, confident prediction of labour resource requirements based on project design information is vitally important for cost estimating, planning and controlling construction projects. Structural engineers also need a reliable assessment of construction cost implication in evaluating alternative project designs (Wiesenberger 2011). Recent advances in information technology, including the adoption of building information Modelling (BIM), and bar-coding for resource tracking, has gradually improved project data availability (Shen and Issa 2010) while also

presenting a great opportunity for improving construction productivity and efficiency (Walasek and Barszcz 2017). Monteiro and Poças Martins (2013) demonstrated the possibility of extracting quantities from a BIM model and created a model ready for visualization for estimators and planners. Plebankiewicz et al. (2015) investigated BIM-based cost estimating systems and found BIM could generate accurate quantity take-offs for project cost estimating. Taghaddos et al. (2016) proposed an automated system to perform material take-offs from BIM models and pointed the need in further research in this area by analyzing productivity data and deriving labour-hours based on estimated volumes and weights from the model. In recent years, the number of steel fabricators which implement the BIM technology in creating fabrication models and developing detailed shop drawings has been steadily growing. Adoption of BIM on construction projects opens new gateways to data collection and presents new possibilities for data analysis (Taghados 2015). Nevertheless, an analytical framework has yet to be formalized for selecting proper prediction factors and establishing valid prediction models in correlating project design features with required labour-hours in practical applications. The present research is intended to address the following issues identified in the current practice:

1. Lack of integration between labour cost tracking systems and project estimating and planning systems makes it difficult to validate and implement developed productivity models.
2. The set of inputs factored in the majority of labour cost predictive models is generally insufficient due to data availability constraints.
3. The missing connection with design features in most of the established labour cost prediction models does not allow for a straight-forward evaluation of various design alternatives in terms of project cost performance.

Multiple Linear Regression (MLR) is widely implemented as an effective technique for developing data-driven prediction models on a variety of engineering applications (Jafarzadeh et al. 2015). MLR is a form of regression analysis where two or more input variables are used to predict an output variable. Selected examples of MLR applications in the literature include predicting building construction cost (Lowe et al. 2006), assessing the service condition of pipelines (El-abbasy et al. 2014), and planning seismic retrofit construction (Jafarzadeh et al. 2015). Before implementing more complicated nonlinear methods such as Artificial Neural Networks (ANN), it is worthwhile to take MLR as the foundation methodology and take full advantage of MLR in gaining insight into the problem definition and the available data (Jafarzadeh et al. 2015; Verlinden et al. 2008; Siu et al. 2014).

Frequently used methods for variable selection in MLR include correlation analysis and trial-and-error, which are time-consuming, tedious, and often end up with sub-optimal and case-dependent solutions (Draper and Smith 1998). This research proposes a new analytical framework to determine the best achievable prediction accuracy in applying MLR and maintain simplicity in tackling the complexity inherent in real-world problems. As such, the minimal set of inputs can be identified in creation of an MLR model in its leanest form without compromising the maximum prediction accuracy achievable. The framework is proposed for the practical context of structural steel fabrication by using historical data from BIM and project labour costing systems. Before presenting the framework and the implementation case, background of structural steel fabrication is first given.

3.2 STRUCTURAL STEEL FABRICATION

Steel is one of the primary materials used for providing structural stability for residential and commercial buildings, industrial plants such as oil and gas pipe racks, and infrastructure projects such as bridge girders (Warrian 2010). Fabrication of structural steel elements brings significant quality and productivity benefits in construction (Liddy and Cross 2002). Steel fabrication is characterized by labour-intensive work processes on bespoke project designs. Specialized trades of labourers perform a variety of operations such as handling, cutting, fitting, welding, and surface processing (e.g., sandblasting and painting). Structural steel fabrication is generally estimated based on the weight of a steel project, ignoring project complexity and design details (Sawada et al. 2006). Song and AbouRizk (2003) introduced a simulation model for a steel fabrication shop, which represented the flows of steel pieces and resources on the fabrication shop floor; the simulation model was further developed into a hybrid model utilizing ANN and operations simulation in an attempt to predict labour costs on structural steel fabrication. O'Neil and Rozmarin (2010) created a Monte Carlo simulation model to estimate labour costs in bridge steel fabrication with the purpose of assessing the effects of change orders.

The various levels of granularity for data collection in the structural steel industry are generalized in Figure 3-1. The finest level is the material level, where raw materials (e.g. steel plates) are purchased from different suppliers and transferred to the fabrication shops for further processing. The raw materials are then processed through cutting, drilling, and welding to create a piece – that is a structural element as per design specifications. As an example, a beam fabricated out of wide flange sections with two end plates is classified as a piece. Establishing the work breakdown structure of a project according to the construction sequence results in the definition of "Division".

A division of a steel project is defined with fabrication processes (e.g., handling, cutting, etc.), key engineering design parameters (e.g., section dimensions) and detailed design features (e.g., welds, bolts etc.). Several structural steel pieces that are part of an erection phase and need to be ready at the construction site prior to field erection constitute a division. Finally, a project is one division or a combination of a limited number of interrelated divisions.

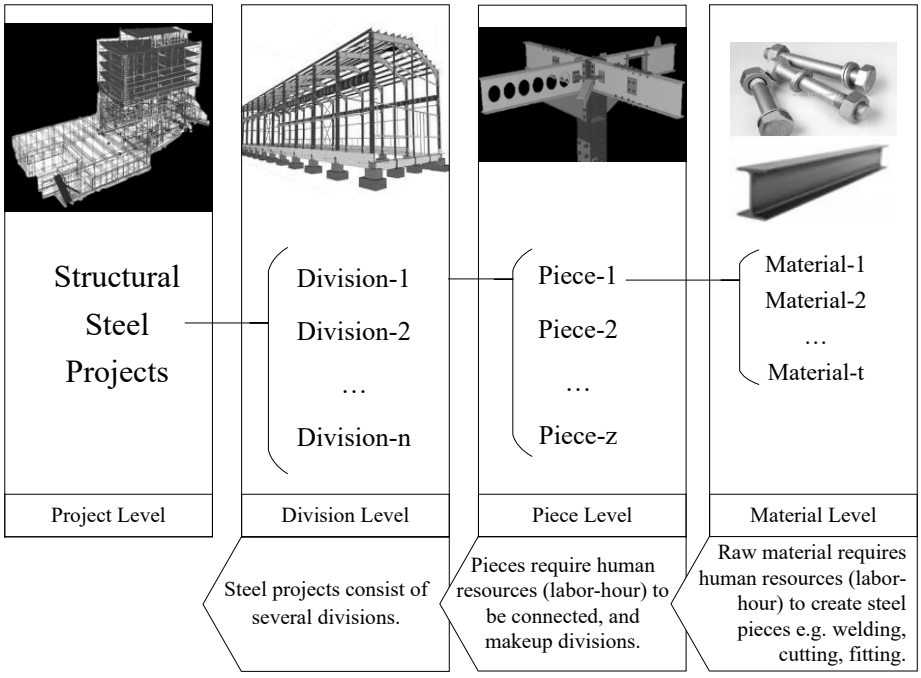

Figure 3-1: Steel fabrication project's scope structure

In the current practice of project management, the division-level is the finest level of granularity for keeping track of actual labour-hours spent in job costing and project control systems (Hu et al. 2014). It is also worth mentioning that the granularity of data captured in a company's job costing system (e.g. labour-hours by division in the case of structural steel fabrication) is generally dictated by contractual requirements. For example, a fabricator collects division-level data that are sufficient to generate project earned value reports required by clients. On the other hand, it is prohibitively expensive and hence not practically feasible to track labour-hours on work packages

at the piece level or the material level on the shop floor. Hence, this study sets focus on the characterization of the complexity of division considering engineering design features; in such a way, historical data accumulated in databases underlying BIM models for those structural steel projects completed over the past decade can be taken advantage of to its fullest.

3.3 PROPOSED FRAMEWORK AND IMPLEMENTATION

This research proposes an application framework that uses MLR as the analytical methodology to associate project labour-hours and project design features. MLR provides a predictive model that is one of the simplest forms of predictive analytics; however, it represents a transparent and straight-forward mathematical equation generalized from data, while also enabling insightful evaluation of quality of the available data and significance of input factors. It is noteworthy that if the quality of data is insufficient, direct application of more complex analytics might cover up noises in the data, potentially resulting in unrealistic/over-fitted models. Given a dataset representing a certain scope of structural steel fabrication, this framework is simple and effective to select the most relevant predictive factors in the creation of a streamlined predictive model assisting in the determination of required labour-hours.

In the nutshell, the framework relies on the application of an enhanced version of the stepwise regression technique to select the most relevant predictive factors and generates a predictive model without compromising the achievable accuracy of regression. The complete framework incorporates proven analytical and statistical methods in support of enabling MLR application, validating the resulting model, and establishing range estimates for point-value predictions. This framework is best suited to real-world application scenarios (1) where a large number of input parameters are present and the historical data are likely to contain noises (incomplete or inaccurate records); (2) where there is a need to develop a quantitatively reliable, statistically significant predictive model in the leanest, simplest form, which features the most important input factors and tolerates noise in data to a certain degree, but is not over-fitted with noise. The roadmap for guiding the implementation of the proposed framework is shown in Figure 3-2. This framework

has been implemented specifically for streamlining design features in the prediction of labour-hours required for a division in structural steel fabrication, which is to be elaborated in the remainder of this Chapter.

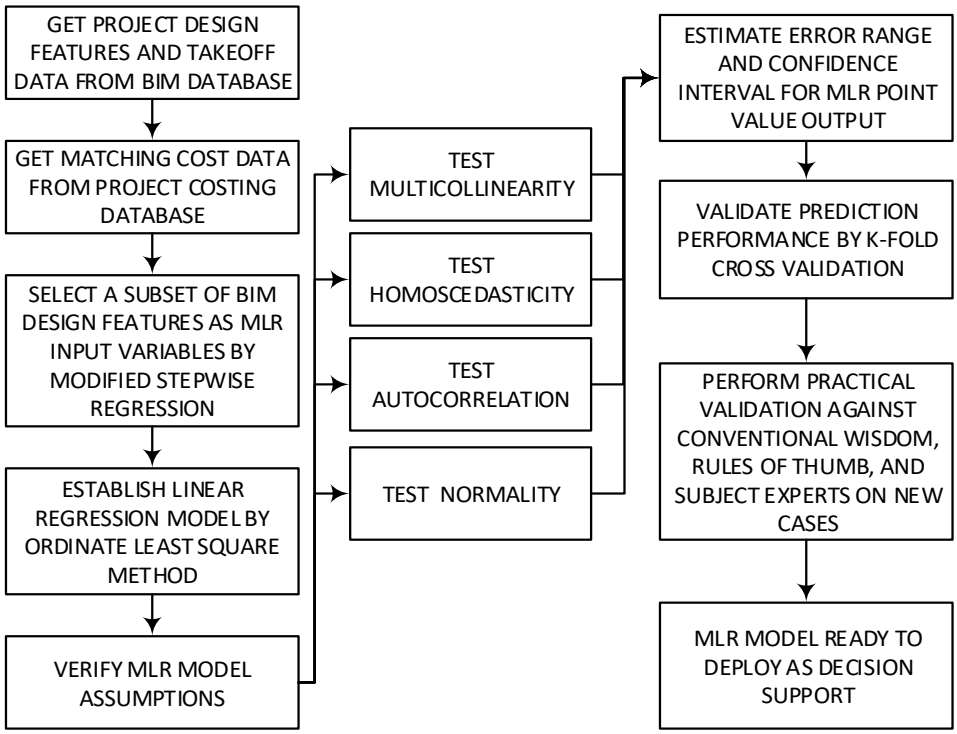

Figure 3-2: Proposed framework application

3.3.1 Input variable selection

Using fewer input variables is favourable from the practical point of view as it minimizes data collection efforts in the long run. A regression model created with a smaller subset of the identified input features, instead of all, also has advantages in statistical modelling. Minimizing the number of input variables significantly reduces the likelihood of over-fitting, collinearity (high correlation between input variables), and transferring noise from data to the calibrated model (Ivanescu et al. 2016). Having too many input variables, the regression model tends to fit itself to the noise hidden

in the training set instead of generalizing underlying patterns and hidden relationships. A proper method for variable selection removes those insignificant or redundant input variables from the regression model (Akinwande et al. 2015). To elaborate more on the variable selection, Figure 3-3 (a) represents the output variable Y, and input variables X_1, X_2, X_3, and X_4. Each variable accounts for part of the output variable Y (Figure 3-3 (b)). Figure 3-3 (c) and (d) depicts the explanatory power of the group of input variables in predicting Y. Although X_2 and X_3 individually have high explanatory power, by removing them, the explanatory strength of the streamlined group of input variables (X_1 and X_4) would not be compromised, as illustrated in Figure 3-3 (d).

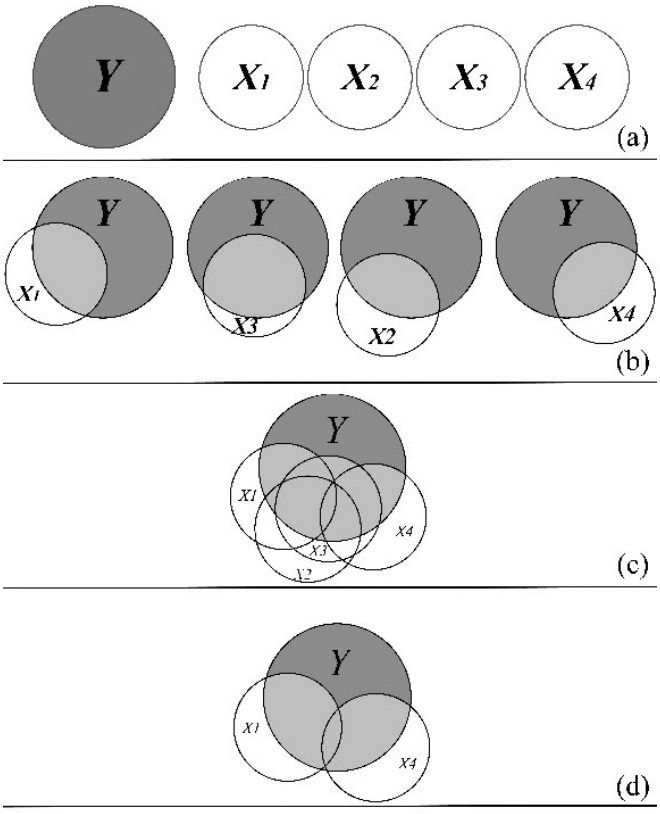

Figure 3-3: Variable selection concept: (a) inputs (X1, X2, X3, & X4) and output (Y), (b) explanatory power of individual inputs, (c) explanatory power of all the inputs combined, (d) explanatory power of selected variables.

At the core of the proposed MLR modelling framework, a variable selection technique called Modified Stepwise Regression (MSR) –which has been developed in authors' previous computing research (Mohsenijam et al. 2016)- is utilized for identifying only those design features relevant to the prediction of steel fabrication labour-hours. This step is presented in Figure 3-2 "Selecting a Subset of BIM Design Features as MLR Input Variables". Note, once the subset is selected in such an analytical way, adding more variables would not enhance the performance of the MLR model in terms of accuracy.

3.3.2 Data Preparation

Databases associated with BIM in our partner company contains forty-two project-related design features (columns) and 1559records (rows), each record representing a project division for fabrication, listed in Table 3-2. Labour-hours spent in fabricating each division are extracted from job costing databases. Note that labour-hours collected for each project division account for total labour-hours spent in fabrication including handling, cutting, fitting, welding, and surface processing. The collected data mainly represent project divisions in the industrial sector, with labour-hours ranging from 100 to 7000. It is worth mentioning that in the current research, the actual dataset was linearly scaled prior to performing analysis in order not to reveal the company's sensitive productivity information while keeping the original patterns and relationships inherent in the data. Sample of raw data used for training in this research is given in

Table 3-1. The dataset used for this Chapter can be accessed at https://figshare.com/s/8de57c3a0ca8f8ed37c4. Note that the collected data represent a steel fabrication in northern Alberta; however, methods proposed in this research are generic and replicatable to other settings.

Although the variables in Table 3-2 are all considered relevant in predicting labour-hours, a certain interdependency and redundancy exist between different variables and some can be explained by others. For instance, material length and weight are highly correlated; by knowing one, the other can be deduced. Next, a variable selection technique is applied to streamline the input data.

Table 3-1: Sample of raw data used for training

Input variables	Description	Unit	Sample 1	Sample 2
X1	Division Weight	Kg	56746	31660
X2	Angle	Meter	88.271	146.34
X9	Wide flange	Meter	246.155	438.04
X14	Plate	Meter-squared	31.981	99.52
X18	Hollow steel sections	Meter	194.325	0
X32	Partial penetration weld	Meter	98.318	32.332
Y	Actual Labour-hours	Hours	1590	912

Table 3-2: Design Features Extracted from The BIM Databases

BIM Extracted Design features	Unit	Category	Labels
Division Weight	Weight (kg)	Material-Weight	***X1***
Angles	Length (m)	Material-Steel Sections	***X2***
Channels	Length (m)	Material-Steel Sections	***X3***
I Beams	Length (m)	Material-Steel Sections	***X4***
Miscellaneous beams	Length (m)	Material-Steel Sections	***X5***
Miscellaneous channels	Length (m)	Material-Steel Sections	***X6***
Structural Tees from W Shapes	Length (m)	Material-Steel Sections	***X7***
Tarpon Z Sections	Length (m)	Material-Steel Sections	***X8***
Wide flange	Length (m)	Material-Steel Sections	***X9***
Crane rails	Length (m)	Material-Steel Sections	***X10***
Bent plate	Area (m^2)	Material-Plate	***X11***
Checker plate	Area (m^2)	Material-Plate	***X12***
Grating	Area (m^2)	Material-Plate	***X13***
Plate	Area (m^2)	Material-Plate	***X14***
Extra Extra Strong Pipe	Length (m)	Material-Pipes	***X15***
Extra Strong Pipe	Length (m)	Material-Pipes	***X16***
Standard Pipe	Length (m)	Material-Pipes	***X17***
Hollow steel sections	Length (m)	Material-Hollow Sections	***X18***
Round hollow steel sections	Length (m)	Material-Hollow Sections	***X19***
Cold formed channels	Length (m)	Material-Cold-formed	***X20***
Tarpon Cold Formed Channels	Length (m)	Material-Cold-formed	***X21***
Flat bar	Length (m)	Material-Bars	***X22***
Rebar	Length (m)	Material-Bars	***X23***
Round bar	Length (m)	Material-Bars	***X24***
Square bar	Length (m)	Material-Bars	***X25***
Hex Bar	Length (m)	Material-Bars	***X26***
Expansion Anchor Bolts	Quantity	Material-Anchors	***X27***
Heavy Duty Expansion Anchor Bolts	Quantity	Material-Anchors	***X28***
Threaded Anchor Rods	Quantity	Material-Anchors	***X29***
Adhesive Anchor Cartridges	Quantity	Material-Anchors	***X30***
Complete penetration weld	Length (m)	Connection-Welding	***X31***
Partial Penetration Weld	Length (m)	Connection-Welding	***X32***
Bevelled Washers	Quantity	Connection-Bolted	***X33***
Button Head Machine Bolt	Quantity	Connection-Bolted	***X34***
Compressible Washers with DTI	Quantity	Connection-Bolted	***X35***
Flat Washers	Quantity	Connection-Bolted	***X36***
Hex Head Machine Bolt	Quantity	Connection-Bolted	***X37***
Hex Nuts	Quantity	Connection-Bolted	***X38***
Hex Type Bolts	Quantity	Connection-Bolted	***X39***
M Type Bolts	Quantity	Connection-Bolted	***X40***
Mechanical Pipes	Length (m)	Material-Pipe	***X41***
Nelson Studs	Quantity	Connection-Stud	***X42***

3.4 INPUT VARIABLE SELECTION

For an MLR model with *n* predictors and one output, there are 2^n subsets of variables that can be correlated with the output. In the current case, the original number of division design features in BIM is forty-two, thus making it practically infeasible to examine all the possible subsets of input variables. Commonly-applied stepwise methods for variable selection in the literature in connection with regression analysis can be classified into *forward selection* vs. *backward selection*.

The *forward selection* technique starts with a linear regression model that contains no input variables and adds variables to the regression model based on correlation and statistical significance (Seber and Lee 2003). On the other hand, *backward elimination* begins with the full set of variables and iteratively removes insignificant variables to reach the final subset (Wang and Jain 2003). However, a selected variable in forward selection is never removed in later iterations; likewise, a variable cannot be reintroduced once it has been eliminated in the backward elimination process. As such, both forward selection and backward elimination methods tend to produce a near-optimum subset of variables (Draper and Smith 1998). Given large datasets, applying forward selection and backward elimination separately on a set of input variables often leads to two different subsets of selected variables (Thompson 1978, Mendenhall and Sincich 2015). By combining the advantages from both forward selection and backward elimination while overcoming respective limitations of each, a modified version of stepwise regression is developed (Mohsenijam et al. 2016), which is briefly explained below:

The MSR starts with an empty set of selected variables and a full set of ignored variables (same as forward selection). In each iteration, a variable with the highest partial correlation (Equation 3-1) is selected from ignored variables; all the selected variables are tested with partial F-test (Equation 3-3) for statistical significance (Figure 3-4 and Figure 3-5). Note that the partial correlation quantifies the explanatory power of the ignored variables, which is not yet accounted by the selected variables.

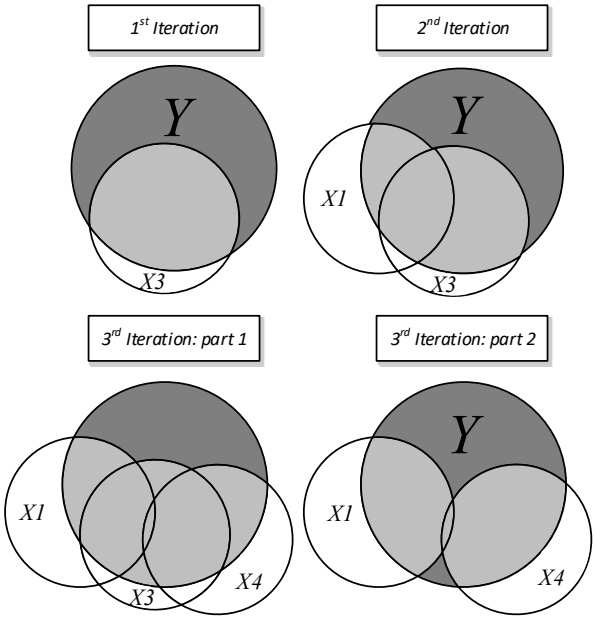

Figure 3-4: Modified stepwise regression visual representation

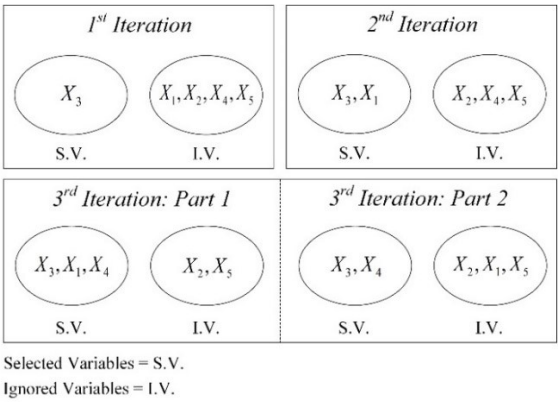

Figure 3-5: Modified stepwise regression iterations

$$r_{X_1Y.X_k} = \frac{r_{X_1Y} - r_{X_1X_k}r_{YX_k}}{\sqrt{(1 - r_{X_1X_k}^2)(1 - r_{YX_k}^2)}} \tag{4-1}$$

Where X_1 is the variable that is selected in previous iterations, $X_k$is the variable for which the partial correlation is measured against, $Y$is the independent variable, $r_{ab}$is the correlation between a and b which can be obtained from Equation (3-2).

$$r_{ab} = \frac{n\sum ab - \sum a \sum b}{\sqrt{[n\sum a^2 - (\sum a)^2][n\sum b^2 - (\sum b)^2]}} \tag{3-2}$$

Where r_{ab} measures the correlation between a and b, and n is the size of the dataset.

$$F_{x_i} = \frac{(SSE_{k-1,X_i} - SSE_k)}{(SSE_k)/(n-k-1)} \tag{3-3}$$

Where $SSE_k$is the standard error of the regression with k variables calculated by Equation (3-4), and $SSE_{k-1,X_i}$is the standard error of the regression before adding $X_i$to the model, k is the number of variables and n is the number of observations.

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3-4}$$

Where$y_i$is the dependent variable and$\hat{y}_i$ is the predicted value of the MLR model.

For example, given a case with five input variables (demonstrated in Figure 3-4 and Figure 3-5), in the first iteration, X_3 is selected, since it has the highest correlation with Y. In the second

iteration, by determining the partial correlation X_1 is selected while both variables are identified to be significant by F-test. In the third iteration, X_4 is selected, but by performing F-test it is decided that removing X_3 has an insignificant effect on the regression model's performance. The Modified Stepwise Regression would allow previously eliminated variables to be reinstated in the regression model; while those already selected variables can be removed in future iterations. More details on the analytical procedures of the Modified Stepwise Regression can be found in Mohsenijam et al. (2016) , along with two applications featuring relatively small numbers of input variables (four and seven input variables respectively), which are in contrast to a much larger dataset in the current research with forty-two input variables to start with.

MATLAB 2016b (MathWorks 2016) was used for creating MLR models, performing MSR, and testing regression assumptions in this research. Applying MSR on the given set of input variables (Table 3-3) resulted in the selection of six variables out of forty-two as shown in Table 3-3 under "Model 6". To demonstrate the fact that adding more variables would not enhance the performance of the predictive model, Models 7 and 8 are also given in Table 3. It is observed the R-square value remains steady at 0.680 in Models 6, 7 and 8. The six chosen input variables are (1) Division weight (X1, in kg), (2) Square hollow steel sections (X18, in meters), (3) Plate (X14, in square meters), (4) Hex type bolts (X39, quantity), (5) Complete penetration weld (X31, in meters), and (6) Partial penetration weld (X32, in meters).

Table 3-3: Variables entered the MLR model and their effect on model performance

Model	Variables in the MLR Model	R-Square	Adjusted R-Square
1	(Constant), X1	0.476	0.475
2	(Constant), X1, X18	0.666	0.664
3	(Constant), X1, X18, X14	0.671	0.670
4	(Constant), X1, X18, X14, X39	0.675	0.674
5	(Constant), X1, X18, X14, X39, X31	0.677	0.676
6	**(Constant), X1, X18, X14, X39, X31, X32**	0.680	0.678
7	(Constant), X1, X18, X14, X39, X31, X32, X5	0.680	0.676
8	(Constant), X1, X18, X14, X39, X31, X32, X5, X12	0.680	0.675

It is noteworthy the resulting linear regression model (Model 6) given in Equation (3-5) is calibrated by applying the Ordinary Least Square (OLS) coefficient estimation method. The regression coefficients and statistical significance (p-value) for each variable are given in Table 3-4. All the variables selected in Model 6 have a p-value smaller than 0.05 (listed in Table 3-4) and thus are considered significant. Among the selected design features, “Division weight” (i.e., X1) and “Complete penetration welds” (i.e., X31) are identified to be more significant in predicting division-specific labour-hours (with smaller p-values). In addition, coefficients of all the variables in Equation (3-5) are positive, which implies that given more weight of a division, or more welding work required to fabricate the division, it would take more labour-hours to fabricate the division.

It is noteworthy the resulting linear regression model (Model 6) given in Equation (3-5) is calibrated by applying the OLS coefficient estimation method. The regression coefficients and statistical significance (p-value) for each variable are given in Table 3-3. All the variables selected in Model 6 have a p-value smaller than 0.05 (listed in Table 3-3) and thus are considered significant. Among the selected design features, “Division weight” (i.e., X1) and “Complete

penetration welds" (i.e., X33) are identified to be more significant in predicting division-specific labour-hours, as indicated by their smaller p-values. In addition, coefficients of all the variables in Equation (3-5) are positive, which implies that given more weight of a division, or more welding work required to fabricate the division, it would take more labour-hours.

$$Y = 47.828 + 0.0037X1 + 0.1495X18 + 0.0245X14 + 0.0126X39 + 2.6064X31 + 0.3328X32 \quad (3\text{-}5)$$

Table 3-4: MLR variables, OLS Coefficients, and p-values

Selected variables	Coefficient	p-value
Intercept	47.8283	6.81×10^{-4}
X1	0.0037	3.24×10^{-134}
X21	0.1495	1.34×10^{-5}
X15	0.0245	3.76×10^{-3}
X41	0.0126	1.60×10^{-5}
X33	2.6064	9.79×10^{-158}
X34	0.3328	4.73×10^{-2}

Results from variable selection in the current case are found to well align with the conventional wisdom and the rules of thumb applied in the current practice of steel fabrication. This is further elaborated as follows: Division weight is the most commonly used parameter for estimating steel fabrication cost (Sawada et al. 2006). However, weight does not describe project complexity in full, and hence it is necessary to consider other factors to improve estimating accuracy. The fact that the factor "*square hollow steel sections*" is identified as a significant variable is attributed to the significant welding work required to connect them, thus considerably increasing welding hours (Figure 3-7).

In steel fabrication, plates are used for stiffening steel sections (i.e., beams and columns) and creating connections between structural elements. The high plate volume in a division indicates a significant amount of cutting and welding in fabrication (Figure 3-6). Besides welding, bolted connections are common practice for splicing steel pieces. Bolted connections are used to permanently connect fabricated components in the construction site. The higher the number of bolts indicates more efforts of drilling holes and handling pieces in steel fabrication. Groove welding is a method of permanently connecting steel pieces, which can be categorized into two types, namely: complete penetration welds vs. partial penetration welds. Both types of groove welding are labour-intensive. As the name implies, the complete penetration welds are thicker than the partial penetration welds, thus requiring more welding passes to be performed. The differences in terms of labour-hours required for each type of groove welding are clearly characterized by the sign and magnitude of the coefficients in connection with respective factors in the MLR model (Table 3-5). Complete penetration welds require much more labour-hour per unit of length than partial penetration welds.

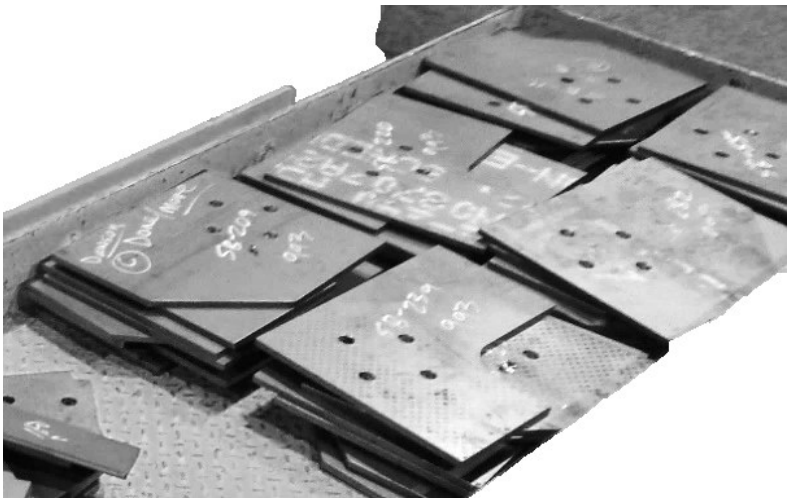

Figure 3-6: Cut plates with drilled holes ready to be welded to steel sections

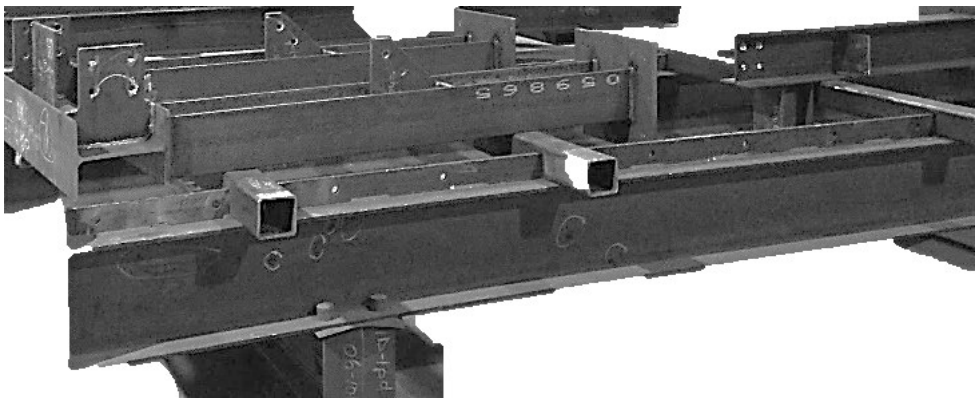

Figure 3-7: Steel sections with end plates attached to square hollow sections

3.5 VERIFICATION AND VALIDATION

As one of the key steps in the proposed framework, the generated model needs to be verified and validated to ensure the reliability and accuracy of its predictions. The verification process tests the underpinning theoretical assumptions of the model while also ensuring that the model's representation of the problem is correct for the intended purpose of use (Sargent 2013). Model validation is concerned with the fact that the model achieves a consistent level of accuracy in its application domain. Since the proposed framework utilizes MLR in model generation, the model needs to first pass desired regression performance measures (i.e. RMSE or R-squared) and satisfy all the theoretical assumptions in applying MLR. Once the model passes the verification stage, methods of historical data validation, range estimation (confidence intervals) and face validation are used to ensure that model has a sufficient accuracy in its intended application domain (Sargent 2013). The available historical data are generally utilized to build a model and validate it. The historical data validation methods proposed in this framework is *K-fold* validation and the results are given in the following sections. Further, range estimation provides a confidence interval around the point prediction with statistical significance; depending on where in the problem domain the specific point prediction falls, the resulting confidence interval would differ. For the purpose of face validation, two unseen cases were presented to both domain experts and the developed MLR model for cross-checking prediction accuracy.

3.5.1 MLR Model Assumptions Testing

The high correlation between input variables is called *multicollinearity*, which is tested by examining the correlation between two input variables. Variance inflation factor (VIF) is a commonly accepted indicator of multicollinearity (Kutner et al. 2004). For the derived regression

model given in Equation (3-5), the VIF values were computed and shown in Table 3-5. All the input variables are associated with VIF values less than three. This indicates no significant multicollinearity in the MLR model, which also validates the effectiveness of MSR in selecting relevant input factors.

Table 3-5: VIF multicollinearity test results

Variable	VIF Value
X1	2.84
X18	1.05
X14	1.05
X39	1.62
X31	1.01
X32	2.13

The second assumption that needs to be validated in MLR modelling is the constant variance of errors. *Heteroscedasticity* occurs when the variance of the errors shows varying patterns with different observations. Heteroscedasticity can be examined by using White, Goldfeld-Quandt test or Breusch-Pagan test (Kaufman 2013). In the presence of heteroscedasticity, coefficients estimated by OLS along with regression error analysis become biased and unreliable. In this research, the Breusch-Pagan test was used to check the null hypothesis of constant variance on error terms, with results shown in Table 3-6. The Breusch-Pagan p-value was calculated as 0.0001 based on Chi-square of 109.94, which was smaller than the significant level of 0.05, thus confirming the presence of heteroscedasticity in the developed model in the current case.

To eliminate heteroscedastic errors, the Weighted Least Square (WLS) method was applied to scale input variables linearly and remove any dependence of regression output error on the regression output (i.e., *Y*). It is noteworthy that linear scaling on variables would not affect

correlations between them, and therefore, would not influence the outcome of variable selection (Greene 2008). The MLR model consisting of the same input variables as in Equation 3-5 was calibrated by WLS, resulting in a new set of coefficients as shown in Equation (3-6) and Table 3-6. The Breusch-Pagan test was conducted again to verify the effectiveness of the WLS method in eliminating heteroscedasticity, resulting in Chi-square of 3.59 and p-value of 0.0554. As the Breusch-Pagan p-value is larger than 0.05, the variance of the error terms is deemed constant; thus, the regression output errors from MLR are deemed homoscedastic.

$$Y = 18.701 + 0.0042X1 + 0.1947X18 + 0.0853X14 + 0.0008X39 + 1.7968X31 + 0.1990X32 \quad (3\text{-}6)$$

Table 3-6: MLR variables, WLS coefficients, and p-values

Selected variables	Coefficient	p-value
Intercept	18.7014	1.08×10^{-3}
X1	0.0042	4.36×10^{-77}
X18	0.1947	1.29×10^{-4}
X14	0.0853	2.19×10^{-8}
X39	0.0008	8.10×10^{-3}
X31	1.7968	5.04×10^{-15}
X32	0.1990	8.27×10^{-5}

The third assumption underlying MLR states that residuals from a linear regression of different observations should be independent. *Autocorrelation* occurs if patterns in output errors are recognized between serial observations. Durbin-Watson test is a common method for testing autocorrelation, which calculates the d-statistic value by Equation (3-7) (Durbin and Watson 1950). Autocorrelation exists if the d-statistic is close to 4 or 0. A more systematic approach is to

select a significance level (i.e., 0.05 in this research study) and look up the Durbin-Watson Table of critical values (Durbin and Watson 1950) in order to determine the two values of dU and dL. The result shows that the d-statistic value is 1.128 in the current case, which is smaller than dL. Hence, autocorrelation is not identified in this regression model.

$$\text{Durbin-Watson d-statistic } (6, 1558) = 1.128 < dL(6,1558)=1.85 \quad (3\text{-}7)$$

In addition, *normality* of the residuals can be checked statistically by conducting the Anderson-Darling test (Stephens 1974). The hypothesis of normality is rejected if Anderson's statistic value (a-stat) exceeds a given critical value at a presumed significance level [Stephens (1974) provides such critical values]. The normality assumption is critical when the sample size is small. In the current case, the number of data points (1500 records in BIM database) is much larger than 100; hence checking normality in error terms is ignored (Stephens 1987).

3.5.2 K-Fold Cross-Validation

The K-fold cross-validation splits the available modelling data into a training set and a test set for k times and evaluates model performance only based on the test set. In the k-fold cross-validation, sum of squared errors (SSE) for respective segments is compared with the main model's SSE value (Table 3-7). If the two SSE values are close to each other (Equation 3-8) (e.g., within 10%-15% difference), the MLR model is validated (Sargent 2013). In this case study, the ratio between Folds SSE and WLS SSE is 1.009 (Equation 3-9), which shows a difference of less than 1 percent, thus, the model is validated.

$$SSE_{k-fold} = \sum_{i=1}^{k} Test_i(SSE) = 3.2457 \times 10^7 \tag{3-8}$$

$$\frac{SSE_{k-fold}}{SSE_{WLS\ Model}} = \frac{3.2457 \times 10^7}{3.2150 \times 10^7} = 1.009 \tag{3-9}$$

Table 3-7: SSE values for k-fold Cross-validation

Fold test	$SSE_{10-fold}$
Test 1	4.279×10^6
Test 2	1.620×10^6
Test 3	4.687×10^6
Test 4	3.185×10^6
Test 5	1.126×10^6
Test 6	2.029×10^6
Test 7	3.347×10^6
Test 8	9.224×10^6
Test 9	1.252×10^6
Test 10	1.616×10^6

3.5.3 Error Range Estimation

Validated MLR models make reliable point estimates; however, there is no identification of the prediction's error range. Estimates based on MLR models inherit errors from noise in data available or approximations in regression fitting process (Liu 2010). A statistical approach to quantify the prediction's uncertainty is to measure the confidence interval for the MLR point-value estimate (as illustrated in Figure 3-8). The confidence interval gives an expected range of the point-value prediction defined at a certain confidence level. According to Liu (2010), the prediction interval for a point-value estimate is statistically defined by Equation (3-10).

$$\hat{Y}_0 \pm t_{(p,n-k-1)} \times \sigma_p \tag{3-10}$$

Where $\hat{y}_0$ is the predicted point value of the regression model; $t_{(p,n-k-1)}$is the critical value of T-distribution with a probability of p and degrees of freedom of n-k-1, and $\sigma_p$is the standard deviation of the prediction determined by Equation (3-11).

$$\sigma_p = \sqrt{\sigma_{res}^2 + \sigma_f^2} \tag{3-11}$$

Whereσ_{res} is the regression residual standard deviation which can be calculated by Equation (3-12); σ_f is the standard error of regression, which can be measured by Equation (3-13).

$$\sigma_{res}^2 = SSE/(n-k-1) \tag{3-12}$$

Where SSE is the sum of squared errors of regression, and n-k-1 is the regression degrees of freedom.

$$\sigma_f = \sqrt{\sigma_{res}^2 \, X_0 (X^T X)^{-1} {X_0}^T} \tag{3-13}$$

WhereX_0, in form of $[1 \quad x_{01} \quad \cdots \quad x_{0k}]$, is an array of input variables, for which, the confidence interval of the associated regression output needs to be established; X with n rows (n=number of observations) and k+1 columns (k=number of input variables) is the matrix of recorded data presented in the form of Equation (3-14).

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1k} \\ 1 & x_{21} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & \cdots & x_{nk} \end{bmatrix} \quad (3\text{-}14)$$

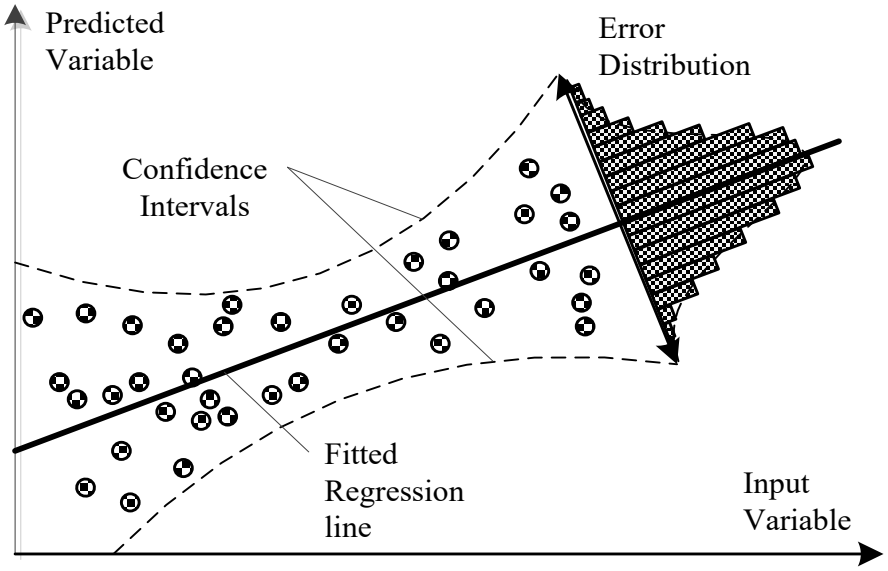

Figure 3-8: Regression confidence interval

3.5.4 Model Validations Based on New Cases

Application of the derived labour cost model (Equation 3-6) is demonstrated in two new cases with detailed inputs given in Table 3-8. Note Cases (1) and (2) were prepared by extracting data from more recently completed projects by the same steel fabricator, which had not been "seen" by the calibrated regression model. The actual labour-hours spent on this project were also available in the labour costing system for validating the predicted point and range estimates. The detailed calculation for creating a range estimate for case 1 is presented by Equation (3-15) to (3-21). The labour-hour point estimate for $X_{case\,1}$, using the developed MLR model (Equation 3-6), is presented by Equation (3-16). The residual standard deviation was calculated as shown in Equation (3-17) for the fitted regression model (Equation 18). For the given point of $X_{case\,1}$, the standard

error of regression was calculated based on Equation (3-17) and the details are shown in Equation (3-18) and (3-19). Using the result of Equation (3-17) and (3-19) the standard deviation of the prediction was determined (Equation 3-20). With 90 percent confidence (i.e. $t_{(0.90,1552)}$) the prediction interval for the point estimate $\hat{Y}_0$ is presented by Equation (3-21). The estimated labour-hours in the form of point and range estimates, for both cases, are given in Table 3-8 and Table 3-9.

Table 3-8: List of Six input variables for model validation

Input variables	Description	Unit	Case 1	Case 2
X1	Division Weight	kg	221703.7	103713
X18	Square hollow steel sections	meter	0	7.12
X14	Plate	square meter	757.16	917.4
X39	Hex Type Bolts	number	9508.1	16
X31	Complete penetration weld	meter	0	554.1
X32	Partial Penetration Weld	meter	4.76	17.9

$$X_1 = [1 \quad 221703.7 \quad 0 \quad 757.16 \quad 9508.1 \quad 0 \quad 4.76] \tag{3-15}$$

$$\hat{Y}_1 = 1025.1 \text{ Labour-hours} \tag{3-16}$$

$$\sigma_{res}^2 = \frac{3.2150\times10^7}{1559-6-1} = 2.0715 \times 10^4 \tag{3-17}$$

$$(X^TX)^{-1} = \begin{bmatrix} 1051714 & -3.59 & -311.54 & -35.26 & -39.57 & 214.61 & -836.32 \\ -3.59 & 0.00 & 0.00 & 0.00 & 0.00 & -0.02 & -0.01 \\ -311.54 & 0.00 & 6.24 & 0.10 & 0.07 & 1.06 & 0.86 \\ -35.26 & 0.00 & 0.10 & 0.38 & 0.00 & -0.15 & 0.01 \\ -39.57 & 0.00 & 0.07 & 0.00 & 0.09 & 6.00 & 0.09 \\ 214.61 & -0.02 & 1.06 & -0.15 & 6.00 & 39.75 & -2.05 \\ -836.32 & -0.01 & 0.86 & 0.01 & 0.09 & -2.05 & 149.70 \end{bmatrix} \times 10^{-9} \tag{3-18}$$

$$\sigma_f^2 = (2.0715 \times 10^4) \times (\ 0.0044) = 91.074 \quad (3\text{-}19)$$

$$\sigma_p = \sqrt{\sigma_{res}^2 + \sigma_f^2} = 144.243 \quad (3\text{-}20)$$

$$\hat{Y}_0 \pm t_{(p,n-k-1)} \times \sigma_p = 1025.1 \pm 1.2821 \times 144.243 = \ (840.2,1210.0) \quad (3\text{-}21)$$

Table 3-9: Results of Labour-hour estimation, range estimation, and actual Labour-hours

Description	Case 1	Case 2
Labour-hours estimated by Equation 6	1025.1	7140.3
Labour-hours estimate range	(840.2 , 1210.0)	(6934.4 , 7346.2)
Actual labour-hours	1103.4	7312.7

3.6 DISCUSSION ON VALIDATION OF MODEL

The developed model for predicting labour-hours required in fabrication of a division with certain design features was verified and validated through checking MLR theoretical assumptions, historical validation, range estimation, and face validation. The proposed framework is relevant to a range of application cases in civil engineering for developing predictive models. However, the model developed using this framework could only be validated in its problem domain and based on the collected data. It would be only applicable to the decision-making situation analogous to the one in the current problem definition. As long as there is no paradigm shift to the practice of structural steel fabrication method leading to major change in labour productivity, the resulting model would remain valid. An example of such a paradigm shift would be elimination of manual operations through the implementation of robotic automation in the steel fabrication industry in the future. In short, to extend MLR implementation to other projects based on an updated dataset (other companies or other industries), this framework will provide valid guidance, starting from problem definition, data collection to model validation. For instance, utilizing the proposed framework, precast concrete panels labour-hour could also be predicted factoring design details and using historical data.

The general problem of associating project design features with required labour-hours will remain relevant and critical in the fields of construction engineering and structural engineering. With recent advances in information technology (such as adoption of BIM, bar-coding or radio frequency identification tags for resource tracking) data availability constraints on solving such problems have been gradually relaxed. On the other hand, MLR lends itself well to construction engineering applications due to simplicity and transparency. Before implementing more

complicated nonlinear methods such as Artificial Neural Networks (ANN), it is advisable to take MLR as the foundation methodology and take full advantage of MLR in gaining insight in the problem definition and available data. Therefore, a practical MLR-based methodology for selecting proper project design features in prediction of required labour-hours is highly desired to improve current industry practice in regard to project cost estimating, planning, and design

The proposed data-driven framework attempts to reap benefits from the current industry's data/information management systems. The scalability of this research is constrained by investments in data management by the construction industry. The value of the framework lies in its potential to extend application to other projects (other companies/other industries) in guiding a repeat implementation step by step starting from problem definition and collecting data, to model validation. It is noteworthy that the derived model itself would lose its value when tackling a new problem or there is a significant change in the current problem domain. The model would not be applicable any more. Nonetheless, the modelling framework is still applicable to reproduce an updated, relevant predictive model. In particular, the step-wise method for MLR input factor selection will remain cost-effective to identify an updated list of parameters that account for the model output of the model in an analytical fashion as opposed to making such decisions by resorting to trial-and-error or "gut feel".

3.7 CONCLUSION

This research has devised an application framework for developing a MLR model in its simplest form and remains valid for labour-hours prediction based on project design features. A case study based on real-world data of structural steel fabrication is presented. In collaboration with a partner company, we consolidated a database holding over one thousand and five hundred historical records (project divisions) accumulated over the span of three years; each record includes forty-two design features for a project division and the associated actual labour-hours spent in fabrication. Out of the forty-two project design features, the six most relevant input features were analytically selected, resulting in a streamlined MLR model. The complex relationships and hidden patterns underlying all the data are represented in a regression equation.

This research has validated the effectiveness of the proposed framework by addressing a real-world problem featuring relatively large datasets in terms of the number of input features defined and the number of records in the dataset. Streamlining the number of input features leads to the generation of a simple model for practical use while entailing less effort in data collection in the future. In short, the proposed framework will potentially assist in developing simple yet sufficient decision-support solutions in the real-world based on fully harnessing available data (such as BIM data and labour cost data) –which is indeed not limited to structural steel fabrication. It is worth mentioning that the proposed framework is selected to provide a transparent model and elucidate the structure of data more so than other methods to the best of the author's knowledge. Therefore, the generated productivity models are not only applicable to productivity prediction, but also to understand the productivity-influencing factors, and possibly productivity improvement.

The variable selection method in connection with the proposed model is instrumental in identifying relevant input factors and generalization of the predictive regression model. However, to tackle noisy, non-homogenous, and highly non-linear data, the proposed model would likely fail due to inherent limitations of MLR. In such cases, the resulting MLR model would end up with poorer performance in point prediction accuracy (i.e. low R-squared value) while producing too wide a range in associated confidence interval estimate. Under such circumstances, it is recommended data needs to be cleansed of noise or pre-processed with clustering techniques to transform a highly non-linear problem into a linear problem ready for applying the proposed MLR methodology.

There is a great opportunity for future work on how to cope with intractable non-linear features inherent in real-world data when tackling complex practical problems. Immediate follow-up research is required so to enhance the proposed framework by adding non-linear classifiers prior to MLR regression in order to represent more complex relationships in the problem; while at the same time, attempting to maintain the simplicity of the resulting model.

CHAPTER 4 : PRODUCTIVITY MODELING OF OFFSITE STEEL FABRICATION

This chapter further expands on MSR application and prediction accuracy by integrating it with Model Trees (MT). The proposed methodology has been validated on a University of California Irvine concrete slump dataset and labour productivity dataset from steel fabrication industry. This Chapter elaborates on developing a framework for generating a streamlined system of MLR equations by coupling the power of MSR and MT, and analyzing the trade-off between fit quality, prediction accuracy, and model complexity to further assist with model selection and validation.

4.1 INTRODUCTION

With increasingly complex design, construction projects require planners to account for design details in predicting labour cost. To expedite project delivery time, mitigate environmental impact on project execution while achieving high-quality standards, construction projects resort to off-site prefabrication and assembly of structural components in weatherproof facilities. Prefabricated steel girders or precast concrete segments for building highway overpasses are a great example of such projects. Off-site fabrication shops provide a safer work environment for labourers conducive to higher productivity, while also removing uncertainties and risks associated with site conditions to some extent. Many research studies have validated the cost, safety, quality and environmental advantages of prefabrication (Jaillon and Poon 2014; Li et al. 2014).

Unlike manufacturing, the off-site fabrication of structural steel still relies heavily on manual labour; products are custom made at fabrication shops based on clients' requirements. As a result, labour cost takes up from twenty to fifty percent of the project budget (Sweis et al. 2009). Therefore, for management and operational purposes, it would be crucial for a fabrication facility to have an accurate prediction of the labour effort required to complete a specific scope of work. Prefabrication creates a unique condition for productivity analysis as the majority of productivity-relevant factors remain constant, and productivity is largely influenced by work content and engineering design.

Due to the labour-intensive nature of construction activities, productivity is commonly referred to as labour productivity, represented as a ratio between the input of labour-hours and the output of installed or fabricated units (Equation 4-1) (Dozzi and AbouRizk 1993). As labour is the most significant resource in construction operations, costs of other resources like tools, equipment and field overheads are generally correlated to labour-hours and factored as an add-on to the labour-hourly rates (Song and AbouRizk 2008).

$$Labour\ productivity = \frac{Labour\text{-}hours}{Fabricated\ unites} \tag{4-1}$$

One of the methods for productivity analysis and forecasting is productivity modelling. Productivity models are effective decision-support tools for planning, estimating, and scheduling. These models are used to quantify the relationship between the productivity rate and relevant influential factors (Said and Prathyaj 2017). Factors influencing labour productivity can be grouped into two major categories: (1) nature of the work to be done or Work Content (WC)

and (2) Environment Conditions (EC) (Sweis et al. 2009). In practice, accurately predicting, measuring and controlling the environmental conditions in field construction is nearly impossible (Sweis et al. 2009). In contrast, prefabrication processes are more driven by work content and less affected by environmental conditions.

Many research studies in the past decade emphasized on the importance of historical data and maintained that the most accurate and reliable estimate can be obtained from past project data in terms of labour costs, progress information, project details, and past performances (Said and Prathyaj 2017). However, data gathered from construction projects are almost certain to contain noise and inconsistency. In addition, the construction-related datasets feature a large number of variables and are often influenced by unpredictable events (Sweis et al. 2009). Prefabrication, on the other hand, has created a unique situation for productivity modelling and analysis, where many environmental factors can be isolated, thus significantly increasing the chances of success in data-driven productivity modelling.

The goal of this research is to correlate engineering design features with fabrication productivity in an off-site facility while taking advantage of existing historical data in the development of productivity models. The novelty lies in its unsupervised approach in classifying projects based on work content and design features. The contributions of this study include: (1) proposing a framework to develop productivity models and identifying effects of work content factors, (2) developing an analytical methodology for generating a system of Multiple Linear Regression equations by coupling the power of Modified Stepwise Regression (MSR) and Model Tree (MT), and (3) analyzing the trade-off between model fit quality, prediction accuracy, and model complexity.

In the following sections, first, relevant literature on productivity modelling and model trees are reviewed. Then, the research methodology and objectives are described. The performance of the proposed methodology is first benchmarked on a concrete slump dataset from the University of California Irvine (UCI) machine learning repository. Thereafter, a practical case of steel fabrication productivity modelling is presented before drawing conclusions.

4.2 MODEL TREES

Researchers have utilized various modelling techniques to study the relationship between influential factors and labour productivity; tools such as regression models, expert systems, and artificial neural networks (ANNs) have been used to develop productivity models (Najafi and Kong 2015, Said and Prathyaj 2017). On one end of this spectrum, we have Multiple Linear Regression (MLR) which lends itself well to generalize a simple representation of the relationship between input factors and the output variable (Najafi and Kong 2015). On the other end, more sophisticated modelling tools such as ANN, instance-based learning, and deep neural nets result in productivity models that function like a "black box" without revealing the implicit relationships between input factors and the reasoning logic applied in deriving the output variable based on input factors (Wang and Witten 1996). Nevertheless, in order to accept a model as decision-support tool, most end users of productivity models in construction would weigh more on understanding factor selection and reasoning logic of the model than achieving marginal gains on the accuracy of the prediction model.

Development of MT started as an extension to classification trees by Morgan and Sonquist (1963) which used the automatic interaction detection (AID) method to generate Regression Trees (RT). RTs create a predictive model represented by a tree structure where the feature domain splits between tree branches; at the end of each branch, tree leaves have constant values. Breiman et al. (1984) improved on AID and developed Classification and Regression Tree (CART) to approximate non-linear functions by discretizing them into piecewise models, suitable for predicting both continuous and categorical variables. Quinlan (1992) extended CART application by replacing constant values on tree leaves with linear functions, resulting in a method called M5;

as a result, enabling CART to model non-linear datasets with piecewise linear functions. More recently, Loh (2002) presented a new algorithm called GUIDE for generalized, unbiased interaction detection and estimation. GUIDE divides the residuals of a linear model into negative and positive signs and uses a Chi-square test instead of a t-test to determine the best split. Alternating Trees (ATREE) is one of the latest methods proposed in MT development that uses an additive forward stage-wise approach to build the trees (Frank et al. 2015).

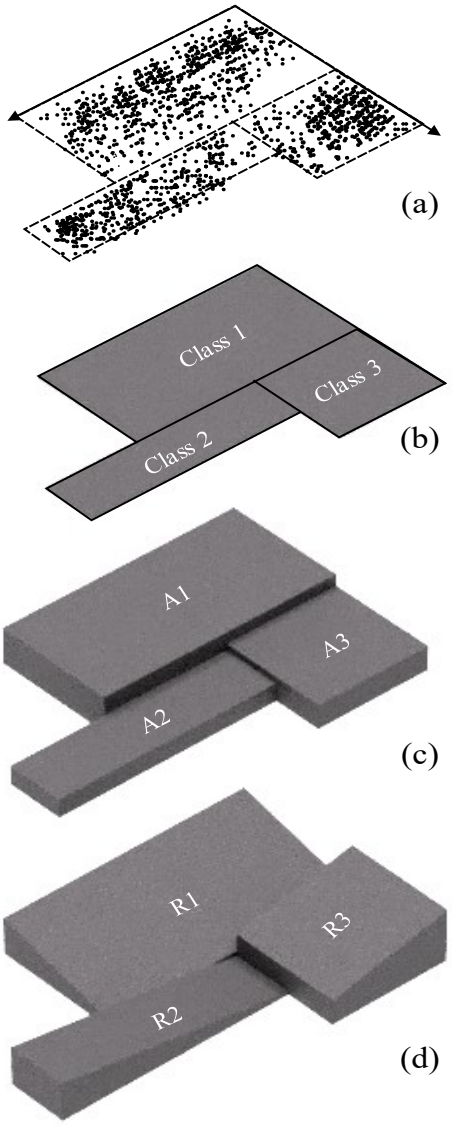

Figure 4-1: (a) Data point representation, (b) Classification, (c) RT, (d) MT

Figure 4-1(a) provides a simplified representation of data points in a given domain. Classification Trees cluster data into discrete classes through binary recursive partitioning over training data and generate a tree-like structure for future prediction (Figure 4-1(b)). RTs are similar to classification trees; however, constant values are assigned –instead of classes– to tree leaves, therefore, providing a constant value for each future prediction that belongs to a specific tree leaf (Figure 4-1(c)). To replace constant values in RT, MTs attach regression models to the end of their branches and utilize input/output correlation to generate future predictions (Figure 4-1(d)). MTs split the data points so the similar samples are clustered for performing MLR at each leaf of the tree (Figure 4-2), resulting in a piecewise regression model, as shown in Figure 4-3 (Quinlan 1992). MTs are more accurate than commonly-applied MLR methods for numeric predictions and can produce more insightful models compared to the opaque structure and implicit formulation of ANN (Frank et al. 1998). In brief, MT is an analytical method for creating predictive models with particular emphasis on the generalization of the problem.

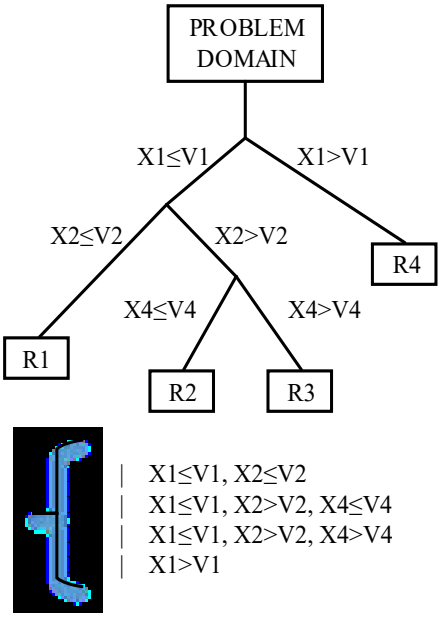

Figure 4-2: MT structure and formulation

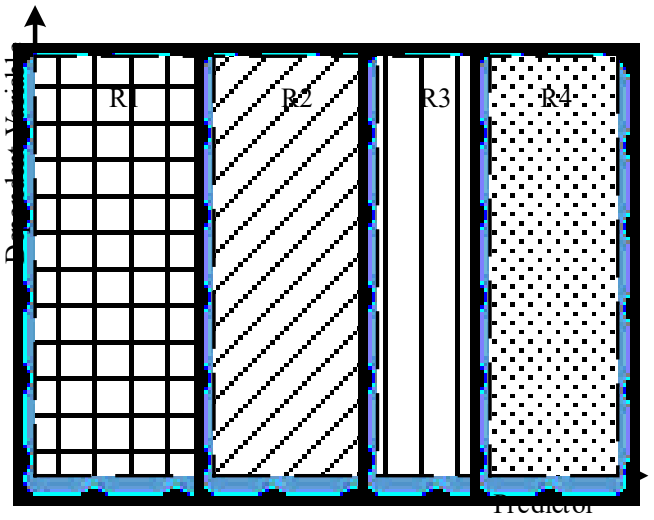

Figure 4-3: MT piecewise representation of non-linear trend in data

4.3 LITERATURE REVIEW

4.3.1 Productivity Modelling

Productivity modelling has been extensively addressed in the construction literature. Randolph et al. (1990) researched different definitions for productivity and proposed two methods to model productivity: a factor-based model and an expectancy model. Rifat and Rowings (1998) developed productivity models considering factors like temperature, quantity and crew size using neural networks and regression analysis. Dawood (1998) proposed Monte Carlo simulation to generate more reliable duration estimates, considering variations in quality of material, weather, and labour productivity. Song and AbouRizk (2008) used ANN and discrete-event simulation to analyze project historical data and develop labour productivity models. Sweis et al. (2009) performed labour productivity modelling using data on fourteen projects that had similar scope, size, specification, and quality requirements and concluded that productivity affecting factors can be categorized into Work Environment (WE) and Work Content (WC). Najafi and Kong (2015) presented a MLR model for predicting the duration of precast concrete installation and noted that MLR could create simple and clear relationships between inputs and outputs. To model the relationship between productivity factors and influencing parameters, Heravi and Eslamdoost (2015) utilized ANN and experienced over-fitting issues that were later resolved by early stopping of the training process and applying Bayesian regularization algorithms. Lee et al. (2017) demonstrated the advantages of Building Information Modeling (BIM) in performing quantity take-offs and productivity analysis through an integrated approach for productivity measurement. El-Gohary et al. (2017) performed a productivity analysis on formwork and reinforced concrete installation for residential and commercial buildings. Said and Prathyaj (2017) investigated

ductwork prefabrication productivity and utilized MLR and ANN to develop predictive models for fabrication labour-hours of different ductwork fittings.

Based on reviewed literature, more in-depth research is desired in understanding the relationship between work content and productivity in off-site prefabrication settings, due to the isolation of most of the environmental factors. There is a need for formalizing an analytical methodology that takes project complexity and design features into consideration in developing a productivity model. The desired productivity model should feature high prediction accuracy while maintaining simplicity and transparency for ease of communication and application. To avoid productivity models from over-fitting, there is also a need for a formalized approach to perform complexity-accuracy trade-off analysis in the model selection.

4.3.2 Model Tree Related Applications in Construction Management

Lee et al. (2004) quantified the productivity loss due to project change orders by applying the GUIDE methodology. They concluded that significant higher accuracy can be obtained by using MTs compared to other methods in construction where data often feature high intercorrelation and contain noise. Desai and Joshi (2010) applied decision trees with constant branch nodes to analyze and predict labour productivity; they implemented a heuristic approach to select influential attributes in building the decision tree. Deshpande et al. (2014) compared the results of non-linear regression, MT and ANN in predicting the compressive strength of recycled aggregate concrete. They indicated that if the input variables are limited, ANN would result in the best single model; however, MTs can produce a set of models with different levels of complexity and accuracy. Omran et al. (2016) compared the effectiveness of applying regression, MTs, and neural networks in predicting concrete compressive strength. They concluded that time consumed in training

advanced models would hinder practicality, which should be considered as a factor in model comparison. Behnood et al. (2017) used MTs –specifically M5– to predict concrete compressive strength. They observed MTs can provide more insight into data and achieve high prediction accuracy while maintaining modelling transparency. Afsarian et al. (2018) implemented M5 trees as a transparent method in predicting building energy consumption.

4.3.3 Variable Selection on MLR

Developing a valid predictive model for labour productivity becomes more challenging with the increase in the number of influencing parameters (Said and Prathyaj 2017). Many researchers have noticed the drawbacks of having too many variables in a predictive model (Gardner et al. 2016, Said and Prathyaj 2017, Mohsenijam et al. 2017). Redundant input parameters in a predictive model would increase the chances of over-fitting while potentially introducing noise into predictive models (Gardner et al. 2016). Interdependencies between variables in a model need to be understood in order to achieve a more transparent model, and redundancies need to be eliminated. Interdependencies within the data refer to cases where a variable could be explained and predicted by other input variables; therefore, there is no need to include it in a predictive model. In the case of the structural steel data collected in this research study, the steel member weight and length are highly correlated, which can be predicted from one another.

Having the optimal number of input variables significantly reduces the likelihood of over-fitting, collinearity, and transferring noise in data to the predictive model (Ivanescu et al. 2016). Mohsenijam et al. (2016) developed a variable selection technique based on forward selection (FS) and backward elimination (BE), called MSR. Forward selection technique starts with a linear regression model that contains no input variables; adds input variables to MLR-based on

correlation and statistical significance; stops when there is no predictive gain in adding more variables (Seber and Lee 2003). Backward elimination begins with an MLR model with all input variables; iteratively removes insignificant variables to reach the final subset; stops when there is a loss in model performance by removing any more variables (Wang and Jain 2003). Both forward selection and backward elimination tend to produce a near-optimum subset of variables and given large datasets, applying the two MLR model streamlining technique separately often results in two different sets of input variables (Mendenhall and Sincich 2015). It is noteworthy that the MSR method combines the advantages of forward selection and backward elimination while overcoming respective limitations of each (Mohsenijam et al. 2016). The MSR starts with an empty set of input variables, similar to forward selection. In an iterative process, variables with the high significance are added to MLR, and in the same iteration, all the selected MLR variables are tested for their statistical significance (Same as the backward elimination method). Unlike forward selection and backward elimination, MSR allows previously eliminated variables to be reinstated in the MLR model, while the already selected variables can be removed in future iterations.

Besides applying M5 methodology to define rules for clustering observations, this research implements MSR at the resulting M5 leaves to eliminate redundant input features. As such, a streamlined model at each leaf can be obtained through purposefully selecting the variables by MSR.

4.4 RESEARCH OBJECTIVE AND METHODOLOGY

The proposed research methodology consists of three main steps, namely: (1) data collection, (2) productivity model generation, and (3) model performance comparison (shown in Figure 4-4). This methodology explores three areas in the MT development process in the application context of construction productivity analysis. They are: (1) grouping recorded data into branches and leaves using MTs, (2) selecting influential attributes using MSR for derived leaves and creating regression models, and (3) performing accuracy-complexity trade-off analysis in predictive modelling.

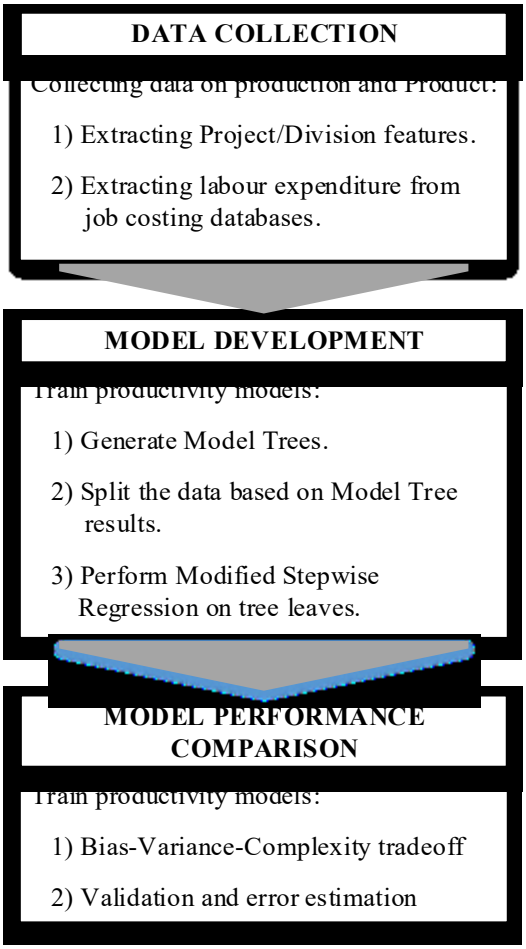

Figure 4-4: Structure of research methodologyM5 Implementation

MTs are generated through (1) tree generation, (2) pruning the tree, and (3) replacing sub-trees with linear regression functions when it is appropriate (Frank et al. 1998). The model structure of MTs consists of root, branch splitting, and leaf nodes. M5 is the selected technique for implementing MT in the present research due to its simplicity and proven performance. Appendix I provides an in-depth justification and comparison on using M5 against other options of MTs.

1. **Tree generation:** M5 develops the tree by recursive partitioning starting from the root node. Branches are created at each splitting node and when there is no predictive gain from partitioning, splitting stops at leaf nodes. The partitioning takes place with the aim to minimize intra-subset variability, measured by Equation (4-2), where SDR is the standard deviation reduction as a result of partitioning, SD is the standard deviation (Equation 4-3), T is the number of observations as a complete set. The resulting partition based on variable X_i with a value of a is two sets of observations: (1) observations where X_i is smaller or equal to a, (2) observations where X_i is bigger than a. T_i is the number of observations for each portioning condition, and $SD(T_i)$ is the standard deviation with that subset; in M5 binary splits, i can take two values. M5 calculates SDR value for input variables and splits dataset based on the specific variable with its particular value that maximizes the expected error reduction. Note that SD is calculated for output values by Equation 4-3, based on the observations subject to each branching condition. Splitting ends if only few instances remain in the branch (i.e. a minimum of thirty observations required for forming a robust linear regression) or expected error reduction is insignificant. The minimum number of observations in each branch is set to be thirty in this study for generating a solid linear regression and being able to perform variable selection.

$$SDR = SD(T) - \sum_i \frac{T_i}{T} \times SD(T_i) \quad (4\text{-}2)$$

$$SD = \sqrt{\frac{\sum_j (y_j - \bar{y})}{n}} \quad (4\text{-}3)$$

2. **Regression fitting**: M5 fits an MLR model for all the splitting nodes using data associated with that node and all the attributes leading to that node.

3. **Pruning the tree**: Pruning refers to the process of removing excessive branches and reducing the tree to a smaller size, which takes into account prediction accuracy and generalization gain in applying MLR at leaves. Specifically, MSR is implemented to analytically derive an MLR model in the leanest form by eliminating less relevant input factors in the dataset being studied. If the expected error of the MLR at a splitting node is lower than the combined results from its branches, then branches are removed (pruning). Note that pruning starts from leaf nodes and removes them in an iterative process.

4.4.1 Bias-Variance-Complexity Trade-off

Numerous statistical and machine learning approaches are available to create models based on given data. However, an important question that needs to be addressed is the trade-off between bias, variance and model complexity. This section elaborates how to assess the combined power of M5 and MSR ("M5+MSR") in regard to model complexity and misclassification error against other methods. In order to assess models developed for complex systems, model selection criteria must be based on the trade-off between bias, variance, and complexity (Yu et al. 2006). Bias is an indicator of the quality of fit, which can be defined as the learning error for the algorithm of choice. A model with a high bias oversimplifies the problem being studied,−analogous to using a linear model to represent a linearly inseparable problem. Variance, as a measure of future prediction

accuracy, denotes the sensitivity of the developed model to the training data provided. In other words, variance represents a model's prediction performance on unseen or new cases, given that a different training data (or a subset of training data) had been used for training. As a result, a predictive model with a high variance does not generalize patterns inherent in the problem from the data, but merely memorize the training data, which would perform poorly on cases it has not seen before.

Lastly, certain noise (i.e. irreducible error) inevitably exists in the data originating from measuring tools, data collection errors, and human-induced errors. The theoretical formulation for the interaction of bias, variance and noise known as bias-variance decomposition of error is given in Equation 4; where, $y = f(x) + \varepsilon$ is the true function to be approximated in predictive modeling, ε is a normally distributed noise, and σ is the standard deviation. Assume that we assess a fitted function of $h(x)$ that has been trained on the training set of data. Given the test set of x' and observed values of y', the $h(x)$ prediction expected error is calculated as Equation 4-4.

$$
\begin{aligned}
E[(h(x') - y')^2] &= E[h(x')^2 - 2h(x')y' + y'^2] = E[h(x')^2] - 2E[h(x')]E(y') + E[y'^2] \\
&= E\left[(h(x')^2 - \bar{h}(x'))^2\right] + (\bar{h}(x'))^2 - 2\bar{h}(x')f(x') + E[(y' - f(x'))^2] \\
&+ f(x')^2 = E\left[(h(x')^2 - \bar{h}(x'))^2\right] + E[(y' - f(x'))^2] + [(\bar{h}(x') - f(x'))^2] \\
&= Var(h(x')) + \sigma^2 + Bias^2(h(x'))
\end{aligned}
\tag{4-4}
$$

The expected error in prediction between a true function and a fitted model can be represented by the summation of variance, bias squared and noise. Therefore, to reduce prediction error, both variance and bias need to be minimized simultaneously. Bias (i.e. error of predictive algorithm) decreases as the model complexity increases; conversely, it would cause variance to escalate

(Figure 4-5) (Geman et al. 1992). Nevertheless, oversimplified models can be associated with higher bias and lower variance in terms of the prediction error. The best predictive model would achieve the best trade-off between bias and variance, thus minimizing the total error in prediction.

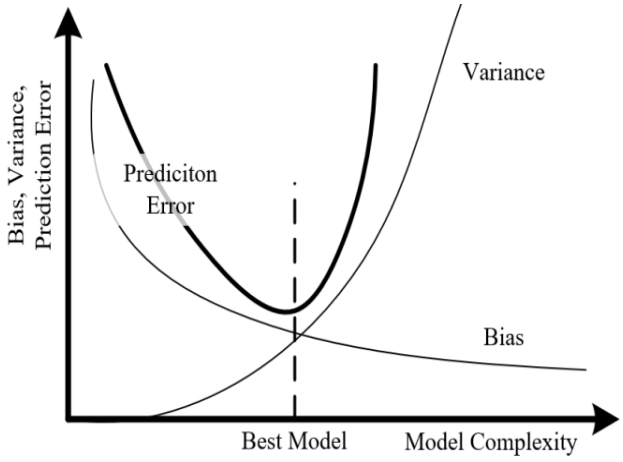

Figure 4-5: Bias, variance and complexity trade-off

The "best fit" or the "best model" can be assessed as the model that has the best generalization ability by demonstrating the best performance in predicting new observations (Moody 1994). Figure 4-6 shows an example of a simple linear model that can generalize better than a non-linear function given the same observation data. The linear model in this example has a higher bias compared to the non-linear model; however, the linear model has significantly less variance, thus is deemed a better fit by giving more accurate future predictions.

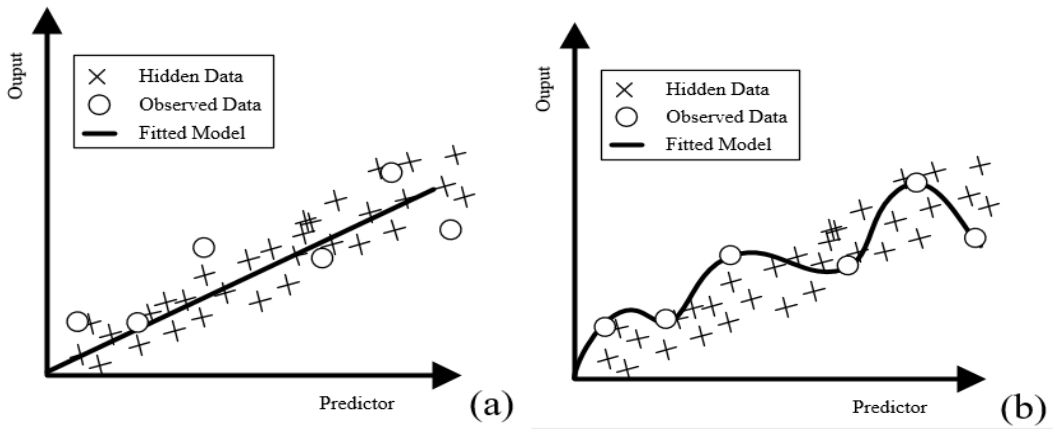

Figure 4-6: Model generalization: (a) linear model (b) non-linear model

There are several model selection methods that take bias, variance and complexity into account; three of the main categories of model selection methods are as follows:

(1) Root-Mean Square Error ($RMSE$), and coefficient of determination (R^2) (Lowe et al.2006).

(2) Akaike Information Criteria (AIC) (Akaike 1977) and Bayesian Information Criteria (BIC) (Schwartz 1978).

(3) Final Prediction Error (FPE) (Yu et al. 2006).

The first group ($RSME$ and R^2) is the most widely used predictive model performance measurements for indicating the goodness of fit and fit quality (Bias) in model selection (Lowe et al. 2006). RSME and R^2 can be measured by Equation (4-5) and (4-6) respectively, where y′ is the observation value, $\bar{y}$ is average of observations, h is the fitted function, n is the sample size, and D is the model's degrees of freedom. Degrees of freedom in predictive models can be defined as the number of independent variables and their coefficients affecting the dependent variable. A larger D indicates the higher complexity of the model structure.

AIC value gives an estimate of the model performance in terms of future predictions, which can also be used as a model selection tool. A model with the smaller value of AIC has better chances of mitigating model over-fitting by penalizing the high number of degrees of freedom in the model. In contrast to AIC, BIC penalizes more on the degrees of freedom in smaller sample sizes. Therefore, AIC is a better indicator applicable to problems with large sample sizes and BIC is more meaningful in dealing with smaller sample sizes. AIC and BIC can be calculated by Equation 4-7 and 4-8, Where n is the number of data points, RSS is the residual sum of squares, and D is the model degrees of freedom. When two models perform comparably well with respect to AIC and BIC (i.e. producing similar RSS value on identical sample sizes), the one with smaller degrees of freedom (i.e. simpler model) is more favourable.

FPE is a model fitness measure based on the trade-off between variance, bias, and complexity, where lower values of FPE is more desirable; the FPE formulation is presented in Equation 9.

$$RMSE = \sqrt{\frac{\sum\left(y' - h(x')\right)^2}{(n - D)}} \tag{4-5}$$

$$R^2 = \frac{\sum(h(x') - \bar{y})^2}{\sum(y' - \bar{y})^2} \tag{4-6}$$

$$AIC = n + nlog(2\pi) + nlog(RSS/n) + 2D \tag{4-7}$$

$$BIC = n + nlog(2\pi) + nlog(RSS/n) + \log(n)\,(D) \tag{4-8}$$

$$FPE = [(n + D)/(n - D)].[Bias^2 + Var] \tag{4-9}$$

4.5 METHOD PERFORMANCE BENCHMARKING

The proposed methodology is first demonstrated on a benchmarking dataset from UCI machine learning repository (https://archive.ics.uci.edu/ml/index.html). Yeh (2006) collected one hundred and three (103) slump results of different concrete mix results, with different values for cement (X1), blast furnace slag (X2), fly ash (X3), water (X4), super-plasticizer (X5), coarse aggregate (X6), and fine aggregate (X7), all in kg/m3. Detailed results for MLR and MSR implementation and comparison on the same dataset can be found in Mohsenijam et.al (2017); herein, the developed predictive models and their evaluation metrics are presented in Table 4-1 and Table 4-2, respectively. To shed light on the "M5+MSR" process in accordance with the proposed methodology, the following steps need to be followed:

***Step (1) finding the 1st splitting variable and setting its value*:** in this step, the dataset is sorted based on each variable and standard deviation reduction (i.e. SDR as in Equation 4-2) is calculated for each of the seven variables with their one hundred and three (103) values. A variable and its value with the highest SDR is chosen for the first split; data set for future splitting is divided based on values of splitting variable. The results from the first step on the slump data set indicate that X_4 with SDR value of 1.495 has the highest SDR value among all the variables; sample calculation are demonstrated in Equations 4-10, 4-11, 4-12, and 4-13. Result of this split leads to a tree with two branches, one with thirty-four samples, and the other branch with sixty-nine samples. With the purpose of MLR in mind, enough samples are required for a robust model; therefore, the branch with thirty-four samples will not be further split.

$$SD(T) = \sqrt{\frac{\sum_j (y_j - \bar{y})}{n}} = \sqrt{7887.46/103} = 8.750 \quad (4\text{-}10)$$

$$SD(T_1) = \sqrt{3459.23/34} = 10.086 \quad (4\text{-}11)$$

$$SD(T_2) = \sqrt{2369.50/69} = 5.860 \quad (4\text{-}12)$$

$$SDR = SD(T) - \sum_i \frac{T_i}{T} \times SD(T_i) = 8.75 - [(34/103) \times 10.086 + (69/103) \times 5.86] = 1.495 \quad (4\text{-}13)$$

Step (2) finding the second splitting variable and value:

Performing the same method on the branch with 69 samples results in selection of X_7 with the value of 737 to be the splitting criteria. As a result, the developed tree would have 32 and 37 samples for MLR. At the end of this stage, the tree is generated and needed to be pruned by removing insignificant branches.

***Step (3) Regression fitting*:**

In this step, a linear regression model for all the nodes with samples reaching that node needs to be fitted. Generalization performances of the regression models are compared in Table 4-2.

***Step (4) Pruning*:**

as the regression models represented by the full tree (i.e. three branches and three leaves) have a significant standard deviation reduction, there is no need for pruning in this case.

***Step (5) Variable selection*:**

the tree structure and the samples that reach the leaves resulting from the previous steps need to undergo MSR variable selection process [detailed steps can be found in Mohsenijam et al. (2017)]. The resulting M5+MSR model maintains the tree structure with sufficient variables in regression models at the tree leaves (Table 4-1).

Table 4-1: Alternative models generated for slump dataset

Modelling Method	Generated Model	Splitting Condition
MLR	R= -88.525+0.010X1-0.013X2+0.006X3+0.259X4-0.184X5+0.030X6+0.039X7	No condition
MSR	R=-18.099-0.040X2+0.199X4	No condition
M5+MSR	R1 = -53.109-0.043X1-0.113X2+0.468X4 R2 = 101.57-0.0395X2-0.114X7 R3 = 22.681	X4≤184 X4>184 X4>184; X7>741

Based on the collected data, Yeh (2007) produced two predictive models for concrete slump: (1) Non-Linear MLR (NLMLR) model and (2) ANN model. The NLMLR model has fifty-six (56) regression coefficients, shown in Equation 4-14, where x_i is the ith independent variable, q is the total number of independent variables; β_i, β_{ij} are regression coefficients. The ANN model has seven input variables, one hidden layer with seven hidden nodes and fifty-six (56) ANN transformation weights. Yeh (2006) reported the RMSE values for the NLMLR and ANN models being 5.95 and 5.53, respectively. For further testing and analysis in this research, the NLMLR and ANN were reproduced in the current research.

$$y = \sum_{i=1}^{q} \beta_i x_i + \sum_{i<j}^{q} \sum \beta_{ij} x_i x_j \quad (4\text{-}14)$$

In the Bias-Variance-Complexity analysis, performance indicators mentioned in the previous section are calculated for the slump dataset, and results are presented in Table 4-2. Although ANN has higher model fit quality (i.e. lower RMSE and higher $\boldsymbol{R^2}$), when compared with "M5+MSR", it has lower values of AIC, BIC, and FPE, implying over-learning from the training dataset. "M5+MSR" features the second least number of coefficients to be estimated next to MSR (i.e. 4) while the second lowest RMSE or R^2 next to ANN. As a result, "M5+MSR" is declared the best trade-off between fit quality, prediction accuracy, and complexity in this case.

Table 4-2: Model comparison on slump dataset

Modelling method	Coefficients to be estimated	RMSE	$\boldsymbol{R^2}$	AIC	BIC	FPE
MLR	8	7.42	0.32	375.35	375.44	83.30
MSR	2	7.44	0.29	369.40	369.44	78.86
M5+MSR	4	5.70	0.59	347.11	347.16	61.28
NLMLR	29	5.95	0.52	381.47	381.85	84.93
ANN	56	5.53	0.71	389.34	389.72	92.62

4.6 PRODUCTIVITY MODELLING APPLICATION

In this section, a predictive model for the steel fabrication labour productivity is developed utilizing the proposed methodology. The objective is to develop a productivity model by correlating work content and labour-hour expenditure and analyze model fit quality, prediction accuracy, and complexity.

The steel industry is considered as one of the main drivers of prefabrication movement; steel would retain its fabrication properties and have lighter structural pieces for more efficient transportation (Warrian 2010). Examples of off-site steel fabrication projects include pipe and pipe rack fabrication, hollow core floor systems, modular house construction, and steel bridge girders. Fabrication of structural steel in the controlled environment of fabrication shops significantly improves quality and productivity (Liddy and Cross 2002). Prefabrication would create a uniform environment with proper supervision, similar tools and equipment, adequate inspection, and much more control over processes and activities. To a large degree, off-site construction would limit the number of influencing factors (e.g. weather and site condition) or normalize effects of the remaining factors (e.g. crew skill level).

In steel fabrication facilities, similar to most of the prefabrication projects, construction sequence defines the work breakdown structure in work planning. The same work breakdown structure defines the cost accounts of the job costing and tracking systems in which labour-hours actually incurred on the shop floor are recorded. The component of this work breakdown structure in steel fabrication industry is generally referred to as “Project Division”. Labour-hour expended on

fabrication processes like handling, cutting, fitting, welding and surface preparation is measured against project divisions (Hu et al. 2014).

Data gathered for this study includes 1558 records for structural steel project, each record representing labour-hour spent on each steel fabrication division. These labour-hour records are associated with forty-two project-related design features collected from BIM databases (details of design feature are given in Table 3-2). For each project division, collected labour-hour is the summation of hours spent on activities such as handling, cutting, fitting, welding, and surface processing in the fabrication shop. Collected data for this research is sourced from a steel fabricator in Alberta, Canada, with mainly industrial steel fabrication projects. Labour-hours in each division can vary from 100 to 7000 labour-hours. In order to conceal sensitive company productivity information while maintaining the patterns in the data, all the captured data has been linearly scaled. For more information and to gain access to the database, refer to the Data Availability Statement.

The "M5+MSR" method is applied to model labour productivity following the steps demonstrated in the previous section (Figure 2-7). Note MSR method is applied to the tree leaves, creating streamlined MLR regression models. The combined results of branching and MSR are given in Table 4-3; X1 is the division weight (kg), X2 is angle length (m), X3 is channel length, X6 is miscellaneous beams (m), X9 is wide flange beam length (m), X12 is checker plate area (m^2), X14 is the plate area (m^2), X18 is hollow steel section length (m), X19 is round hollow steel sections length (m), and X32 is partial penetration weld length (m).

The selection of plate area (X14) by applying M5 method as the branching variable is well aligned with industry know-how. Among practitioners, the amount of plate required in a steel project is an indicator of the complexity of a project. High plate content in a steel fabrication project generally means more complex connections and higher welding hours. Projects with less plate are branched into R1 (Figure 4-7). In R1, MSR method application indicates that productivity can be predicted by those variables related to structural section lengths (i.e. wide flange (X9), hollow sections (X18), angels (X2) and channels (X3)). This is consistent with current practice as steel industry would refer to these types of projects as Stick-built. Projects with higher plate requirement are branched into R2 and R3 (Figure 4-7). MSR application identifies weight (X1), weld length (X32), and round hollow steel section length (X19) being the main predictors of productivity in high plate content projects. In practice, projects with high plate requirement are generally referred to as Platework. These projects require a significant amount of fitting and welding in fabrication and assembly.

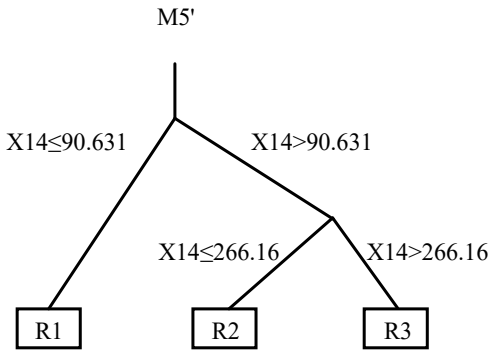

Figure 4-7: M5 tree branching structure

Table 4-3: Result of different branching methods and MSR

Branching Method	MSR results for Tree leaves	RMSE	R^2
M5	R1 = 14.84+0.004X1+0.69X9+0.29X3+8.32X6+ 0.254X2+0.566X18+1.854X14+1.930X12+11.29X32 R2 = 290.43+0.014X1+1826.1X19+9.871X32 R3 = 371.5+0.016X1+9.752X18+12.039X32	369.86	0.763

The results of the proposed methodology are later compared with MLR, MSR, and ANN. The metrics defined in pervious sections are utilized to assist in finding the balance in Bias-Variance-Complexity; results are shown in Table 4-4. The results indicate ANN has higher fit quality by having higher R^2; however, similar to the findings in slump dataset, "M5+MSR" has lower values of AIC, BIC, FPE. As a result, "M5+MSR" achieves higher fit quality and pattern generalization performance compared to MSR or MLR; when exposed to new data points (unseen cases), the model resulting from "M5+MSR" would have higher prediction accuracy while maintaining transparency.

Table 4-4: Results of model selection criteria for different modeling methods

Modelling method	**Degrees of freedom**	**RMSE**	R^2	**AIC**	**BIC**	**FPE**
MLR	25	426.94	0.687	11036.1	11065.9	17256.0
MSR	6	426.64	0.682	11043.4	11012.6	16932.0
M5+MSR	16	369.86	0.763	10849.3	10914.3	15245.5
ANN	130	523.93	0.876	11313.1	12141.0	29297.8

4.6.1 Model Validation

In the previous section, it has been pointed out that the "M5+MSR" has the highest prediction accuracy among methods tested. To further validate the model, the percentage split method is used for model validation; ten percent of data (i.e. 150 unseen instances) were randomly selected and reserved to test the resulting model and calculate the prediction error. If the training and testing results are close to each other (within 10%), the model prediction accuracy is validated (Sargent 2013). The correlation coefficient and root mean square for the training and test data are given in Table 4-5. The RMSE from training and test sets is shown to be close to one another, Figure 4-8 (within the 10 percent range).

Table 4-5: Model validation results

Indicator	Training set	Test set
RMSE	369.86	399.96
$\boldsymbol{R^2}$	0.763	0.689

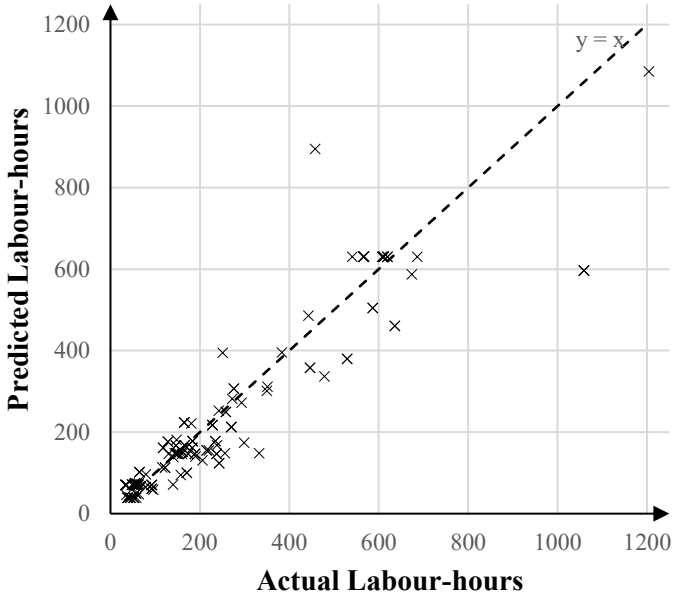

Figure 4-8: Predicted vs. Actual labour-hours validation results

4.6.2 Prediction Demonstration

Application of the derived labour productivity model is presented in this section with two new cases (detail of inputs given in Table 4-6). This data was taken from more recent projects and had not been part of the model training data. Note zero-value variables are not shown for ease of demonstration. Labour-hour required for case one with the plate area of 31.98 needs to be predicted by R1; labour-hour prediction for case two is predicted by R2, as X14 has a value of 99.52. The results for the two cases are 1713.38 and 1052.82 labour-hours, respectively. Against the actual labour-hours, the differences are 123.38 labour-hours (i.e. 1713.38 – 1590) for Case 1 and 140.82 labour-hours (i.e. 1052.82 – 912) for Case 2, which represent 7.76% and 15.4% difference relative to actual labour-hours, respectively.

Table 4-6: Demo case input variables

Input variables	Description	Unit	Case 1	Case 2
X1	Division Weight	Kg	56746	31660
X2	Angle	Meter	88.271	146.34
X9	Wide flange	Meter	246.155	438.04
X14	Plate	Meter-squared	31.981	99.52
X18	Hollow steel sections	Meter	194.325	0
X32	Partial penetration weld	Meter	98.318	32.332
Y	Actual Labour-hours	Hours	1590	912

4.7 BENCHMARKING M5+MSR AGAINST OTHER MODEL TREES METHODS

In recent decade, researchers have proposed many methods for Model Tree development. This section explores results of coupling MSR with other MT and RT methods for steel fabrication productivity modelling. The compared methods are CART, M5, GUIDE, and ATREE. CART regression tree is generated in Salford Predictive Modeler® (2018); M5 Trees are developed using M5PrimeLab toolbox (Jakabsons 2016); GUIDE tree is created by GUIDE software from Loh (2002), and ATREE is performed by using AlternatingModelTrees (Frank et al. 2015) package for WEKA (Hall et.al 2009). CART, M5, GUIDE and ATREE are distinguished in their approach in building trees; the results of each branching methodology is given in Figure 4-9, 4-10, and 4-11. The results of applying MSR to tree leaves after each branching method are presented in Table 4-7.

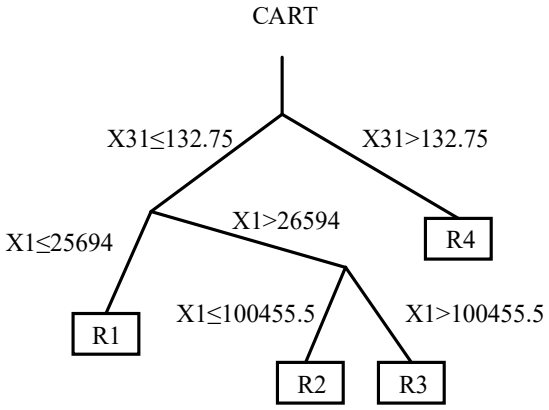

Figure 4-9: CART tree branching structure

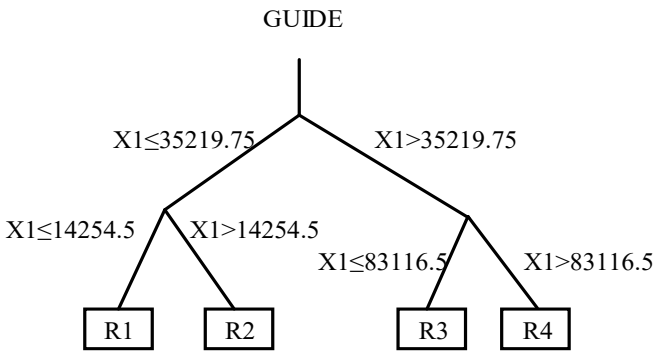

Figure 4-10: GUIDE tree branching structure

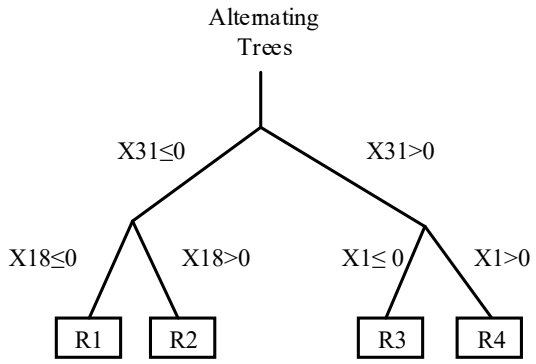

Figure 4-11: ATREE tree branching structure

Table 4-7: Result of different branching methods and MSR

Branching method	MSR results for Tree leaves
CART	***R1*** *= 18.04+0.019X1+1.224X18+0.45X14+9.81X32* ***R2*** *= 381.44+0.012X1+10.064X32* ***R3*** *= 2922.4* ***R4*** *= 7109.4-76.789X3*
M5	***R1*** *= 14.84+0.004X1+0.69X9+0.29X3+8.32X6* *0.254X2+0.566X18+1.854X11+1.930X13+11.29X32* ***R2*** *= 290.43+0.014X1+1826.1X19+9.871X32* ***R3*** *= 371.5+0.016X1+9.752X18+12.039X32*
GUIDE	***R1*** *= 25.14+0.022X1+0.71X18* ***R2*** *= 555.13+14.184X32* ***R3*** *= 1074.3+13.715X32* ***R4*** *= 2401.1+10.77X32*
ATREES	***R1*** *= 29.07+0.016X1+0.383X9* ***R2*** *= 78.56+0.013X1-18.568X4+71.415X19 +3.414X14* ***R3*** *= -27.554+0.025X1+13.881X32* ***R4*** *= 4235.7*

Model selection metrics, presented in Bias-Variance-Complexity section, for different branching methods in combination with MSR are given in Table 4-8. The results of all metrics unanimously indicate that combination of M5 and MSR has superior fit quality while having higher prediction accuracy.

Table 4-8: Results of model selection criteria for different branching methods

Modelling method	**RMSE**	$\boldsymbol{R^2}$	**AIC**	**BIC**	**FPE**
CART+MSR	410.77	0.7075	10991.3	11021.1	16694.3
M5+MSR	369.86	0.7628	10849.3	10914.3	15245.5
GUIDE+MSR	440.23	0.6636	11085.9	11163.6	18251.3
AT+MSR	431.98	0.6761	11060.3	11106.1	17680.8

4.8 CONCLUSION

MLR models are used for many decades in engineering and construction applications for transparency, simplicity, and ease of use. In coping with complicated problems and large datasets in the real-world, MLR has shown limitations in prediction accuracy. As a result, many researchers have resort to more complex methods such as nonlinear regressions and ANN models in order to achieve higher accuracies. Unfortunately, such analytically complex methods fall short of explaining how the model reasons. In some cases, to achieve higher accuracy, the model tends to overfit itself onto the training data by memorizing noises instead of generalizing patterns. The proposed new method combines the advantages of Model Trees and MLR (M5+MSR), which can achieve both transparency and accuracy by generating predictive models using piecewise approximation. M5 coupled with MSR variable selection for predictive model generation have shown promising results on two application cases presented in this study. In modeling Labour productivity and concrete slump prediction, the proposed new method achieves higher fit quality and pattern generalization performance compared to commonly-applied linear or non-linear regression techniques; when exposed to new data points (unseen cases), the resulting model would have higher prediction accuracy while maintaining simplicity and transparency. To avoid models from over-fitting, this research has also formalized an approach to perform complexity-accuracy trade-off analysis in model selection.

With great advancements in data management acquisition technologies and the push for a more modularized and off-site construction, there is a pressing need to develop a different approach to perform productivity analysis of for construction prefabrication facilities in construction. Prefabrication has created a unique situation for productivity modelling and analysis, where many

environmental factors can be isolated, thus significantly increasing the chances of success in data-driven productivity modelling. In the steel fabrication productivity modelling application, variables selected for splitting in M5, and variables selected for regression modelling are well aligned with industry practitioner's know-how. This degree of transparency in reasoning logic is generally impossible for highly non-linear regression models such ANN to attain.

There are application problems where the relationships are extremely complex and there are no meaningful relationships between variables (e.g. image recognition, computer vision). Therefore, the method for predictive modelling would be chosen only based on highest prediction accuracy achievable. When a predictive model is desired to (1) be transparent in factor selection and reasoning logic, (2) be straightforward to be implemented in practice, (2) be able to generalize patterns in training data for reliably predicting unseen cases, the proposed "M5+MSR" approach holds high potential to provide the analytical solution to develop data-driven predictive models.

CHAPTER 5 : CONCLUSION

This Chapter draws the research conclusions, restate the academic and practical contributions, and finally state the limitations of this research for further research.

5.1 RESEARCH CONCLUSION

This research advances the existing knowledge of predictive analytics for construction productivity modelling. Within this research, quantitative methods have been proposed, developed, and validated that would benefit researchers and practitioners by tapping into knowledge that is captured through data collection and streamlining their predictive models, while simultaneously maintaining transparency and accuracy. From a practical point of view, the proposed methodology is not deemed “black box” and is built of analytical foundations instead of heurisitics and simulations. Thus, the derived results do not require the “trial and error” process as applied in neural network training while interpretation of the results is straightford in terms of what role each input parameter plays in deriving the predicted output and how the output is related to the inpout factors in addition to making a point-value prediction. This is critical to make an AI model acceptable and effecitive in rendering decision support in the intended application setting in the real world.

This research focuses on productivity modelling in construction prefabrication facilities due to their ability to limit the influence of environmental factors on productivity. The controlled environment of prefabrication facilities allows for higher quality data collection, and their consistent crew size and standardized processes cause productivity to be a function of project

content and details. Overall, the industry push for prefabrication in recent decades has created a unique situation for productivity modelling and analysis, where many environmental factors can be isolated, thus significantly increasing the chances of success in data-driven productivity modelling. Therefore, the primary constraint of external environmental factors identified in productivity data collection and analysis does not exist in this setting. Overcoming the challenge of developing a transparent, accurate, and simple method for productivity modelling using readily available data is the outcome of this research. The data collected for this research has considerably reduced the influence of other productivity impact factors, since the data has been captured under similar work conditions, following similar production processes, and executed labour crew with equivalent skills and qualifications. It is worth mentioning that the proposed framework is selected to provide a transparent model and elucidate the structure of data more so than other methods to the best of the author's knowledge. Therefore, the generated productivity models are not only applicable to productivity prediction, but also to understand the productivity-influencing factors, and possibly productivity improvement.

As discussed in Chapter 2, regression has been frequently applied to input/output prediction problems; however, given fast-growing data collection technologies, there is a need for variable selection, validation, and verification for practical application needs. There is a high demand for methods that are computationally simple, fast to calibrate, straight-forward to explain, and easy to update as new data become available. Modified Stepwise Regression (MSR) is a proposed solution to solve such problems; MSR analytically selects a proper set of variables by testing their significance in predicting an output, resulting in a reduced number of input variables, and hence, less data collection efforts and less introduced noise. MSR takes Multiple Linear Regression (MLR) assumptions into consideration, and if assumptions are violated, uses Weighted Least

Square (WLS) for generating the MLR coefficients. The proposed MSR approach is illustrated and benchmarked on two case studies of segmental bridge installation and concrete slump testing. Through a practical case study, MSR demonstrated significant advantages over the trial-and-error approach in variable selection. It is worth mentioning that both case studies used in this research feature a relatively small number of input variables (the concrete slump dataset has seven inputs, while the precast bridge dataset has four).

Chapter 3 further elaborates on associating project design features with project cost, leveraging recent advancements in information technology such as BIM, bar-coding or radio frequency identification tags for resource tracking and data collection. MSR methodology is used to select relevant project design features in order to predict required labour-hours. The size of the dataset used in the chapter validated the applicability of the MSR approach in practical settings, when there are numerous variables and a noisy dataset. Out of 42 project design features, the six most relevant input features were analytically selected, resulting in a streamlined MLR model. The complex relationships and hidden patterns underlying all data are represented in a regression equation in its simplest form. It is worth mentioning that this research has validated the effectiveness of the proposed framework by addressing a real-world problem featuring relatively large datasets (in terms of number of input features defined and the number of records in the dataset). Streamlining the number of input features simplifies the model for practical use which minimizes future data collection efforts. In short, the proposed framework will potentially assist in developing simple, yet sufficient, decision-support solutions in the real-world, by fully harnessing available BIM data and labour cost data. And indeed, the applications of these solutions are not limited to the structural steel fabrication domain.

Chapter 4 describes a novel methodology that combines the advantages of Model Trees and MSR to achieve both transparency and accuracy by generating predictive models using piecewise approximation. M5 coupled with MSR variable selection for a predictive model generation have shown promising results on two application cases, labour productivity modelling and concrete slump, presented in this chapter. The proposed new method achieves higher fit quality and pattern generalization performance compared to commonly-applied linear and non-linear regression techniques. When exposed to new data points (unseen cases), the resulting model would have higher prediction accuracy while maintaining simplicity and transparency. To avoid models from over-fitting, this research has also formalized an approach to perform complexity-accuracy trade-off analysis in model selection. When a predictive model is desired to be (1) transparent in factor selection and reasoning logic, (2) straightforward to implement in practice, (3) able to generalize patterns in training data in order to reliably predict unseen cases, the proposed "M5+MSR" approach holds high potential to provide the analytical solution to develop data-driven predictive models. Productivity models generated using the proposed M5+MSR method

5.2 ACADEMIC CONTRIBUTION

The academic contributions of this research study to existing knowledge include the following:

- A reliable MLR model requires an appropriate set of input variables that can satisfy the underlying assumptions of Best Linear Unbiased Estimators (BLUE). In this research study, an analytical framework is proposed for developing MLR-based predictive models by (1) selecting input variables by a modified stepwise approach, (2) verifying the BLUE assumptions, and (3) validating the prediction performance of the regression model.

- There is no formalized method for how to perform stepwise regression in the current literature. To the best of our knowledge, the applications of stepwise regression are limited to the use of statistical software, and there is no insight into the method itself. Additionally, there is no single reference which explains stepwise regression in a straightforward fashion. This research clarifies the stepwise regression procedure and modifies it to incorporate the checking of BLUE assumptions and performing error analysis.

- This study formalizes a generic framework for generating MLR models consisting of variable selection, model verification, model validation, and prediction error estimation. A refined version of stepwise regression is implemented for variable selection; if any of the OLS (Ordinary Least Square) assumptions are violated, the WLS is used for estimating the MLR coefficients. The proposed framework is illustrated and tested in two case studies.

- A practical and widely applicable framework is proposed that uses MLR to associate project costs and project-specific design features.

- The modified stepwise regression identifies a minimal set of design features as input variables to account for project labour cost in fabrication or construction based on a dataset that is of practical size and contains noise. By fully harnessing available BIM and labour cost data in a real-world application setting, the proposed framework assists with developing sufficient, yet straight-forward, decision-support solutions for a variety of construction applications.

- This research study enhances the predictive modelling capabilities of Multiple Linear Regression (MLR) by integrating Model Trees (MT) and Modified Stepwise Regression (MSR).

- This research study proposes an analytical application framework for M5+MSR application in engineering, resulting in a system of MLR equations, each having the least amount of relevant input factors.

- The proposed methodology enables MLR to mimic non-linear regression or ANN while maintaining modelling simplicity and transparency, which is crucial to applications in civil engineering. A detailed study on bias-variance-complexity analysis was also performed to compare the proposed new method against ANN.

- The research enables the advancement of productivity modelling in construction by correlating engineering design features with fabrication productivity in an off-site facility.

5.3 INDUSTRIAL CONTRIBUTION

- The industrial contributions of this research are summarized based on collaborative research with the partner company and a real-world project case study, as below:
- The proposed methodology efficiently uses available data and guides the development of sufficient and reliable MLR models based on data gathered in practical settings. This research proposes a new framework to determine the achievable prediction accuracy when applying MLR to a practical problem using real-world data. This framework is designed to identify the minimal set of inputs required for MLR modelling (i.e., the most relevant input factors, based on the available dataset and given a particular problem definition) without compromising the achievable maximal prediction accuracy. The framework encompasses analytical methods for verifying the MLR application, validating the resulting model, and setting confidence intervals on point-value predictions.
- With great advancements in data management acquisition technologies and a practical demand for modularized and off-site construction, there is a pressing need to develop a different approach to performing productivity analysis for construction prefabrication facilities. Prefabrication has created a unique situation for productivity modelling and analysis, where many environmental factors can be isolated, thus significantly increasing the chances of success in data-driven productivity modelling. In the steel fabrication productivity modelling application, variables selected for splitting in M5 and variables selected for regression modelling, are well aligned with industry practitioners' know-how. This degree of transparency in reasoning logic is generally impossible to attain for highly non-linear regression models such as ANN.

- The proposed approach is capable of reducing data collection efforts for MLR modelling in application fields, ensuring the MLR models' validity, providing point estimates based on a streamlined linear regression equation, and quantifying the error of point estimates according to the desired confidence level.
- The proposed approach guides the construction industry in best utilizing the data available within each company, similar to the dataset of Building Information Model (BIM), and predicting labour productivity by analyzing project work content relevant to key design features of projects.

5.4 LIMITATIONS AND RECOMMENDATION FOR FUTURE RESEARCH

This research, in short, proposes a more transparent, simple and practical application framework for developing predictive models in construction engineering and management. The proposed method of MSR streamlines the variables used in a predictive model by properly selecting input parameters. However, to tackle noisy, non-homogenous, and highly non-linear data, the proposed model would likely fail due to the inherent limitations of MLR. In such cases, the resulting MLR model would have poorer performance in point prediction accuracy (i.e. low R-squared) while producing wider than desired range estimates. In consequence, either the data needs to be cleansed of noise or preprocessed with clustering techniques that transform a highly non-linear problem into a linear problem, prior to applying MLR. To address nonlinearity limitations of MLR, MSR was later coupled with Model Trees (MT) to split the feature domain and to assign a predictive model to each tree leaf. This approach created a piecewise linear model to adjust to the nonlinearities inherent in the data structure.

The scalability of this research is limited by data management in the construction industry. The status of data management in the construction industry has led to incomplete, incompatible, and noisy data being collected. The proposed data-driven framework attempts to reap some benefits from the industry's investment in data/information management systems. The value of the framework lies in the potential of extending its application to other projects (other companies/other industries) in guiding a repeat implementation step by step starting from problem definition and collecting data, to model validation.

It is noteworthy that the derived models in this study would lose their value when tackling a new problem or there is a significant change in the current problem domain. The model would not be applicable anymore. Nonetheless, the modelling framework is still applicable to reproduce an updated, relevant predictive model. In particular, the step-wise method for MLR input factor selection will remain cost-effective to identify an updated list of parameters that account for the output of the model analytically and objectively, as opposed to resorting to trial-and-error or personal judgement. Moreover, the generated model has confined the variability in data and leveraged standardized practices and processes based on analysis of data gathered from one steel fabricator. Further research and evaluation and analysis with the help of proposed framework and methodologies need to be preformed to replicate the success of implementing this research in similar application settings.

REFERENCES

Afsarian, F., Saber, A., Pourzangbar, A., Olabi, A. G., Khanmohammadi, M. A. (2018). "Analysis of Recycled Aggregates Effect on Energy Conservation using M5′ Model Tree Algorithm." Energy, 156, 264-277.

Akinwande, M., Dikko, H. and Samson, A. (2015). “Variance Inflation Factor: As a Condition for the Inclusion of Suppressor Variable(s) in Regression Analysis.”, Open Journal of Statistics, 5, 754-767.

Alfeld, L. E. (1988). Construction Productivity: On-Site Measurement and Management, McGraw-Hill, New York.

Barrett, B. E., and Gray, J. B. (1994). “A computational framework for variable selection in multivariate regression.” Statistics and computing, 4, 203–212.

Behnood, A., Behnood, V., Modiri Gharehveran, M., Alyamac, K. E. (2017). "Prediction of the Compressive Strength of Normal and High-Performance Concretes using M5P Model Tree Algorithm." Construction and Building Materials, 142, 199-207.

Berry, W. D. D., and Feldman, S. (1985). Multiple regression in practice. Quantitative applications in the social sciences. Sage Publication, Santa Barbara, California.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). Classification and regression trees. Wadsworth, Belmont.

Breusch, T. S., and Pagan, A. R. (1979). “A simple test for heteroscedasticity and random coefficient variation.” Econometrica: Journal of the econometric society, 47(5), 1287–1294.

Chan, W. H., and Lu, M. (2008). “Materials handling system simulation in precast viaduct construction: modeling, analysis, and implementation.” Journal of construction engineering and management, 10.1061/(ASCE)0733-9364(2008)134:4(300), 300–310.

Cheung, F. K. T., and Skitmore, M. (2006). “Application of cross validation techniques for modelling construction costs during the very early design stage.” Building and environment, 41(12), 1973–1990.

Choi, K., Haque, M., Lee, H. W., Cho, Y. K., and Kwak, Y. H. (2013). “Macroeconomic labour productivity and its impact on firm’s profitability.” Journal of the operational research society, Nature Publishing Group, 64(8), 1258–1268.

Choi, K., Kim, Y., Bae, J., and Lee, H. (2015). “Determining future maintenance costs of low-volume highway rehabilitation projects for incorporation into life-cycle cost analysis.” Journal of computing in civil engineering, 1–10.

Dawood, N. (1998). "Estimating Project and Activity Duration: A Risk Management Approach using Network Analysis." Constr. Manage. Econ., 16(1), 41-48.

Desai, V. S., and Joshi, S. (2010). "Application of decision tree technique to analyze construction project data." Proc., Information Systems, Springer Berlin Heidelberg, ICISTM 2010: Information Systems, Technology and Management, 304-313.

Deshpande, N., Londhe, S., Kulkarni, S. (2014). "Modeling Compressive Strength of Recycled Aggregate Concrete by Artificial Neural Network, Model Tree and Non-Linear Regression." International Journal of Sustainable Built Environment, 3(2), 187-198.

Draper, N. R., and Smith, H. (1998). Applied Regression Analysis, Wiley, New York, USA.

Dozzi, S. P., and Abourizk, S. M. (1993). Productivity in construction. Institute for Research in Construction. National Research Council, Ottawa, ON, Canada

Durbin, J., and Watson, G. S. (1951). "Testing for serial correlation in least squares regression." Biometrika, 38(1-2), 159–178.

El-abbasy, M. S., Senouci, A., Zayed, T., and Mirahadi, F. (2014). "Condition prediction models for oil and gas pipelines using regression analysis." 1–17.

El-Sawy, I., Hosny, H., and Razek, M. A. (2011). "A neural network model for construction projects site overhead cost estimating in Egypt." International journal of computer science issues, 8(3), 273–283.

Feng, W., Zhu, W., and Zhou, Y. (2010). "The application of genetic algorithm and neural network in construction cost estimate." Third international symposium on electronic commerce and security workshops (ISECS'10), 29–31, July 2010, 151–155.

Frank, E., Mayo, M., Kramer, S. (2015). "Alternating model trees." Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, 871-878.

Fox, J. (1991). Regression diagnostics. Sage Publications.

Gardner, B. J., Gransberg, D. D., David, J. H. (2016). "Reducing Data-Collection Efforts for Conceptual Cost Estimating at a Highway Agency." J. Constr. Eng. Manage., 142(11), 04016057.

Geman, S., Bienenstock, E., Doursat, R. (1992). "Neural Networks and the Bias/Variance Dilemma." Neural Comput., 4(1), 1-58.

Greene, W. H. (2008). Econometric analysis. Econometric analysis, Pearson Education, Upper Saddle River.

Gu, N. & London, K. (2010). Understanding and facilitating BIM adoption in the AEC industry. Automation in Construction, 19(8), 988-999.

Gujarati, D. N. (2004). Basic econometrics. New York, McGraw-Hill.

Hair, J. F. J., Black, W. C., Babin, B. J., and Anderson, R. E. (2010). Multivariate data analysis. Pearson Education, Upper Saddle River, New Jersey.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H. (2009). The WEKA Data Mining Software: An Update. SIGKDD Explorations, Volume 11, Issue 1.

Hendrickson, C. 2008. “Project Management for Construction”. *Department of Civil and Environmental Engineering, Carnegie Mellon University*, Pittsburgh, PA l52l3. Copyright C. Hendrickson 1998.

Heravi, G., and Eslamdoost, E. (2015). "Applying Artificial Neural Networks for Measuring and Predicting Construction-Labor Productivity." J. Constr. Eng. Manage., 141(10), 04015032.

Holiday, D. B., Ballard, J. E., and Mckeown, B. C. (1995). “PRESS-related statistics regression tools for cross-validation and case diagnostics.” Medicine and science in sports and excercise, 27(4), 612.

Hu X., Lu M., AbouRizk, S. (2014). "BIM-based data mining approach to estimating job man-hour requirements in structural steel fabrication." IEEE, 3399-3410.

Ivanescu A. E., Li P., George B., Brown A. W., Keith S. W., Raju D., Allison D. B. (2016). “The importance of prediction model validation and assessment in obesity and nutrition research.” International Journal of Obesity, 40, 887–894.

Jaillon, L., and Poon, C. S. (2014). “Life cycle design and prefabrication in buildings: A review and case studies in Hong Kong.” Autom. Constr., 39, 195–202.

Jakabsons, G. (2016), M5' regression tree, model tree, and tree ensemble toolbox for Matlab/Octave (ver. 1.7.0), Riga Technical University, Institute of Applied Computer Systems, Riga, Latvia

Jafarzadeh, R., Ingham, J. M., Walsh, K. Q., Hassani, N., and Ghodrati Amiri, G. R. (2015). “Using statistical regression analysis to establish construction cost models for seismic retrofit of confined masonry buildings.” Journal of construction engineering and management, 04014098.

Jafarzadeh, R., Wilkinson, S., González, V., Ingham, J. M., and Ghodrati Amiri, G. (2014). “Predicting seismic retrofit construction cost for buildings with framed structures using multilinear regression analysis.” Journal of construction engineering and management, 1–10.

Kaufman, R. L. (2013). Heteroskedasticity in regression. SAGE publicationss, Thousand Oaks, California.

King, G. (1986). “How not to lie with statistics: Avoiding common mistakes in quantitative political science.” American journal of political science, 30(3), 666–687.

Knowles, P. (1997). ‘‘Predicting labor productivity using neural networks.’’ MS thesis, University of Alberta, Edmonton, Alta., Canada.

Kutner, M. H., Nachtsheim, C. J., and Neter, J. (2004). Applied linear regression models. McGraw-Hill, New York.

Lee Min-Jae, Hanna, A. S., Loh Wei-Yin. (2004). "Decision Tree Approach to Classify and Quantify Cumulative Impact of Change Orders on Productivity." J. Comput. Civ. Eng., 18(2), 132-144.

Lee, J., Park, Y., Choi, C., Han, C. (2017). "BIM-Assisted Labor Productivity Measurement Method for Structural Formwork." Automation in Construction, 84, 121-132.

Leung, A. W. T., Tam, C. M., and Liu, D. K. (2001). “Comparative study of artificial neural networks and multiple regression analysis for predicting hoisting times of tower cranes.” Building and environment, 36(4), 457–467.

Lewis-Beck, M. S. (1978). “Stepwise regression: a caution.” Political methodology, 5(2), 213–240.

Li, Z., Shen, G. Q., and Xue, X. (2014). “Critical review of the research on the management of prefabricated construction.” Habitat Int., 43, 240–249.

Liddy, W., and Cross, J. (2002). "Conceptual Estimating, Design-Build and the Steel Fabricator." Modern Steel Construction, 42(10), 48-54.

Liu, W. (2010). Simultaneous inference in regression. Boca Raton. CRC Press.

Loh, W. Y. (2002), Regression trees with unbiased variable selection and interaction detection, Statistica Sinica, vol. 12, 361-386.

Lowe, D. J., Emsley, M. W., and Harding, A. (2006). "Predicting construction cost using multiple regression techniques." Journal of construction engineering and management, 132(7), 750–759.

Lu, M., AbouRizk, S. M., Hermann, U. H. (2000). "Estimating Labor Productivity using Probability Inference Neural Network." J. Comput. Civ. Eng., 14(4), 241-248.

Lu, M., AbouRizk, S., and Hermann, U. (2001), "Sensitivity analysis of neural networks in spool fabrication productivity studies", Journal of Computing in Civil Engineering, ASCE, Vol 15(4), 299-308.

Malerba, D., Esposito, F., Ceci, M., Appice, A. (2004). "Top-Down Induction of Model Trees with Regression and Splitting Nodes." IEEE Transactions on Pattern Analysis & Machine Intelligence, 26, 612-625.

Matlab 2016b (2016), Mathworks, Natick, United States, <https://www.mathworks.com/> (05/04/2017).

Mendenhall, W. M., Sincich, T. L. (2015), Statistics for Engineering and the Sciences Sixth Edition. Chapman and Hall/CRC

Mohsenijam, A., and Lu, M. (2016) “Achieving Sustainable Structural Steel Design by Estimating Fabrication Labor Cost Based on BIM Data,” Procedia Engineering, vol. 145, pp. 654–661.

Mohsenijam, A., Siu, M. F., Lu, M. (2017). "Modified Stepwise Regression Approach to Streamlining Predictive Analytics for Construction Engineering Applications." J. Comput. Civ. Eng., 31(3), 04016066.

Moody, J. (1994). "Prediction risk and architecture selection for neural networks." Proc., From Statistics to Neural Networks, Springer Berlin Heidelberg, 147-165.

Morgan, J. N., and Sonquist, J. A. (1963). "Problems in the Analysis of Survey Data, and a Proposal." Journal of the American Statistical Association, 58(302), 415-434.

Monteiro, A., and Poças Martins, J. (2013). "A Survey on Modeling Guidelines for Quantity Takeoff-Oriented BIM-Based Design." Automation in Construction, 35, 238-253.

Najafi, A., and Kong, R. T. L. (2015). “Productivity modeling of precast concrete installation using multiple regression analysis.” ARPN Journal of Engineering and Applied Sciences, 10(6), 2496-2503.

Omran, B. A., Qian, C., Ruoyu, J. (2016). "Comparison of Data Mining Techniques for Predicting Compressive Strength of Environmentally Friendly Concrete." J. Comput. Civ. Eng., 30(6), 04016029.

Quinlan, J. R. (1992). "Learning with Continuous Classes." Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, AI’92, Pages 343-348. World Science.

Randolph, T. H., Maloney, W. F., Malcolm, H. R., Smith, G. R., Handa, V. K., Sanders, S. R. (1990). "Modeling Construction Labor Productivity." J. Constr. Eng. Manage., 116(4), 705-726.

Rifat, S., and Rowings, J. E. (1998). "Construction Labor Productivity Modeling with Neural Networks." J. Constr. Eng. Manage., 124(6), 498-504.

Said, H. M., and Prathyaj, K. (2017). "Performance Measurement of Building Sheet-Metal Ductwork Prefabrication Under Batch Production Settings." J. Constr. Eng. Manage., 144(2), 04017107.

Salford Predictive Modeler 8. (2018). San Diego, California, United States: Salford Systems, a Mintab company.

Sargent, R. G. (2013). “Verification and validation of Simulation Models”, Journal of Simulation, 7:1, 12-24

Sawada, K., Shimizu, H., Matsuo, A., Sasaki, T., Yasui, T., Namba, A. (2006). "A simple estimation of fabrication cost and minimum cost design for steel frames." Proc., Proceedings of the Fourth China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems,89-94.

Seber, G. A. F., and Lee, A. J. (2003). Linear regression analysis. Wiley series in probability and statistics. Wiley, New York City, NY, United States.

Shen, Z., and Issa, R. R. A. (2010). "Quantitative Evaluation of the BIM-Assisted Construction Detailed Cost Estimates." Electron. J. Inf. Technol. Constr., 15, 234-257.

Silva, A., Dias, J. L., Gaspar, P. L., and de Brito, J. (2013). “Statistical models applied to service life prediction of rendered façades.” Automation in construction, Elsevier B.V., 30, 151–160.

Siu, M. F., Lu, M., and AbouRizk, S. (2014). "Strategies for Optimizing Labour Resource Planning on Plant Shutdown and Turnaround. " ASCE Construction Research Congress 2014, May 19–21, 2014, Atlanta, Georgia, United States, 1676–1685

Smith, H., and Draper, N. R. (1998). Applied regression analysis. Wiley, New York City, NY, United States.

Smith, S. D. (1999). “Earthmoving productivity estimation using linear regression techniques.” Journal of construction engineering and management, 125, 133–141.

Song, L. Y., and Abourizk, S. M. (2008). “Measuring and modeling labor productivity using historical data.” Journal of construction engineering and management, 134(10), 786–795.

Stephens, M. (1974). “EDF statistics for goodness of fit and some comparisons.” Journal of the american statistical association, 69(347), 730–737.

Stephens, M. A., Scholz, F. W. (1987). “K-Sample Anderson-Darling Tests ", Journal of the American Statistical Association, Vol. 82, No. 399, pp. 918-924.

Sweis, R. J., Sweis, G. J., Abu Hammad, A. A., Abu Rumman, M. (2009). "Modeling the Variability of Labor Productivity in Masonry Construction." Jordan Journal of Civil Engineering, 3(3), 197-212.

Taghados, H., Mashayekhi, A., Sherafat, B. (2016). “Automation of Construction Quantity Take-off: Using Building Information Modeling (BIM)”, Construction Research Congress 2016

Thompson, M. L. (1978). “Selection of variables in multiple regression: Part I. A review and evaluation.” International statisical review, 46(1), 1–19.

Verlinden, B., Duflou, J.R., Collin, P., Cattrysse, D. (2008). “Cost estimation for sheet metal parts using multiple regression and artificial neural networks: A case study”, International Journal of Production Economics, Volume 111, Issue 2, 2008, Pages 484-492.

Walasek, D., Barszcz, A. (2017). “Analysis of the adoption rate of Building Information Modeling [BIM] and its rate of Return on Investment [ROI]”, Modern Building Materials, Structures and Techniques, Procedia Engineering, 172, 1227-1234.

Warrian, P. (2010). The Importance of Steel Manufacturing to Canada: A Research Study, Munk School of Global Affairs, Toronto.

Wang, Y., and Witten, I. H. (1996). “Induction of Model Trees for Predicting Continuous Classes.” Proc. Ninth European Conf. Machine Learning.

Wang, G. C. S., and Jain, C. L. (2003). Regression analysis: modeling and forecasting. Graceway publishing company, New York.

Wiesenberger, G., (2011), Sustainability and the Structural Engineering, Practice Periodical on Structural Design and Construction, 16(4): 146-150

Wong, C. H. (2004). “Contractor performance prediction model for the United Kingdom construction contractor: study of logistic regression approach.” Journal of construction engineering and management, 130(5), 691–698.

Yeh, I. C. (2006). “Exploring concrete slump model using artificial neural networks.” Journal of computing in civil engineering, 10.1061/(ASCE)0887-3801(2006)20:3(217), 217–221.

Yeh, I. C. (2007). “Modeling slump flow of concrete using second-order regressions and artificial neural networks.” Cement and concrete composites, 29(6), 474–480.

Yeh, I. C. (2009). “Simulation of concrete slump using neural networks.” Proceedings of the ICE - construction materials, 10.1680/coma.2009.162.1.11, 11–18.

Yu, L., Lai, K. K., Wang, S., Huang, W. (2006). "A bias-variance-complexity trade-off framework for complex system modeling." Proc., Computational Science and its Applications - ICCSA 2006, Springer Berlin Heidelberg, 518-527.

APPENDIX (I) : MATLAB SOURCE CODE

This Chapter has the MATLAB source code used for data analysis. The MATLAB codes provided are for (1) developing MSR, (2) Checking Heteroscedasticity, (3) K-Fold Cross Validation, (4) PRESS Validation, and (5) M5 Model Tree.

5.5 DEVELOPING MSR (MODIFIED CODE FROM MATALB 2016B)

```
function model = stepwiselm(X,varargin) % [X, y | DS], start, ...
%STEPWISELM Create linear regression model by stepwise regression.
%   LM = STEPWISELM(DS,MODELSPEC) fits the model specified by MODELSPEC to
%   variables in the dataset/table DS, adds or removes terms by stepwise
%   regression and returns the linear model LM. MODELSPEC can be any of
%   the values accepted by the FITLM function. The default is 'constant' to
%   start with only the constant term in the model.
%
%   After the initial fit, the function examines a set of available terms,
%   and adds the best one to the model if an F-test for adding the term has
%   a p-value 0.05 or less. If no terms can be added, it examines the terms
%   currently in the model, and removes the worst one if an F-test for
%   removing it has a p-value 0.10 or greater. It repeats this process
%   until no terms can be added or removed. The function never removes the
%   constant term. It may add terms defined by MODELSPEC, as well as terms
%   that are two-way interactions among the predictors.
%
%   LM = STEPWISELM(X,Y) fits a linear regression model using the column
%   vector Y as a response variable and the columns of the matrix X as
%   predictor variables, performs stepwise regression, and returns the
%   final result as the linear model LM.
%
%   LM = STEPWISELM(X,Y,MODELSPEC) uses the model specified by MODELSPEC as
%   the initial model. See FITLM for valid MODELSPEC values.
%
%   LM = STEPWISELM(...,PARAM1,VAL1,PARAM2,VAL2,...) specifies one or more
%   of the following name/value pairs:
%
%      'Weights'          Vector of N non-negative weights, where N is the
%                         number of rows in DS or Y. Default is ones(N,1).
%      'VarNames'         Cell array of strings specifying the names to use
%                         for the columns in X. Default is {'x1','x2',...}
%                         for the predictors and 'y' for the response.
%                         Not allowed when fitting to a dataset/table.
%      'CategoricalVars'  Vector of integer or logical indices specifying
%                         the variables in DS or the columns in X that
%                         should be treated as categorical. Default is to
```

```
%                       treat DS variables as categorical if they are
%                       categorical, logical, or char arrays, or cell
%                       arrays of strings.
%     'Exclude'         Vector of integer or logical indices into the
%                       rows of DS, or X and Y, that should be excluded
%                       from the fit. Default is to use all rows.
%     'Intercept'       true (default) to include a constant term in the
%                       model, or false to omit it.
%     'PredictorVars'   A specification of the variables to use as
%                       predictors, either as a cell array of variable
%                       names, or a vector of integer or logical indices
%                       into the variables in DS or the columns in X.
%     'ResponseVar'     The response variable, specified either as a
%                       variable name or number.
%     'Lower'           A model specification defining the terms that
%                       cannot be removed from the model. Default
%                       'constant', meaning only the intercept.
%     'Upper'           A model specification defining the terms
%                       available to be added to the model. Default
%                       'interactions' for pairwise interaction terms.
%     'Criterion'       The criterion to be used in choosing terms to add
%                       or remove, chosen from 'SSE' (default), 'AIC',
%                       'BIC', 'Rsquared', 'AdjRsquared'.
%     'PEnter'          For the 'SSE' criterion, a value E such that a
%                       term may be added if its p-value is less than or
%                       equal to E. For the other criteria, a term may be
%                       added if the improvement in the criterion is at
%                       least E.
%     'PRemove'         For the 'SSE' criterion, a value R such that a
%                       term may be removed if its p-value is greater or
%                       equal to R. For the other criteria, a term may be
%                       added if it reduces the criterion no more than R.
%     'NSteps'          The maximum number of steps that may be taken,
%                       starting from the initial model. Default is no
%                       limit on the number of steps.
%     'Verbose'         An integer from 0 to 2 controlling the display of
%                       information. Verbose=1 (the default) displays the
%                       action taken at each step. Verbose=2 also
%                       displays the actions evaluated at each step.
%                       Verbose=0 suppresses all display.
%
%   The following table shows the default 'PEnter' and 'PRemove' values for
%   the different criteria, and indicates which must be larger than the
%   other:
%
%      Criterion     PEnter   PRemove    Compared against
%      'SSE'         0.05   < 0.10       p-value for F test
%      'AIC'         0      < 0.01       change in AIC
%      'BIC'         0      < 0.01       change in BIC
%      'Rsquared'    0.1    > 0.05       increase in R-squared
%      'AdjRsquared' 0      > -0.05      increase in adjusted R-squared
%
%   Example:
%      % Perform stepwise regression, starting from the constant model
%      % (intercept only), adding linear terms as required.
%      load hald
%      lm = stepwiselm(ingredients,heat,'constant','upper','linear')
```

5.6 CHECKING HETEROSCEDASTICITY

```
function [HetResult,secondModel,Xw,Yw,Nres] = Heter(X,Y)
%This function is to check if the model passes Heteroscedasticity.
%The function takes the matrix of input and output and test if the residuals
are pass the HeterTEST.
[m n] = size(X);
Xreg=[ones(m,1) X];

InitailModel = fitlm(X,Y);
res = InitailModel.Residuals.Raw;
Yhat = InitailModel.Fitted;
    TEST=TestHet(res, X, '-BPK')

res2= res.^2;
Hetreg=stepwiselm(X,res2,'quadratic');
Hettest=anova(Hetreg,'summary');
if Hettest.pValue(2)>0.05;
    HetResult='No Heteroscedasticity';
else
    HetResult='Heteroscedasticity';

    PRE = Hetreg.Fitted;
    Xw=zeros(m,n+1);
    for i=1:n+1;
      Xw(:,i)=Xreg(:,i)./PRE;
    end
     Yw=Y./PRE;
   secondModel = fitlm(Xw,Yw,'Intercept',false);
    %secondModel = fitlm(Xw,Yw)
    Nres=Y-Xreg*secondModel.Coefficients.Estimate
   % NYhat=Xreg*secondModel.Coefficients.Estimate;
    %Nres=secondModel.Residuals.Raw;
    NYhat=secondModel.Fitted;
    NTEST=TestHet(Nres, X, '-BPK')
    %scatter(res,Yhat,'b');hold on;scatter(Nres,NYhat,'g')
    %scatter(res,Yhat,'b')
    %scatter(Nres,NYhat,'g')
    %[bw,se,pval,inmodel,stats,nextstep,history] = stepwisefit(Xw,Yw)
   % Wpredicted=[stats.intercept bw'.*inmodel]*Xreg';
   % Wres=Y-Wpredicted';
   % Wres2= Wres.^2;
   % Wp=[Wpredicted',Wpredicted'.^2];
end

function [pVal,Chi2] = TestHet(Res, X, Whichtest, Yhat)

% TESTHET Tests wether heteroskedasticity affects data. Need 'regstats' and
'chi2cdf' (Stat TB).
%
%   PVAL = TESTHET(RES, X, WHICHTEST, YHAT)
%   Inputs:
```

```
%   - Res: residuals obtained by regressing Y on x1, x2 etc...(1) It can be a
numeric 'n-by-1' vector or
%          a 'n-by-p' matrix with 'p' residuals obtained from different
regressions. The # of obs. is 'n'.
%   - X: predictors suspected of causing heteroskedasticity. Not necessarily
all those in (1). Same format as
%        Res.
%   - Whichtest: test chosen in format string.
%                a. Breush-Pagan, Koenker modification   -->  -BPK
(Breush-Pagan 1979; Koenker 1981)
%                b. White                                -->  -W
(White 1980b)
%                c. White, Wooldridge special case       -->  -Ws
(White 1980b; Wooldridge 2006, p.286)
%   [OPTIONAL]
%   - Yhat: only for '-Ws' test. Fitted values from (1). Same format as Res.
%
%   Output:
%   A '1-by-p' array with p-values.
%
%   EXAMPLE:
%       1. Regress Y on x1, x2 --> regstats(Y, [x1 x2], 'linear', {'r',
'yhat'})
%       2. Test with -Ws:
%              TestHet(r,[x1, x2], '-Ws', yhat)
%
% Ninputs
error(nargchk(3,4,nargin))
% Yhat (for White simpified case)
if strcmp(Whichtest, '-Ws')
   if nargin == 3
        error('TestHet:YhatMissing','Can''t perform -Ws test without Yhat.')
   end
elseif nargin == 4;
    warning('TestHet:InpOverSpec', 'Performing -W test. Yhat not required.')
else
    Yhat = ones(size(Res,1)); % for check purposes
end
% Numeric format
if ~isnumeric(X) || ~isnumeric(Res) || ~isnumeric(Yhat)
    error('TestHet:NumericFormat', 'Res, X and Yhat (if specified) must be
numeric.')
end
% Whichtest
if ischar(Whichtest)
    if all(~strcmp(Whichtest, {'-BPK','-W','-Ws'}))
        error('TestHet:WhichtestNotAllowed','Whichtest: choose among those
allowed.')
    end
else
    error('TestHet:WhichtestNotString','Whichtest must be a string.')
end
% Nobservations
if any(diff(cellfun(@(x) size(x,1), {Res,X,Yhat})))
    error('TestHet:NumberObservations','Res, X and Yhat (if specified must
have the same number of observations')
end
```

```
% STEP 1: inputs manipulation
% --------------------------
Res2 = Res.^2;                                                  % Squared
residuals
if nargin == 4;
    Yhat2 = Yhat.^2;                                            % Squared
Yhat (for -Ws test only)
end
Nseries = size(Res,2);                                          % # of
series to test
pVal = NaN(1,Nseries);                                          %
Preallocation

% STEP 2: settings
%-----------------
model = 'linear'; Regressors = X;                               % Default
settings

switch Whichtest                                                %
Specific settings
    % [-BPK] Breush-Pagan
    case '-BPK'
        df = size(X,2); % degrees of freedom
    % [-WH] White
    case '-W'
        model = 'quadratic';
        % For degrees of freedom don't take the "constant".
        % Reference on the interaction form : 'x2fx'.
        df = size(X,2)*2 + max(cumsum(1:size(X,2))) - size(X,2);
    % [-Ws] White special case
    case '-Ws'
        % Degrees of freedom fixed; the terms are always Yhat and Yhat^2.
        df = 2;
end

% STEP 3: p-values
% ----------------
% [1] LOOP for Nseries
for s = 1:Nseries
    % [2a] CONDITION if Ws test, 'Regressors' are combined matrixes
    if strcmpi(Whichtest, '-Ws'); Regressors = [Yhat(:,s),Yhat2(:,s)]; end;
%[2a]
    % [2b] CONDITION Regressors+1 must be < Nobserv
    if df+1 < sum(~isnan(any(Regressors,2)+ Res2(:,s)))
        % 1. R^2res^2: res^2 on the regression terms
        Temp = regstats(Res2(:,s), Regressors, model, {'rsquare'});
        % 2. pVal = 1-cdf(LM statistic, df) from a Chi^2 distribution.
        %    Where LM statistic = R^2res^2 * #obs
        pVal(1,s) = 1-chi2cdf(Temp.rsquare*nnz(~isnan(Res2(:,s))),df);
        Chi2(1,s)=Temp.rsquare*nnz(~isnan(Res2(:,s)));
    end % [2b]

end % [1]

end
```

5.7 K-FOLD VALIDATION

```
function [ ESS,SSE,res2 ] = Crossvalidation2(X,Y,k)
%Get the WLS result (Coeficient and weighted variables).
% a is result of heteroscedasiticity, Model is the regression under PRESS
% test, XX is weighted X, and YY is weighted Y.
[HetResult,TEST,chi,secondModel,Xw,Yw,res,Nres,NNres]=HetWLS(X,Y);
[m,n] = size(Xw);
% Pratitioning an array of data for k-fold test
c = cvpartition(m,'kfold',k);

%running the crossvalidation
for i=1:k
% for each set of validation data we need to find the Tr (training array)
% and Te (testing array)
Tr=training(c,i);
Te=test(c,i);
[f,g]=size(Tr(Tr~= 0));
Xt=[ones(m,1) X];
%Input data need to be partitioned. To do so the data XTr and YTr are used
%for training, and the XTe and YTe are used for testing.
XTr=zeros(f,n);
XTe=zeros(m-f,n);
    for j=1:n %for each column
        %Partitioning Xw
        XTrain=Tr.*Xw(:,j);
        XTest=Te.*Xt(:,j);
        % Zero values are removed for correct regression
        XTr(:,j)=XTrain((Tr~= 0));
        XTe(:,j)=XTest((Te~= 0));
    end
%Partitioning YY
YTr=Tr.*Yw;
YTe=Te.*Y;
% Zero values are removed for correct regression
YTr=YTr((Tr~= 0));
YTe=YTe((Te~= 0));
% Fitting the regression with the training set
TrainedModel = fitlm(XTr,YTr);

% Calculating test set predicted values
XTeReg=[ones(m-f,1) XTe];
Yhat=XTeReg*TrainedModel.Coefficients.Estimate;

% PRESS residuals
res=Yhat-YTe;

%Saving Press residuals
res2(1,i)=sumsqr(res);
end
ESS=sum(res2);
SSE=sumsqr(Nres);
end
```

5.8 PRESS VALIDATION

```
function [SSE,PRESS,ress] = PRESS2(X,Y)

%Get the WLS result (Coeficient and weighted variables).
% a is result of heteroscedasiticity, Model is the regression under PRESS
% test, XX is weighted X, and YY is weighted Y.
[a,Model,XX,YY,Nres]=HetWLS(X,Y);
[m,n] = size(XX);

% Pratitioning an array of data for PRESS test
c = cvpartition(m,'LeaveOut');

%running the crossvalidation
for i=1:m
% for each set of validation data we need to find the Tr (training array)
% and Te (testing array)
Tr=training(c,i);
Te=test(c,i);
[f,g]=size(Tr(Tr~= 0));
Xt=[ones(m,1) X];
%Input data need to be partitioned. To do so the data XTr and YTr are used
%for training, and the XTe and YTe are used for testing.
XTr=zeros(f,n);
XTe=zeros(m-f,n);
    for j=1:n
        %Partitioning XX
        XTrain=Tr.*XX(:,j);
        XTest=Te.*Xt(:,j);
        % Zero values are removed for correct regression
        XTr(:,j)=XTrain((Tr~= 0));
        XTe(:,j)=XTest((Te~= 0));
    end
%Partitioning YY
YTr=Tr.*YY;
YTe=Te.*Y;

% Zero values are removed for correct regression
YTr=YTr((Tr~= 0));
YTe=YTe((Te~= 0));
% Fitting the regression with the training set
TrainedModel = fitlm(XTr,YTr,'Intercept',false);
% Calculating test set predicted values
Yhat=XTe*TrainedModel.Coefficients.Estimate;

% PRESS residuals
res=Yhat-YTe;
%Saving Press residuals
ress(1,i)=res;
end
PRESS=sumsqr(ress);
SSE=sumsqr(Nres);
end
```

5.9 M5 MODEL TREE (JEKABSONS G. 2016)

```
function [model, time, ensembleResults] = m5pbuild(Xtr, Ytr, trainParams, ...
    isBinCat, trainParamsEnsemble, keepNodeInfo, verbose)
% m5pbuild
% Builds M5' regression tree, model tree, or ensemble of trees. Trees can
% also be linearized into decision rules. For tree ensembles, can also
% assess input variable importances as well as provide data for ensemble
% interpretation.
%
% Call:
%   [model, time, ensembleResults] = m5pbuild(Xtr, Ytr, trainParams, ...
%       isBinCat, trainParamsEnsemble, keepNodeInfo, verbose)
%
% All the input arguments, except the first two, are optional. Empty values
% are also accepted (the corresponding defaults will be used).
%
% Input:
%   Xtr, Ytr      : Xtr is a matrix with rows corresponding to
%                   observations and columns corresponding to input
%                   variables. Ytr is a column vector of response values.
%                   Input variables can be continuous, binary, as well as
%                   categorical (indicated using isBinCat). All values must
%                   be numeric. Categorical variables with more than two
%                   categories will be automatically replaced with
%                   synthetic binary variables (in accordance with the M5'
%                   method). Missing values in Xtr must be indicated as
%                   NaN.
%   trainParams   : A structure of training parameters for the algorithm.
%                   If not provided, defaults will be used (see function
%                   m5pparams for details).
%   isBinCat      : A vector of flags indicating type of each input
%                   variable - either continuous (false) or categorical
%                   (true) with any number of categories, including binary.
%                   The vector should be of the same length as the number
%                   of columns in Xtr. m5pbuild then detects all the
%                   actually possible values for categorical variables from
%                   the training data. Any new values detected later, i.e.,
%                   during prediction, will be treated as NaN. By default,
%                   the vector is created with all values equal to false,
%                   meaning that all the variables are treated as
%                   continuous.
%   trainParamsEnsemble : A structure of parameters for building ensemble
%                   of trees. If not provided, a single tree is built. See
%                   function m5pparamsensemble for details. This can also
%                   be useful for variable importance assessment. See
%                   user's manual for examples of usage.
%                   Note that the ensemble building algorithm employs
%                   random number generator for which you can set seed
%                   before calling m5pbuild.
%   keepNodeInfo  : Whether to keep models (in model trees) and response
%                   values (in regression trees) in interior nodes of
%                   trees. And whether to keep indices of training
%                   observations that reached each node and standard
```

```
%                   deviation of each node. These are useful for further
%                   analysis and plotting. Default value = true. If set to
%                   false, the information is removed from the trees so
%                   that the structure takes up less memory. Note that
%                   interior nodes of smoothed trees will not contain
%                   models or response values regardless of the value of
%                   this parameter because only the models in the leaves
%                   are smoothed. Also note that the standard deviations
%                   are saved before doing smoothing.
%   verbose       : Whether to output additional information to console.
%                   (default value = true)
%
% Output:
%   model         : A single M5' tree (or a decision rule set) or a cell
%                   array of M5' trees (or decision rule sets) if an
%                   ensemble is built. A structure defining one tree (or a
%                   decision rule set) has the following fields:
%     binCat      : Information regarding original (continuous / binary /
%                   categorical) input variables, transformed (synthetic
%                   binary) input variables, possible values for
%                   categorical input variables and other information.
%     trainParams : A structure of training parameters for the algorithm
%                   (updated if any values are chosen automatically).
%     tree, rules, outcomes : Structures and arrays defining either the
%                   built tree or the generated decision rules.
%   time          : Algorithm execution time (in seconds).
%   ensembleResults : A structure of results for ensembles of trees or
%                   decision rules. Not available for Extra-Trees. The
%                   structure has the following fields:
%     OOBError    : Out-of-bag estimate of prediction Mean Squared Error of
%                   the ensemble after each new tree is built. The number
%                   of rows is equal to the number of trees built. OOBError
%                   is available only if getOOBError in trainParamsEnsemble
%                   is set to true. Note that it's possible to calculate
%                   mean out-of-bag predictions (and therefore out-of-bag
%                   errors for each individual training data observation)
%                   by summing the columns of OOBContrib.
%     OOBIndices  : Logical matrix. For each tree (column) indicates which
%                   observations were out-of-bag (and thus used in
%                   computing OOBError). The number of rows in OOBIndices
%                   is equal to the number of rows in Xtr and Ytr.
%                   OOBIndices is available only if getOOBError or
%                   getOOBContrib in trainParamsEnsemble is set to true.
%     OOBNum      : Number of times observations were out-of-bag (and thus
%                   used in computing OOBError). The length of OOBNum is
%                   equal to the number of rows in Xtr and Ytr. OOBNum is
%                   available only if getOOBError or getOOBContrib in
%                   trainParamsEnsemble is set to true.
%     OOBContrib  : A matrix allowing interpreting ensembles in accordance
%                   with the Forest Floor methodology (Welling et al.,
%                   2016). See also example of usage in Section 3.2 of
%                   user's manual.
%                   It is a matrix of contributions of each input variable
%                   to the response for each Xtr row in terms of response
%                   value changes along the prediction path of a tree
%                   (averaged over the whole ensemble) so that Yoob =
%                   in-bag_mean + x1_contribution + x2_contribution + ... +
```

```
%                   xn_contribution, where Yoob is prediction of response
%                   for out-of-bag observation. OOBContrib has the same
%                   number of columns as Xtr plus one, the last column
%                   being the in-bag response mean. The sum of columns of
%                   OOBContrib is equal to Yoob of the whole ensemble for
%                   each row of Xtr.
%                   OOBContrib is available only if getOOBContrib in
%                   trainParamsEnsemble is set to true.
%                   Note that it's also possible to compute contributions
%                   and explain predictions for new data (including with
%                   single trees) - see function m5ppredict.
%     varImportance : Variable importance assessment. Calculated when
%                   out-of-bag data of a variable is permuted. A matrix
%                   with four rows and as many columns as there are columns
%                   in Xtr. First row is the average increase of out-of-bag
%                   Mean Absolute Error (MAE), second row is standard
%                   deviation of the average increase of MAE, third row is
%                   the average increase of out-of-bag Mean Squared Error
%                   (MSE), fourth row is standard deviation of the average
%                   increase of MSE. The final variable importance estimate
%                   is often calculated by dividing each MAE or MSE by the
%                   corresponding standard deviation. Bigger values then
%                   indicate bigger importance of the corresponding
%                   variable. See user's manual for example of usage.
%                   varImportance is available only if getVarImportance in
%                   trainParamsEnsemble is > 0.

if nargin < 2
    error('Not enough input arguments.');
end

if isempty(Xtr) || isempty(Ytr)
    error('Training data is empty.');
end
if iscell(Xtr) || iscell(Ytr)
    error('Xtr and Ytr should not be cell arrays.');
end
[n, mOriginal] = size(Xtr); % number of observations and number of input
variables
if size(Ytr,1) ~= n
    error('The number of rows in Xtr and Ytr should be equal.');
end
if size(Ytr,2) ~= 1
    error('Ytr should have one column.');
end
if any(any(isnan(Ytr)))
    error('Cannot handle NaNs in Ytr.');
end

if (nargin < 3) || isempty(trainParams)
    trainParams = m5pparams();
else
    trainParams.minLeafSize = max(1, trainParams.minLeafSize);
    if (trainParams.minLeafSize == 1) && trainParams.prune
        error('M5'' does not allow minLeafSize=1 if pruning is enabled.');
    end
```

```
    trainParams.minParentSize = max(trainParams.minLeafSize * 2,
trainParams.minParentSize);
    if (trainParams.extractRules < 2)
        trainParams.smoothingK = max(0, trainParams.smoothingK);
    else
        if trainParams.smoothingK > 0
            warning('Smoothing for M5''Rules method is always disabled.');
        end
        trainParams.smoothingK = 0;
    end
    trainParams.splitThreshold = max(0, trainParams.splitThreshold);
    trainParams.maxDepth = max(0, floor(trainParams.maxDepth));
    trainParams.extractRules = max(0, min(2,
floor(trainParams.extractRules)));
end
if (nargin < 4) || isempty(isBinCat)
    isBinCat = false(1,mOriginal);
else
    isBinCat = isBinCat(:)'; % force row vector
    if length(isBinCat) ~= mOriginal
        error('The number of elements in isBinCat should be equal to the
number of columns in Xtr.');
    end
end
if (nargin < 5)
    trainParamsEnsemble = [];
else
    if (~isempty(trainParamsEnsemble)) && trainParamsEnsemble.extraTrees &&
(trainParams.prune || trainParams.modelTree)
        error('Pruning and model trees are not available for Extra-Trees.');
    end
end
if (nargin < 6) || isempty(keepNodeInfo)
    keepNodeInfo = true;
    if trainParams.smoothingK > 0
        keepInteriorModels = false; % forcing
    else
        keepInteriorModels = true;
    end
else
    keepInteriorModels = false;
    keepNodeInfo = false;
end
if (nargin < 7) || isempty(verbose)
    verbose = true;
end

binCat = isBinCat .* 2;
% Transform categorical variables into a number of synthetic binary variables
binCatVals = {};
if any(binCat >= 2)
    binCatNewNum = [];
    binCatCounter = 0;
    Xnew = [];
    model.binCat.varMap = {};
    for i = 1 : mOriginal
        if binCat(i) >= 2
```

```
            XX = Xtr(:,i);
            u = unique(XX(~isnan(XX))); % no NaNs, unique, sorted
            if size(u,1) > 2
                model.binCat.varMap = [model.binCat.varMap
(size(binCatNewNum,2)+1) : (size(binCatNewNum,2)+size(u,1)-1)];
                avg = zeros(size(u,1),1);
                for j = 1 : size(u,1)
                    avg(j) = mean(Ytr(Xtr(:,i) == u(j)));
                end
                [~, ind] = sort(avg);
                u = u(ind);
                Xb = zeros(n,size(u,1)-1);
                for j = 1 : n
                    if isnan(Xtr(j,i))
                        Xb(j,:) = NaN;
                    else
                        Xb(j, 1 : find(Xtr(j,i) == u) - 1) = 1;
                    end
                end
                Xnew = [Xnew Xb];
                binCatNewNum = [binCatNewNum repmat(size(u,1),1,size(u,1)-
1)];
            else
                Xnew = [Xnew Xtr(:,i)];
                binCatNewNum = [binCatNewNum 2];
                model.binCat.varMap = [model.binCat.varMap
size(binCatNewNum,2)];
            end
            binCat(i) = size(u,1);
            binCatCounter = binCatCounter + 1;
            binCatVals{binCatCounter} = u;
            if binCat(i) >= 50
                warning(['Categorical variable #' num2str(i) ' has '
num2str(binCat(i)) ' unique values.']);
            end
        else
            Xnew = [Xnew Xtr(:,i)];
            binCatNewNum = [binCatNewNum 0];
            model.binCat.varMap = [model.binCat.varMap size(binCatNewNum,2)];
        end
    end
    Xtr = Xnew;
    model.binCat.catVals = binCatVals;
else
    binCatNewNum = binCat;
    model.binCat.varMap = num2cell(1:mOriginal);
end

model.binCat.binCat = binCat;
model.binCat.binCatNew = binCatNewNum >= 2; % 0 for continuous; 1 for binary
if any(model.binCat.binCatNew)
    % this is used later for printing/plotting of the trees/rules
    model.binCat.minVals = min(Xtr);
    if (trainParams.extractRules > 0)
        model.binCat.maxVals = max(Xtr);
    end
end
```

```
model.trainParams = trainParams;

if verbose
    if trainParams.modelTree, str = 'model'; else str = 'regression'; end
    if isempty(trainParamsEnsemble)
        if trainParams.extractRules == 0
            disp(['Growing M5'' ' str ' tree...']);
        else
            disp('Generating rule set...');
        end
    else
        if trainParams.extractRules == 0
            disp(['Growing M5'' ' str ' tree ensemble...']);
        else
            disp('Growing ensemble of rule sets...');
        end
    end
end

origWarningState = warning;
if exist('OCTAVE_VERSION', 'builtin')
    warning('off', 'Octave:nearly-singular-matrix');
    warning('off', 'Octave:singular-matrix');
else
    warning('off', 'MATLAB:nearlySingularMatrix');
    warning('off', 'MATLAB:singularMatrix');
end
ttt = tic;

% For the original binary and continuous variables beta = 1
% For synthetic binary variables created from original categorical variables
beta < 1
beta = exp(7 * (2 - max(2, binCatNewNum)) / n);

if isempty(trainParamsEnsemble)

    sd = stdMy(Ytr);
    numNotMissing = sum(~isnan(Xtr),1); % number of non-missing values for
each variable
    model = buildTree(model, Xtr, Ytr, sd, numNotMissing, binCatNewNum, beta,
[], [], false, keepInteriorModels, keepNodeInfo);

    ensembleResults = [];

else

    if trainParamsEnsemble.numVarsTry < 1
        if trainParamsEnsemble.numVarsTry < 0
            trainParamsEnsemble.numVarsTry = mOriginal / 3;
        else
            trainParamsEnsemble.numVarsTry = mOriginal;
        end
    end
```

```
    trainParamsEnsemble.numVarsTry = min(mOriginal, max(1,
floor(trainParamsEnsemble.numVarsTry)));

    if ~trainParamsEnsemble.extraTrees
        % Random Forests or Bagging

        if round(trainParamsEnsemble.inBagFraction * n) < 1
            error('trainParamsEnsemble.inBagFraction too small. In-bag set is
empty.');
        end
        if (~trainParamsEnsemble.withReplacement) &&
(round(trainParamsEnsemble.inBagFraction * n) >= n)
            error('trainParamsEnsemble.inBagFraction too big. Out-of-bag set
is empty.');
        end

        modelBase = model;
        models = cell(trainParamsEnsemble.numTrees, 1);

        if (~trainParamsEnsemble.getOOBError) &&
(trainParamsEnsemble.getVarImportance == 0) &&
(~trainParamsEnsemble.getOOBContrib)
            ensembleResults = [];
        else
            if trainParamsEnsemble.getOOBError ||
trainParamsEnsemble.getOOBContrib
                OOBNum = zeros(n, 1);
                ensembleResults.OOBIndices = false(n,
trainParamsEnsemble.numTrees);
            end
            if trainParamsEnsemble.getOOBContrib
                OOBContrib = zeros(n, mOriginal + 1);
            end
            if trainParamsEnsemble.getOOBError
                OOBPred = zeros(n, 1);
                ensembleResults.OOBError = NaN(trainParamsEnsemble.numTrees,
1);
            end
            if trainParamsEnsemble.getVarImportance > 0
                diffOOBMAE = NaN(trainParamsEnsemble.numTrees, mOriginal);
                diffOOBMSE = NaN(trainParamsEnsemble.numTrees, mOriginal);
                ensembleResults.varImportance = zeros(4, mOriginal); %
increase in MAE, SD, increase in MSE, SD
            end
        end

        % for each tree
        for t = 1 : trainParamsEnsemble.numTrees
            if verbose && (trainParamsEnsemble.verboseNumIter > 0) && ...
                    (mod(t, trainParamsEnsemble.verboseNumIter) == 0)
                if trainParams.extractRules == 0
                    fprintf('Growing tree #%d...\n', t);
                else
                    fprintf('Generating rule set #%d...\n', t);
                end
            end
```

```
            % sampling
            if trainParamsEnsemble.withReplacement
                idx = randi(n, round(trainParamsEnsemble.inBagFraction * n),
1);
                X = Xtr(idx,:);
                Y = Ytr(idx,1);
            else
                perm = randperm(n);
                idx = perm(1:round(trainParamsEnsemble.inBagFraction * n));
                X = Xtr(idx,:);
                Y = Ytr(idx,1);
            end

            if t > 1
                model = modelBase;
            end

            sd = stdMy(Y);
            numNotMissing = sum(~isnan(X),1); % number of non-missing values
for each variable
            model = buildTree(model, X, Y, sd, numNotMissing, binCatNewNum,
beta, ...
                trainParamsEnsemble.numVarsTry, mOriginal, false,
keepInteriorModels, keepNodeInfo);

            % additional calculations, if asked
            if trainParamsEnsemble.getOOBError ||
(trainParamsEnsemble.getVarImportance > 0) ||
trainParamsEnsemble.getOOBContrib
                idxoob = true(n,1);
                idxoob(idx) = false;
                idxoob = find(idxoob);
                if ~isempty(idxoob) % test for the unlikely case when out-of-
bag set is empty
                    Xoob = Xtr(idxoob,:);
                    Yq = zeros(size(Xoob,1),1);

                    if trainParamsEnsemble.getOOBContrib
                        for i = 1 : size(Xoob,1)
                            [Yq(i), OOBContrib2] = predictsingle(model,
Xoob(i,:), modelBase.binCat.varMap);
                            OOBContrib(idxoob(i),:) = OOBContrib(idxoob(i),:)
+ OOBContrib2;
                        end
                    else
                        for i = 1 : size(Xoob,1)
                            Yq(i) = predictsingle(model, Xoob(i,:));
                        end
                    end

                    if trainParamsEnsemble.getOOBError ||
trainParamsEnsemble.getOOBContrib
                        ensembleResults.OOBIndices(idxoob,t) = true;
                        OOBNum(idxoob) = OOBNum(idxoob) + 1;
                    end
```

```
                    if trainParamsEnsemble.getOOBError
                        idxExist = OOBNum ~= 0;
                        OOBPred(idxoob) = OOBPred(idxoob) + Yq;
                        ensembleResults.OOBError(t,1) =
mean(((OOBPred(idxExist) ./ OOBNum(idxExist)) - Ytr(idxExist)) .^ 2);
                        if verbose && (trainParamsEnsemble.verboseNumIter >
0) && ...
                                (mod(t, trainParamsEnsemble.verboseNumIter)
== 0)
                            if trainParams.extractRules == 0
                                fprintf('Out-of-bag MSE with %d trees:
%.5g\n', t, ensembleResults.OOBError(t,1));
                            else
                                fprintf('Out-of-bag MSE with %d rule sets:
%.5g\n', t, ensembleResults.OOBError(t,1));
                            end
                        end
                    end

                    if trainParamsEnsemble.getVarImportance > 0
                        Yqtdiff = Yq - Ytr(idxoob);
                        for v = 1 : mOriginal
                            for iPerm =
1:trainParamsEnsemble.getVarImportance
                                Xoobpert = Xoob;
                                idxoobpert =
idxoob(randperm(size(idxoob,1)),1);
                                % Perturb OOB variables that correspond to
the original vth variable
                                for vnew = model.binCat.varMap{v}
                                    Xoobpert(:,vnew) = Xtr(idxoobpert,vnew);
                                end
                                Yqpert = zeros(size(Xoobpert,1),1);
                                for i = 1 : size(Xoobpert,1)
                                    Yqpert(i) = predictsingle(model,
Xoobpert(i,:));
                                end
                                Yqptdiff = Yqpert - Ytr(idxoob);
                                if iPerm == 1
                                    diffOOBMAE(t,v) = mean(abs(Yqptdiff)) -
mean(abs(Yqtdiff));
                                    diffOOBMSE(t,v) = mean(Yqptdiff .^ 2) -
mean(Yqtdiff .^ 2);
                                else
                                    diffOOBMAE(t,v) = diffOOBMAE(t,v) +
mean(abs(Yqptdiff)) - mean(abs(Yqtdiff));
                                    diffOOBMSE(t,v) = diffOOBMSE(t,v) +
mean(Yqptdiff .^ 2) - mean(Yqtdiff .^ 2);
                                end
                            end
                            if trainParamsEnsemble.getVarImportance > 1
                                diffOOBMAE(t,v) = diffOOBMAE(t,v) /
trainParamsEnsemble.getVarImportance;
                                diffOOBMSE(t,v) = diffOOBMSE(t,v) /
trainParamsEnsemble.getVarImportance;
                            end
                        end
```

```
                    end

                end
            end

            models{t} = model;
        end % end of loop through all trees
        model = models;
        if trainParamsEnsemble.getOOBError ||
trainParamsEnsemble.getOOBContrib
            ensembleResults.OOBNum = OOBNum;
        end
        if trainParamsEnsemble.getOOBContrib
            ensembleResults.OOBContrib = OOBContrib ./ repmat(OOBNum, 1,
mOriginal + 1);
        end
        if trainParamsEnsemble.getVarImportance > 0
            ensembleResults.varImportance(1,:) = mean(diffOOBMAE, 1);
            ensembleResults.varImportance(2,:) = std(diffOOBMAE, 1, 1);
            ensembleResults.varImportance(3,:) = mean(diffOOBMSE, 1);
            ensembleResults.varImportance(4,:) = std(diffOOBMSE, 1, 1);
        end

    else % if extraTrees
        % Extra-Trees

        modelBase = model;
        models = cell(trainParamsEnsemble.numTrees, 1);
        ensembleResults = [];
        sd = stdMy(Ytr);
        numNotMissing = sum(~isnan(Xtr),1); % number of non-missing values
for each variable
        % for each tree
        for t = 1 : trainParamsEnsemble.numTrees
            if verbose && (trainParamsEnsemble.verboseNumIter > 0) && ...
                    (mod(t, trainParamsEnsemble.verboseNumIter) == 0)
                if trainParams.extractRules == 0
                    fprintf('Growing tree #%d...\n', t);
                else
                    fprintf('Generating rule set #%d...\n', t);
                end
            end
            if t > 1
                model = modelBase;
            end
            model = buildTree(model, Xtr, Ytr, sd, numNotMissing,
binCatNewNum, beta, ...
                trainParamsEnsemble.numVarsTry, mOriginal, true,
keepInteriorModels, keepNodeInfo);

            models{t} = model;
        end
        model = models;
    end % end of if extraTrees

end
```

```
time = toc(ttt);
if verbose
    if isempty(trainParamsEnsemble)
        printinfo(model);
    end
    fprintf('Execution time: %0.2f seconds\n', time);
end
warning(origWarningState);
end

%=========================================================================

function model = buildTree(model, X, Y, sd, numNotMissing, binCatNewNum,
beta, numVarsTry, mOriginal, extraTrees, keepInteriorModels, keepNodeInfo)
% Builds a tree. If asked, extracts decision rules and returns them instead
of the tree.
if model.trainParams.extractRules == 0
    % This is normal execution for building M5' trees.
    % Growing the tree
    model.tree = splitNode(X, Y, 1:size(Y,1), 0, sd, numNotMissing,
binCatNewNum, model.trainParams, beta, numVarsTry, mOriginal,
model.binCat.varMap, extraTrees, keepNodeInfo);
    % Pruning the tree and/or filling it with models or mean values
    model.tree = pruneNode(model.tree, X, Y, model.trainParams,
keepNodeInfo);
    if model.trainParams.smoothingK > 0
        totalAttrs = model.binCat.varMap{end}(end);
        model.tree = smoothing(model.tree, [], model.trainParams.modelTree,
model.trainParams.smoothingK, totalAttrs);
    end
    model.tree = cleanUp(model.tree, model.trainParams.modelTree,
~keepInteriorModels, ~keepNodeInfo);
elseif model.trainParams.extractRules == 1
    % Build M5' tree and extract all its decision rules.
    % Growing the tree
    tree = splitNode(X, Y, 1:size(Y,1), 0, sd, numNotMissing, binCatNewNum,
model.trainParams, beta, numVarsTry, mOriginal, model.binCat.varMap,
extraTrees, keepNodeInfo);
    % Pruning the tree and/or filling it with models or mean values
    tree = pruneNode(tree, X, Y, model.trainParams, keepNodeInfo);
    if model.trainParams.smoothingK > 0
        totalAttrs = model.binCat.varMap{end}(end);
        tree = smoothing(tree, [], model.trainParams.modelTree,
model.trainParams.smoothingK, totalAttrs);
    end
    if model.trainParams.modelTree
        [model.rules, model.outcomesCoefs, model.outcomesAttrIdx,
model.outcomesAttrAvg, model.outcomesNumCases, outcomesCaseIdx, outcomesSD] =
...
            createRules(tree, model.trainParams.modelTree, false,
keepNodeInfo);
    else
        [model.rules, model.outcomes, ~, ~, model.outcomesNumCases,
outcomesCaseIdx, outcomesSD] = ...
```

```
            createRules(tree, model.trainParams.modelTree, false,
keepNodeInfo);
    end
    if keepNodeInfo
        model.outcomesCaseIdx = outcomesCaseIdx;
        model.outcomesSD = outcomesSD;
    end
else
    % Builds a list of decision rules using the M5'Rules method.
    model.rules = {};
    if model.trainParams.modelTree
        model.outcomesCoefs = {};
        model.outcomesAttrIdx = {};
        model.outcomesAttrAvg = {};
    else
        model.outcomes = [];
    end
    model.outcomesNumCases = [];
    if keepNodeInfo
        caseIdx = 1 : size(X,1); % so that we get original indices even after
some observations are deleted
        model.outcomesCaseIdx = {};
        model.outcomesSD = [];
    end
    currRule = 0;
    while true
        % Growing the tree
        tree = splitNode(X, Y, 1:size(Y,1), 0, sd, numNotMissing,
binCatNewNum, model.trainParams, beta, numVarsTry, mOriginal,
model.binCat.varMap, extraTrees, keepNodeInfo);
        % Pruning the tree and/or filling it with models or mean values
        tree = pruneNode(tree, X, Y, model.trainParams, keepNodeInfo);
        if model.trainParams.modelTree
            [rules, outcomesCoefs, outcomesAttrIdx, outcomesAttrAvg,
outcomesNumCases, outcomesCaseIdx, outcomesSD] = ...
                createRules(tree, model.trainParams.modelTree, true,
keepNodeInfo);
        else
            [rules, outcomes, ~, ~, outcomesNumCases, outcomesCaseIdx,
outcomesSD] = ...
                createRules(tree, model.trainParams.modelTree, true,
keepNodeInfo);
        end
        [~, idx] = max(outcomesNumCases);

        % Storing the decision rule with the biggest coverage.
        currRule = currRule + 1;
        model.rules{currRule,1} = rules{idx};
        if model.trainParams.modelTree
            model.outcomesCoefs{currRule,1} = outcomesCoefs{idx};
            model.outcomesAttrIdx{currRule,1} = outcomesAttrIdx{idx};
            model.outcomesAttrAvg{currRule,1} = outcomesAttrAvg{idx};
        else
            model.outcomes(currRule,1) = outcomes(idx);
        end
        model.outcomesNumCases(currRule,1) = outcomesNumCases(idx);
        if keepNodeInfo
```

```
            model.outcomesCaseIdx{currRule,1} =
caseIdx(outcomesCaseIdx{idx});
            caseIdx(outcomesCaseIdx{idx}) = [];
            model.outcomesSD(currRule,1) = outcomesSD(idx);
        end

        % Deleting observations covered by the stored rule.
        X(outcomesCaseIdx{idx},:) = [];
        Y(outcomesCaseIdx{idx},:) = [];
        if (size(X,1) == 0)
            break;
        end
        %sd = stdMy(Y);
        %numNotMissing = sum(~isnan(X),1);
    end
end
end

function [node, attrList] = splitNode(X, Y, caseIdx, depth, sd,
numNotMissing, binCatNewNum, trainParams, beta, numVarsTry, mOriginal,
varMap, extraTrees, keepNodeInfo)
% The function splits node into left node and right node
node.caseIdx = caseIdx;
if depth >= trainParams.maxDepth
    node.interior = false; % this node will be a leaf node
    attrList = [];
    if keepNodeInfo
        node.sd = stdMy(Y(caseIdx));
    end
    return;
end
YY = Y(caseIdx);
stdYall = stdMy(YY);
if keepNodeInfo
    node.sd = stdYall;
end
% no need to check minLeafSize*2 because minParentSize is guaranteed to be at
least twice the minLeafSize
% (size(caseIdx,2) < trainParams.minLeafSize * 2) || ...
if (size(caseIdx,2) < trainParams.minParentSize) || ...
   (stdYall < trainParams.splitThreshold * sd)
    node.interior = false; % this node will be a leaf node
    attrList = [];
    return;
end;
sdr = -Inf;
if ~extraTrees
    % This is for individual trees and trees in Bagging and Random Forests
    if isempty(numVarsTry) || (numVarsTry >= mOriginal)
        varsTry = 1:size(X, 2); % try all variables
    else
        % We will try random subset of variables (for building ensembles)
        % For categorical variables, we will try all their synthetic binary
variables
        origVList = randperm(mOriginal);
        varsTry = [varMap{origVList(1:numVarsTry)}];
```

```
    end
else
    % This is for trees in Extra-Trees
    if isempty(numVarsTry) || (numVarsTry >= mOriginal)
        % This is for the typical configuration when we try all variables,
for one split each
        varsTry = [];
        for v = 1 : mOriginal
            vars = varMap{v};
            if size(vars,2) < 2
                varsTry = [varsTry vars];
            else
                % For categorical variables, we will randomly select one
synthetic binary variable
                varsTry = [varsTry vars(randi(size(vars,2),1))];
            end
        end
    else
        % This is for the configuration when we try fewer than all variables
but they should be non constant in the node
        origVList = randperm(mOriginal);
        numVarsUsed = 0;
        varsTry = [];
        for origV = origVList
            vars = varMap{origV};
            nonConstant = false;
            for v = vars
                XX = X(caseIdx,v);
                if min(XX) ~= max(XX)
                    nonConstant = true;
                    break;
                end
            end
            if ~nonConstant
                continue;
            end
            if size(vars,2) >= 2
                % For categorical variables, we will randomly select one
synthetic binary variable
                vars = vars(randi(size(vars,2),1));
            end
            varsTry = [varsTry vars];
            numVarsUsed = numVarsUsed + 1;
            if numVarsUsed >= numVarsTry
                break;
            end
        end
    end
end
% let's find best variable and best split
for i = varsTry
    XX = X(caseIdx,i);
    % NaNs (unknown values) will not be used for split point determination
    % and there is no need to sort because unique already sorts
    nonansIdx = ~isnan(XX);
    XXnonans = XX(nonansIdx);
    if binCatNewNum(i) >= 2
```

```
        % It's simple with binary variables
        minXXnonans = min(XXnonans);
        maxXXnonans = max(XXnonans);
        if minXXnonans == maxXXnonans
            continue;
        end
        splitCandidates = (minXXnonans + maxXXnonans) / 2;
    else
        if ~extraTrees
            sorted = unique(XXnonans);
            if size(sorted,1) < 2
                continue;
            end
            splitCandidates = ((sorted(1:end-1) + sorted(2:end)) ./ 2)';
        else
            minXXnonans = min(XXnonans);
            maxXXnonans = max(XXnonans);
            if minXXnonans == maxXXnonans
                continue;
            end
            splitCandidates = minXXnonans + rand(1) * (maxXXnonans -
minXXnonans);
        end
    end
    sizeAllNoNans = size(XXnonans,1); % size without NaNs
    if (sizeAllNoNans == size(XX,1)) % if there are no NaNs
        stdY = stdYall;
    else
        stdY = stdMy(YY(nonansIdx)); % NaNs are not used for splitting
decisions
    end
    % let's find the best split
    for splitCand = splitCandidates
        leftInd = find(XX <= splitCand);
        if (size(leftInd,1) < trainParams.minLeafSize)
            continue;
        end
        rightInd = find(XX > splitCand);
        if (size(rightInd,1) < trainParams.minLeafSize)
            break; % break loop because we definitely are too near the edge
for any further split point to be allowed
        end
        % calculate SDR for the split point
        if trainParams.vanillaSDR
            sdrNew = stdY - (size(leftInd,1) * stdMy(YY(leftInd)) +
size(rightInd,1) * stdMy(YY(rightInd))) / sizeAllNoNans;
        else
            sdrNew = numNotMissing(i) / sizeAllNoNans * beta(i) * ...
                (stdY - (size(leftInd,1) * stdMy(YY(leftInd)) +
size(rightInd,1) * stdMy(YY(rightInd))) / sizeAllNoNans);
        end
        if sdrNew > sdr
            sdr = sdrNew;
            splitPoint = splitCand;
            attrList = i;
        end
    end
```

```
end
if sdr <= 0
    % This node will be a leaf node
    node.interior = false;
    attrList = [];
else
    % This node will be an interior node
    [leftInd, rightInd] = leftright(splitPoint, X(caseIdx,attrList), YY,
binCatNewNum(attrList));
    leftInd = caseIdx(leftInd);
    rightInd = caseIdx(rightInd);
    node.interior = true;
    node.splitAttr = attrList;
    node.splitLocation = splitPoint;
    [node.left, attrList2] = ...
        splitNode(X, Y, leftInd, depth + 1, sd, numNotMissing, binCatNewNum,
trainParams, ...
                beta, numVarsTry, mOriginal, varMap, extraTrees,
keepNodeInfo);
    if trainParams.modelTree
        attrList = [attrList attrList2];
    end
    [node.right, attrList2] = ...
        splitNode(X, Y, rightInd, depth + 1, sd, numNotMissing, binCatNewNum,
trainParams, ...
                beta, numVarsTry, mOriginal, varMap, extraTrees,
keepNodeInfo);
    if trainParams.modelTree
        attrList = unique([attrList attrList2]); % unique also sorts
        node.attrList = attrList;
    end
end
end

function stdev = stdMy(Y)
% Calculates standard deviation
% Does the same as Matlab's std function but without all the overhead
nn = size(Y,1);
stdev = sqrt(sum((Y - (sum(Y) / nn)) .^ 2) / nn);
end

function [leftInd, rightInd] = leftright(split, X, Y, binCatNewNum)
% Splits all observations into left and right sets. Deals with NaNs
separately.
leftInd = find(X <= split);
rightInd = find(X > split);
% Place observations with NaNs in left or right according to their Y values
isNaN = isnan(X);
if any(isNaN)
    if binCatNewNum < 2
        % For continuous variables
        [~, sorted] = sort(X(leftInd));
        sorted = leftInd(sorted);
        leftAvg = mean(Y(sorted(end - min([2 size(leftInd,1)-1]) : end)));
        [~, sorted] = sort(X(rightInd));
        sorted = rightInd(sorted);
```

```
        rightAvg = mean(Y(sorted(1 : min([3 size(rightInd,1)])))));
    else
        % For both original and synthetic binary variables
        leftAvg = mean(Y(leftInd));
        rightAvg = mean(Y(rightInd));
    end
    avgAvg = (leftAvg + rightAvg) / 2;
    smaller = Y(isNaN) <= avgAvg;
    nanInd = find(isNaN);
    if leftAvg <= rightAvg
        leftInd = [leftInd; nanInd(smaller)];
        rightInd = [rightInd; nanInd(~smaller)];
    else
        leftInd = [leftInd; nanInd(~smaller)];
        rightInd = [rightInd; nanInd(smaller)];
    end
end
end

function node = pruneNode(node, X, Y, trainParams, keepNodeInfo)
% Prunes the tree and fills it with models (or average values).
% If tree pruning is disabled, only filling with models is done.
% For each model, subset selection is done (using sequential backward
selection).
if ~node.interior
    if ~trainParams.modelTree
        node.value = mean(Y(node.caseIdx));
    else
        % Original leaf nodes ignore input variables
        node.modelCoefs = mean(Y(node.caseIdx));
        node.modelAttrIdx = [];
    end
    return;
end
node.left = pruneNode(node.left, X, Y, trainParams, keepNodeInfo);
node.right = pruneNode(node.right, X, Y, trainParams, keepNodeInfo);
if ~trainParams.modelTree
    node.value = mean(Y(node.caseIdx));
    if trainParams.prune
        errNode = calcErrNodeWithAllKnown(node, X, Y, trainParams, true); %
pretend it's known because regression tree doesn't care
    end
else
    attrInd = node.attrList;
    if isempty(attrInd) % no attributes. the model will include only
intercept
        node.modelCoefs = mean(Y(node.caseIdx));
        node.modelAttrIdx = [];
        if trainParams.prune
            errNode = calcErrNodeWithAllKnown(node, X, Y, trainParams, true);
% pretend it's known because no attributes are used
        end
    else
        XX = X;
        isNaN = isnan(X(node.caseIdx,attrInd));
        for i = 1 : length(attrInd)
            % Store average values of the variables (required when the tree
```

```
            % is used for prediction and NaN is encountered)
            % (node.modelAttrIdx provides index for the variable for which
            % modelAttrAvg is the average value)
            node.modelAttrAvg(i) =
mean(X(node.caseIdx(~isNaN(:,i)),attrInd(i)));
            % Replace NaNs by the average values of the corresponding
variables
            % of the training observations reaching the node
            XX(node.caseIdx(isNaN(:,i)),attrInd(i)) = node.modelAttrAvg(i);
        end
        A = [ones(length(node.caseIdx),1) XX(node.caseIdx,attrInd)];
        node.modelCoefs = A \ Y(node.caseIdx);
        node.modelAttrIdx = attrInd;
        if trainParams.prune
            errNode = calcErrNodeWithAllKnown(node, XX, Y, trainParams,
true);
            if trainParams.eliminateTerms
                % Perform variable selection
                attrIndBest = attrInd;
                coefsBest = node.modelCoefs;
                changed = false;
                for j = 1 : length(attrInd)
                    attrIndOld = node.modelAttrIdx;
                    for i = 1 : length(attrIndOld)
                        node.modelAttrIdx = attrIndOld;
                        node.modelAttrIdx(i) = [];
                        A = [ones(length(node.caseIdx),1)
XX(node.caseIdx,node.modelAttrIdx)];
                        node.modelCoefs = A \ Y(node.caseIdx);
                        errTry = calcErrNodeWithAllKnown(node, XX, Y,
trainParams, true);
                        if errTry < errNode
                            errNode = errTry;
                            attrIndBest = node.modelAttrIdx;
                            coefsBest = node.modelCoefs;
                            changed = true;
                        end
                    end
                    node.modelAttrIdx = attrIndBest;
                    node.modelCoefs = coefsBest;
                    if ~changed
                        break;
                    end
                end
                % Update node.modelAttrAvg if the used subset of variables
has changed
                if length(node.modelAttrIdx) < length(attrInd)
                    for i = 1 : length(node.modelAttrIdx)
                        node.modelAttrAvg(i) = node.modelAttrAvg(attrInd ==
node.modelAttrIdx(i));
                    end
                    node.modelAttrAvg =
node.modelAttrAvg(1:length(node.modelAttrIdx));
                end
            end
        end
        if keepNodeInfo
```

```
            val = [ones(length(node.caseIdx),1)
X(node.caseIdx,node.modelAttrIdx)] * node.modelCoefs;
            node.sd = sqrt(mean((val - Y(node.caseIdx)).^2));
        end
    end
end
if trainParams.prune && ...
   ( ...
    ((~trainParams.aggressivePruning) && (calcErrSubtree(node, X, Y,
trainParams) >= errNode)) || ...
    (trainParams.aggressivePruning && (calcErrSubtreeAggressive(node, X, Y,
trainParams) >= errNode)) ...
   )
    % above we could also add "(sd * 1E-6 > errNode)"
    % this node will be a leaf node
    node.interior = false;
    if trainParams.modelTree
        node = rmfield(node, {'splitAttr', 'splitLocation', 'left', 'right',
'attrList'});
    else
        node = rmfield(node, {'splitAttr', 'splitLocation', 'left',
'right'});
    end
else
    if trainParams.modelTree
        node = rmfield(node, 'attrList');
    end
    % Store average value of the split variable (required when the tree
    % is used for prediction and NaN is encountered)
    notNaN = node.caseIdx(~isnan(X(node.caseIdx,node.splitAttr)));
    %node.splitAttrAvg = mean(X(notNaN,node.splitAttr)); % not really needed.
we can just set nanLeft
    node.nanLeft = mean(X(notNaN,node.splitAttr)) <= node.splitLocation;
end
end

function err = calcErrSubtree(node, X, Y, trainParams)
% Calculates error of the subtree
if node.interior
    err = (length(node.left.caseIdx) * calcErrSubtree(node.left, X, Y,
trainParams) + ...
           length(node.right.caseIdx) * calcErrSubtree(node.right, X, Y,
trainParams)) / ...
           length(node.caseIdx);
else
    err = calcErrNode(node, X, Y, trainParams);
end
end

function err = calcErrSubtreeAggressive(node, X, Y, trainParams)
% Calculates error of the subtree, applies penalty
[err, v] = calcErrSubtreeAggressiveDo(node, X, Y, trainParams);
nn = length(node.caseIdx);
if (nn > v)
    err = err * (nn + v * 2) / (nn - v);
else
```

```
    err = err * 10;
end
end
function [err, v] = calcErrSubtreeAggressiveDo(node, X, Y, trainParams)
% Calculates error of the subtree
if node.interior
    [errLeft, vLeft] = calcErrSubtreeAggressiveDo(node.left, X, Y,
trainParams);
    [errRight, vRight] = calcErrSubtreeAggressiveDo(node.right, X, Y,
trainParams);
    err = (length(node.left.caseIdx) * errLeft + length(node.right.caseIdx) *
errRight) / length(node.caseIdx);
    v = vLeft + vRight + 1;
else
    err = calcErrNode(node, X, Y, trainParams);
    if trainParams.modelTree
        v = length(node.modelCoefs);
    else
        v = 1;
    end
end
end

function err = calcErrNode(node, X, Y, trainParams)
% Calculates error of the node. Handles missing values.
if trainParams.modelTree
    % Replace NaNs with the average values of the corresponding variables
    % of the training observations reaching the node
    isNaN = isnan(X(node.caseIdx,node.modelAttrIdx));
    for i = 1 : length(node.modelAttrIdx)
        X(node.caseIdx(isNaN(:,i)),node.modelAttrIdx(i)) =
node.modelAttrAvg(i);
    end
end
err = calcErrNodeWithAllKnown(node, X, Y, trainParams, false);
end

function err = calcErrNodeWithAllKnown(node, X, Y, trainParams,
forDroppingTerms)
% Calculates error of the node. Assumes all values are known.
if trainParams.modelTree
    val = [ones(length(node.caseIdx),1) X(node.caseIdx,node.modelAttrIdx)] *
node.modelCoefs;
    deviation = mean(abs(val - Y(node.caseIdx)));
    v = length(node.modelCoefs);
else
    deviation = mean(abs(node.value - Y(node.caseIdx)));
    v = 1;
end
if ~trainParams.aggressivePruning
    nn = length(node.caseIdx);
    err = (nn + v) / (nn - v) * deviation;
else
    if forDroppingTerms
        nn = length(node.caseIdx);
        err = (nn + v * 2) / (nn - v) * deviation;
```

```
    else
        err = deviation;
    end
end
end

function node = cleanUp(node, modelTree, removeInteriorModels, removeCaseIdx)
% Removing the temporary fields
node.numCases = length(node.caseIdx);
if removeCaseIdx
    node = rmfield(node, 'caseIdx');
end
if node.interior
    if removeInteriorModels
        if modelTree
            node = rmfield(node, {'modelCoefs', 'modelAttrAvg',
'modelAttrIdx'});
        else
            node = rmfield(node, 'value');
        end
    end
    node.left = cleanUp(node.left, modelTree, removeInteriorModels,
removeCaseIdx);
    node.right = cleanUp(node.right, modelTree, removeInteriorModels,
removeCaseIdx);
end
end

function node = smoothing(node, list, modelTree, smoothingK, totalAttrs)
% Performs smoothing by incorporating interior models into leaf models.
% Deals with modelAttrAvg, so that unknown values can be substituted with
% modelAttrAvg at leaves.
if node.interior
    if modelTree
        data.attrIdx = node.modelAttrIdx;
        data.coefs = node.modelCoefs;
        data.attrAvg = zeros(totalAttrs,1);
        data.attrAvg(node.modelAttrIdx) = node.modelAttrAvg;
    else
        data.value = node.value;
    end
    data.numCases = length(node.caseIdx);
    list{end+1} = data; % making a list. will be used at leaf nodes
    node.left = smoothing(node.left, list, modelTree, smoothingK,
totalAttrs);
    node.right = smoothing(node.right, list, modelTree, smoothingK,
totalAttrs);
else
    if modelTree
        len = length(list);
        if len > 0
            attrIdx = node.modelAttrIdx;
            s_n = length(node.caseIdx);
            coefs = zeros(totalAttrs+1,1);
            coefs([1 attrIdx+1]) = node.modelCoefs;
            attrAvg = zeros(totalAttrs,1);
```

```
            if ~isempty(attrIdx)
                attrAvg(attrIdx) = node.modelAttrAvg;
            end
            % pretend to go from the leaf node to the root node
            for i = len:-1:1
                % Update list of used variables
                attrIdx = union(attrIdx, list{i}.attrIdx); % union sorts.
this also will make equations easier to understand
                % Coefs at this node
                coefsHere = zeros(size(coefs));
                coefsHere([1 list{i}.attrIdx+1]) = list{i}.coefs;
                % Recalculate weighted averages for NaNs
                idx = true(size(coefs));
                idx(1) = false;
                idx((coefs == 0) & (coefsHere == 0)) = false;
                if any(idx)
                    idxAttr = idx(2:end);
                    attrAvg(idxAttr) = ...
                        attrAvg(idxAttr) .* s_n .* coefs(idx) ./ (s_n .*
coefs(idx) + smoothingK .* coefsHere(idx)) + ...
                        list{i}.attrAvg(idxAttr) .* smoothingK .*
coefsHere(idx) ./ (s_n .* coefs(idx) + smoothingK .* coefsHere(idx));
                end
                % Recalculate smoothed coefs
                coefs = (s_n * coefs + smoothingK * coefsHere) / (s_n +
smoothingK);
                s_n = list{i}.numCases; % s_n for next iteration
            end
            attrIdx = attrIdx(:)'; % force row vector
            node.modelCoefs = coefs([1 attrIdx+1]);
            node.modelAttrIdx = attrIdx;
            node.modelAttrAvg = attrAvg(attrIdx)';
        end
    else
        len = length(list);
        if len > 0
            value = node.value;
            s_n = length(node.caseIdx);
            % pretend to go from the leaf node to the root node
            for i = len:-1:1
                value = (s_n * value + smoothingK * list{i}.value) / (s_n +
smoothingK); % calculate smoothed values
                s_n = list{i}.numCases; % s_n for next iteration
            end
            node.value = value;
        end
    end
end
end

function [rules, outcomes, outcomesAttrIdx, outcomesAttrAvg,
outcomesNumCases, outcomesCaseIdx, outcomesSD] = ...
        createRules(tree, modelTree, maxCoverageOnly, keepNodeInfo)
% Extracts decision rules from a tree.
totalRules = countRules(tree);
rules = cell(totalRules,1);
if modelTree
```

```
    outcomes = cell(totalRules,1);
    outcomesAttrIdx = cell(totalRules,1);
    outcomesAttrAvg = cell(totalRules,1);
else
    outcomes = zeros(totalRules,1);
    outcomesAttrIdx = [];
    outcomesAttrAvg = [];
end
outcomesNumCases = zeros(totalRules,1);
outcomesCaseIdx = cell(totalRules,1);
if keepNodeInfo
    outcomesSD = nan(totalRules,1);
else
    outcomesSD = [];
end
currRule = 0;
maxNumCases = 0;
createRulesDo(tree, {});
function createRulesDo(node, tests)
    if node.interior
        % Interior nodes become tests in rules
        tests{end+1,1}.attr = node.splitAttr;
        tests{end}.location = node.splitLocation;
        tests{end}.le = true; % "<="
        tests{end}.orNan = node.nanLeft; % whether to accept NaN
        createRulesDo(node.left, tests);
        tests{end}.le = false; % ">"
        tests{end}.orNan = ~node.nanLeft; % whether to accept NaN
        createRulesDo(node.right, tests);
        return;
    end
    % Leaf nodes become outcomes for the rules
    currRule = currRule + 1;
    if maxCoverageOnly && (length(node.caseIdx) <= maxNumCases)
        % If we'll actually want only the rule with the maximum coverage
        % then we don't need to store everything for rules that are already
        % known to be smaller.
        outcomesNumCases(currRule,1) = 0;
        return;
    end
    rules{currRule,1} = tests; % store all tests
    if modelTree
        outcomes{currRule,1} = node.modelCoefs;
        outcomesAttrIdx{currRule,1} = node.modelAttrIdx;
        if isempty(node.modelAttrIdx)
            outcomesAttrAvg{currRule,1} = [];
        else
            outcomesAttrAvg{currRule,1} = node.modelAttrAvg;
        end
    else
        outcomes(currRule,1) = node.value;
    end
    maxNumCases = length(node.caseIdx);
    outcomesNumCases(currRule,1) = maxNumCases;
    outcomesCaseIdx{currRule,1} = node.caseIdx;
    if keepNodeInfo
        outcomesSD(currRule,1) = node.sd;
```

```
    end
end
end

function nRules = countRules(node)
% Counts all rules (equal to the number of leaf nodes) in the tree.
if node.interior
    nRules = countRules(node.left) + countRules(node.right);
else
    nRules = 1;
end
end
```

APPENDIX (II): SAMPLE DATA POINTS

This section provides a set of sample data points and their descriptions. The full dataset is available for any further research and analysis at https://figshare.com/s/8de57c3a0ca8f8ed37c4.

	Sample Data Points										
Lables	**#1**	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**	**#9**	**#10**	**#11**
X1	42864	32080	31529	32053	31660	29952	27949	4852	29096.5	30380	16484
X2	574.7	424.2	435.4	441.7	438	429.7	390.2	43.7	0	0	296.1
X3	0	0	0	0	0	0	0	0	0	0	0
X4	0	0	0	0	0	0	0	0	8.8	8.8	0
X5	0	0	0	0	0	0	0	0	5.9	7.3	0
X6	0	0	0	0	0	0	0	0	0	0	0
X7	204.5	107.8	133.6	146.4	146.3	155.2	103.3	0	0	0	108.7
X8	0	0	0	0	0	0	0	0	4.9	4.9	0
X9	0	0	0	0	0	0	0	0	0	0	1.1
X10	0	0	0	0	0	0	0	0	0	0	0
X11	134.5	102.3	97.9	99.2	99.5	92.3	90.5	20.7	262.2	279.9	13.8
X12	0	0	0	0	0	0	0	0	0	0	0
X13	0	0	0	0	0	0	0	0	0	0	0
X14	659	295	356	360	358	362	305	0	106	114	248
X15	0	0	0	0	0	0	0	0	0	0	0
X16	0	0	0	0	0	0	0	0	0	0	0
X17	0	0	0	0	0	0	0	0	0	0	0
X18	0	0	0	0	0	0	0	0	0	0	0
X19	48.3	30.4	33.5	32.9	32.3	29.3	27.2	1.4	0	0	32.5
X20	0	0	0	0	0	0	0	0	0	0	0
X21	0	0	0	0	0	0	0	0	0	0	0
X22	10.4	7.1	6.8	8.4	7.1	4.7	6.2	3.2	0	0	0
X23	0	0	0	0	0	0	0	0	3.3	3.3	0
X24	254.1	207.2	184.4	183.4	182.4	163.6	161.6	43.9	35.3	34.8	32.9
X25	0	0	0	0	0	0	0	0	0	0	0
X26	0	0	0	0	0	0	0	0	0	0	0
X27	0	0	0	0	0	0	0	0	0	0	0
X28	0	0	0	0	0	0	0	0	0	0	0

X29	0	0	0	0	0	0	0	0	0	0	0
X30	0	0	0	0	0	0	0	0	0	0	0
X31	0	0	0	0	0	0	0	0	0	0	0
X32	0	0	0	0	0	0	0	0	0	0	0
X33	0	0	0	0	0	0	0	0	0	0	0
X34	0	0	0	0	0	0	0	0	0	0	0
X35	0	0	0	0	0	0	0	0	0	0	0
X36	0	0	0	0	0	0	0	0	0	0	0
X37	0	0	0	0	0	0	0	0	0	0	0
X38	0	0	0	0	0	0	0	0	0	0	0
X39	0	0	0	0	0	0	0	0	0	0	0
X40	0	0	0	0	0	0	0	0	0	0	0
X41	0	0	0	0	0	0	0	0	0	0	0
X42	0	0	0	0	0	0	0	0	0	0	0
Y	1095	940.2	723.3	805.9	911.6	844.5	673.8	64.9	1145	1493.5	478.9

Description of Design Features Extracted from The BIM Databases

Labels	BIM Extracted Design features	Unit	Category
X1	Division Weight	Weight (kg)	Material-Weight
X2	Angles	Length (m)	Material-Steel Sections
X3	Channels	Length (m)	Material-Steel Sections
X4	I Beams	Length (m)	Material-Steel Sections
X5	Miscellaneous beams	Length (m)	Material-Steel Sections
X6	Miscellaneous channels	Length (m)	Material-Steel Sections
X7	Structural Tees from W Shapes	Length (m)	Material-Steel Sections
X8	Tarpon Z Sections	Length (m)	Material-Steel Sections
X9	Wide flange	Length (m)	Material-Steel Sections
X10	Crane rails	Length (m)	Material-Steel Sections
X11	Bent plate	Area (m^2)	Material-Plate
X12	Checker plate	Area (m^2)	Material-Plate
X13	Grating	Area (m^2)	Material-Plate
X14	Plate	Area (m^2)	Material-Plate
X15	Extra Extra Strong Pipe	Length (m)	Material-Pipes
X16	Extra Strong Pipe	Length (m)	Material-Pipes
X17	Standard Pipe	Length (m)	Material-Pipes
X18	Hollow steel sections	Length (m)	Material-Hollow Sections
X19	Round hollow steel sections	Length (m)	Material-Hollow Sections
X20	Cold formed channels	Length (m)	Material-Cold-formed
X21	Tarpon Cold Formed Channels	Length (m)	Material-Cold-formed
X22	Flat bar	Length (m)	Material-Bars
X23	Rebar	Length (m)	Material-Bars
X24	Round bar	Length (m)	Material-Bars
X25	Square bar	Length (m)	Material-Bars
X26	Hex Bar	Length (m)	Material-Bars
X27	Expansion Anchor Bolts	Quantity	Material-Anchors
X28	Heavy Duty Expansion Anchor Bolts	Quantity	Material-Anchors
X29	Threaded Anchor Rods	Quantity	Material-Anchors
X30	Adhesive Anchor Cartridges	Quantity	Material-Anchors
X31	Complete penetration weld	Length (m)	Connection-Welding
X32	Partial Penetration Weld	Length (m)	Connection-Welding
X33	Bevelled Washers	Quantity	Connection-Bolted
X34	Button Head Machine Bolt	Quantity	Connection-Bolted
X35	Compressible Washers with DTI	Quantity	Connection-Bolted
X36	Flat Washers	Quantity	Connection-Bolted
X37	Hex Head Machine Bolt	Quantity	Connection-Bolted
X38	Hex Nuts	Quantity	Connection-Bolted
X39	Hex Type Bolts	Quantity	Connection-Bolted
X40	M Type Bolts	Quantity	Connection-Bolted
X41	Mechanical Pipes	Length (m)	Material-Pipe
X42	Nelson Studs	Quantity	Connection-Stud
Y	Labour Hours	Hours	Hours per division