# On Ensemble Models for Associative Classification

by

Md Rayhan Kabir

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

Master of Science

Department of Computing Science

University of Alberta

# Abstract

Associative classifiers have shown competitive performance with state-of-the-art methods for predicting class labels. In addition to accuracy performance, associative classifiers produce human readable rules for classification which provides an easier way to understand the decision process of the model. Early models of associative classifiers suffered from the limitation of selecting proper threshold values which are dataset specific. Recent work on associative classifiers eliminates that restriction by searching for statistically significant rules. However, a high dimensional feature vector in the training data impacts the performance of the model. In this study we propose Dynamic Ensemble Associative Learning (DEAL) where we use associative classifiers as base learners on feature sub-spaces and a dynamic feature sampling procedure which automatically defines the number of base learners and ensures diversity and completeness among the subset of feature vectors. This method eliminates the limitation of high memory requirement and runtime of recent associative classifiers for training datasets having large feature vectors without jeopardising the accuracy of the model.

In addition, to better understand the decision process of our model, we propose an ensemble model, Classification by Frequent Association Rules (CFAR) using associative classifiers as base learners. In our approach, instead of using classical ensemble and a voting method, we rank the generated rules based on frequency and select a subset of the rules for predicting class labels. This ensemble approach CFAR also eliminates the limitation of high memory re-

quirement and runtime of recent associative classifiers. This approach removes the noisy rules for the classification process which further enhances the performance of the model in terms of accuracy.

Further, inspired by the tremendous performance of deep neural networks, we propose Deep Associative Classifier (DAC), an ensemble of associative classifiers that transforms features in a deep model representation. This model has deep neural network like architecture with associative classifiers as a base learner and overcomes some of the limitations of deep neural network architecture as well as associative classifiers.

We use 10 datasets from the UCI repository to evaluate the performance of the models. We compare our approaches with different machine learning models. All three of our proposed models address the limitation of recent associative classifier of requiring high memory and long runtime along with showing competitive performance in accuracy in contrast to various state-of-the-art classifiers.

# Preface

The following papers are a part of this manuscript. Paper 1 and Paper 2 are accepted in International Symposium on Foundations and Applications of Big Data Analytics, FAB 2022 and IEEE International Conference on Knowledge Graph (ICKG-2022) respectively. Paper 3 is accepted at The 38th ACM/SIGAPP Symposium On Applied Computing.

1. Md Rayhan Kabir, Osmar R Zaiane, "Dynamic Ensemble Associative Learning", accepted for publication on International Symposium on Foundations and Applications of Big Data Analytics, FAB 2022, Istanbul, Turkey, November 10-13, 2022.

2. Md Rayhan Kabir, Samridhi Vaid, Nitakshi Sood, Osmar R Zaiane, "Deep Associative Classifier", accepted for the publication on IEEE International Conference on Knowledge Graph (ICKG-2022), Orlando, Florida, November 30-December 1.

3. Md Rayhan Kabir and Osmar R Zaiane. 2023. "Classification by Frequent Association Rules", accepted in The 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23), March 27 - March 31, 2023, Tallinn, Estonia.

Among the mentioned papers, Paper 1 is included in Chapter 3, Paper 2 is included in Chapter 5 and Paper 3 is included in Chapter 4. Paper 1 received the best paper award at International Symposium on Foundations and Applications of Big Data Analytics, FAB 2022. All three papers are my original work and I am the primary contributor. In Paper 2, Nitakshi Sood

provided general idea and Samridhi Vaid helped to find the preliminary results and contributed in writings.

Dr. Osmar R Zaiane supervised all the papers by providing his thoughtful insight and directions.

*To Mom and Dad*

*For inspiring and supporting me throughout my life.*

*And miles to go before I sleep,*
*And miles to go before I sleep.*

– Robert Frost (1923).

# Acknowledgements

I want to express my heartfelt gratitude to Professor Osmar R Zaiane for his valuable guidance, thoughtful insight, support and encouragement. Without his guidance I could not complete this work. His priceless insights and suggestions will always be guidance for my future career and life.

I am grateful to all the members of Amii xAI lab for the insightful discussion. I would like to thank Nitakshi Sood for helping me with the implementation of SigD2. I would also like to thank Samridhi Vaid for collaborating with us in the project "Deep Associative Classifier".

Lastly, I want to express my deepest gratitude to my family who are there in every aspect of my life. My mom and dad who supported me and encouraged me in all my life. My brother Rakib who is always a constant support for me, especially during my difficult times. Without them, I could not come this far in my life.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

With the steady increase in use of machine learning in real-life scenarios, the importance of classification is de facto increasing. Classification is a supervised machine technique learning that labels the data in different distinct class labels. In the present context, this process is being used in various tasks like classification of image, text, tabular data, etc. To solve classification problems, some of the popular algorithms are Logistic Regression, Decision Trees [32], Random Forest [10], Support Vector Machines [15] and Artificial Neural Networks [20]. Even though these models have good performance in classification tasks, most of the classification models work in a black box fashion. After training the model, the model remains opaque without revealing what has actually been learned [4]. For instance, even though amazingly accurate, neural networks, after convergence of the edge weights and node biases, are not interpretable. In addition, during inference, they do not provide necessary explanation for the prediction, which is troublesome in some application fields like medical diagnosis, financial decision making, etc. Since machine learning is becoming popular and ubiquitous, explanation is now even legislated in many jurisdictions. This has lead to many effort attempting to explain salient features in deep learning, but also a return to rule-based classifiers. Associative classifiers are among them.

Associative classifiers use an association rule mining approach [1] to discover frequent patterns, and lately statistically significant patterns, from which

to derive classification rules. The test data is then classified using the rules that are generated during the training phase. A rule is in the form of $X \rightarrow Y$ where $X$, the antecedent, is a conjunction of co-occurring features, and $Y$, the consequent, is a class label. Several associative classifiers have been proposed namely CBA [29], ARC [2], CMAR [28], CPAR [38]. Rules generated by the associative classifiers are simple and human readable thus their predictions can be interpreted. Though the associative classifiers are competitive with the state of the art classifiers and have a huge advantage of built-in explainability, they suffer from the limitation imposed by the required threshold tuning, namely support and confidence. To choose optimal support and confidence values for a dataset requires prior knowledge about the dataset and it differs from dataset to dataset. Another disadvantage of the mentioned algorithms is that sometimes they produce noisy rules during the rule generation phase. Some of the noisy rules prevail even after the rule pruning. If any rule does not add any valid information towards the class label then the rule is considered as a noisy rule. Noisy rules affect the performance of the model as they hinder the classification process to classify the instance to its appropriate class label.

To solve the issue of requiring prior confidence and support values, Li and Zaiane [27] proposed SigDirect where they used the Kingfisher algorithm [21] to find statistically significant classification rules. This model removes the necessity of requiring threshold values which are different for different datasets. Sood and Zaiane [34] showed though SigDirect removes the necessity of hyper parameters, it still produces many noisy rules. Thus they proposed SigD2, a classifier based on SigDirect with a two stage rule pruning strategy which further removes the noisy rules from SigDirect. The improvement is promising as it reduces the number of rules without compromising the accuracy. In fact it was shown that SigD2 outperforms other rule-based classifiers as well as classical classifiers such as SVM, Bayesian, C4.5 and even simple neural networks [34]. SigD2 increases the efficiency of associative classifiers as it uses a minimum number of rules for the classification process. As we discussed earlier, one major advantage of an associative classifier is the fact that the rules produced for the classification process are human readable. However a

large number of rules and noisy rules make it hard to understand the decision process. Using less rules, SigD2 makes the understanding of the decision process of the classifier easier.

Though the performance of SigD2 is very competitive, its performance is limited in terms of required memory and run-time if the feature vector of the feature space is large. This limitation persists in Sigdirect too making both the models non-scalable when the feature vector of the dataset is large. An Ensemble of classifiers, each trained on a subset of the feature space, could address this issue. However, one has to fix the size of the ensemble. To make SigDirect and SigD2 scalable, we propose an ensemble model named Dynamic Ensemble Associative Learner(DEAL) to improve the efficiency of SigD2 in terms of memory requirement and runtime without hampering the accuracy. As a base learner, we considered SigD2 as SigD2 outperforms other associative classifiers [34]. In our model, we add a dynamic feature sampling technique inspired from the work of Cao et al. [12] which eliminates the necessity of defining the number of base learners for the ensemble model and ensures diversity and coverage of the feature space among the base learners. We first sample the features randomly to form a subset of the feature vector. After sampling, we calculate the overlap of the new subset with the previously generated subsets. If the overlap of the new subset is high, we discard the subset and start another sampling. We stop the sampling procedure either if after subsequent sampling, we do not find any more subset which has low overlap with the previous subsets or all the features are covered by the generated subsamples. We train one SigD2 model with each of the subsamples and perform max-vote among the class label predictions. The class label with the highest vote is assigned to the new test instance. Our proposed method reduces the memory requirements and makes the model faster in terms of runtime without affecting the accuracy. On top of that, even after the ensemble, the model remains interpretable as we can gather the rules from the base learners

Classical ensemble approaches make the final class prediction by training the base learners and taking their vote. In DEAL, it is quite difficult to

interpret the decision process as at first the base models who voted for the final class are to be selected and the decision process of each of the base models is to be determined which can be a tedious process. In the case of random forest where the base learners are decision trees, the decision process of each of the trees should be considered individually to understand the decision process. We focus on this part of the ensemble to keep the ensemble model simple so that the decision process can be inferred easily. Though DEAL solves the limitation of SigD2 in the case of dataset with high dimensional feature vector space, to understand the decision process of DEAL we have to go through the rules produced by each of the base learners. Thus we propose Classification by Frequent Associative Rules(CFAR). In this approach, we train the base learners with a subset of the feature vectors. After training, we collect all the rules generated by the base learners. We speculate some base learners can produce noisy rules which do not add any information to the decision process rather affect the performance of the model. Thus, we rank the rules based on their frequency in the base learners. Our assumption is if a rule is produced by many base learners, that rule has more importance in the classification process. We gradually select rules from the pools of rules based on their frequency. We deploy only the selected rules for the classification of test dataset. This approach makes the model simpler as we get a very small number of rules after the selection based on their ranking. In this process, we also have the ranking of the rules based on their frequency, thus we can understand the importance of a rule and features in the decision process. Following this strategy, the decision process becomes easy to interpret. CFAR also shows competitive performance in terms of accuracy compared to SigD2. CFAR requires less memory and is faster than base classifiers. One advantage of CFAR is that the final rules which are selected for classification process are even less than those generated by SigD2. Thus the interpretation of the decision process of CFAR is more suitable than SigD2.

Further, we consider deep learning models for classification as they have impressive result in classification task. Deep learning models show superior performance in comparison with other machine learning models in different

application fields like natural language processing, image processing, prediction, classification, etc. One of the main reasons for the high performance of deep learning techniques is their layered architecture. It has been conjectured that the success of deep learning is due to the layer-by-layer processing, the in-model feature transformation, and the complexity of the model [40]. This powerful layered architecture of deep learning models enhances the performance of the model. In spite of being very efficient deep learning model has several drawbacks including the difficulties to understand the decision process which we already discussed. Another drawback of deep learning models is the necessity of hyper-parameter tuning. The number of hyper parameters in deep learning models is quiet large. To get an efficient model, these hyper-parameters require proper tuning and it is hard to know before hand which set of parameter will work best. In addition to this, while tuning the parameters, it is hard to interpret the effect of changing the parameters. Deep learning models also require a large amount of labelled training data. Training a deep learning model with less data does not often provide good performance because of requiring large amount of data for the convergence of deep learning models. But in many domains, large amount of data is not available. In those domains, deep learning models become less efficient.

Being inspired from the deep leaning models, Zhou et al. [40] tried to utilize the deep representation of deep neural networks while eliminating some of the limitations of the deep neural networks by developing a an ensemble model called the gcForest. We speculate using the idea adapted by gcForest and further exploit the layered architecture of deep neural networks. In the case of understanding the decision process of the classification model, associative classifiers have a huge advantage over the other models. This motivated us to develop a new classifier that utilizes the advantage of the layer-wise data processing architecture of deep neural networks and the interpretability of the associative classifier that will require even fewer hyper-parameters than gcForest, is easy to tune, requires less memory, has the potential to be explainable. Our endeavour in using and combining the deep representation of deep neural networks and the simplicity of the associative classifier results in an

ensemble model, Deep Associative Classifier (DAC) that uses associative classifiers as base learners of the ensemble and has a deep architecture. With this approach, we seek to improve classification accuracy by utilizing the layered architecture of the deep neural networks and associative classifiers. Further, we try to improve the performance of the associative classifier in terms of memory requirement for datasets with high dimensions and reduce the number of hyper-parameter to make the model more practical to tune. Deep Associative Classifier has much fewer hyper-parameters and lower memory requirements than gcForest. Thus the effect of tuning the hyper parameter can be easily understandable from the result of the model. There was another limitation of gcForest which is fixing the number of base learners in each layer and pass the generated output to the next layer without further evaluation. In the case of ensemble models, diversity is one of the major factor but when a large number of base learners are trained on subsets of a small dataset, most of the base models produce a similar result which affects the advantage of the ensemble process. Thus in our proposed model, after training the base learner at each layer, we analyze the generated output of the base learner and filter the output which are exactly the same. This process ensures diversity among the base learners which further enhances the performance of the model. Our proposed model DAC outperforms gcForest in terms of accuracy on the majority of the datasets we tried. While competing with the gcForest and deep neural models in terms of accuracy, our architecture does not require high computing resources i.e. GPU like the deep neural models. Moreover, in comparison with the associative classifier SigD2 which has been used as a base learner of the model, DAC reduces the memory requirement and runtime significantly tough unfortunately explainability is hindered in this process.

## 1.2 Thesis Statement

In this study we intend to solve the drawbacks of the recent associative classifiers. For this we hypothesize the following statements:

1. Associative classifiers have huge advantage over the other classifiers as

the rules generated for the classification purpose are human readable. Though recent associative classifiers solve the drawback of requiring dataset specific threshold values, the model is inefficient in the case of dataset having large feature spaces, we speculate an ensemble model where the base models are trained on a subset of the feature spaces can be an efficient solution to address the drawback of the recent associative classifiers SigDirect and SigD2.

2. Typical ensemble model needs to specify the size of the ensemble at the beginning of the model construction. We speculate, while we need a large number of base learners in the case of a dataset having large feature spaces, a small number of base learners which covers the whole feature spaces with diversity among the subsamples can reduce the number of base learners. Thus we incorporate the idea of dynamically fixing the size of the ensemble of each of the dataset.

3. Typical ensemble model using max-vote strategy can lead to bias and understanding the decision process can be hard. We hypothesize that gathering the rules from the base learners and filtering them can provide better result and better understanding of the decision process of the ensemble.

4. We also hypothesize that one of the important reason of excellent performance of deep learning model in classification is layer by layer data processing. We speculate that we can incorporate the idea of layer by layer data processing using an ensemble of associative classifiers where the training data will be processed in layer by layer fashion. We also speculate, tuning deep learning models is quite inefficient as the number of parameter is quite large. Using the ensemble of associative classifiers, we intend to reduce the number of hyper-parameter so that the model becomes easy to tune.

## 1.3 Thesis Contribution

The contribution of this study is as follows:

1. We Propose Dynamic Ensemble Associative learner (DEAL) which is an ensemble of associative classifier SigD2. DEAL removes the high memory requirements of runtime of SigD2 in the case of dataset having large feature vector space. Side by side, DEAL eliminates the necessity of prefixing the ensemble size as this method dynamically fixes the number of subsamples by ensuring diversity among them. We evaluate our model with the other common classifiers which shows DEAL has better performance than the other classifier in terms of accuracy. DEAL also significantly reduces the memory requirement and makes the model faster in comparison to SigDirect and SigD2.

2. We propose Classification by Frequent Association Rules (CFAR) where instead of the classical max-vote ensemble model, we gathered the rules produced by the base learners. We rank those rules based on their frequency in the base learners. Then the rules are selected based on a threshold which is determined automatically by the model for predicting the class label of test data. This model is also evaluated based on UCI dataset and shows competitive performance in terms of accuracy compared to associative classifiers and DEAL . Another advantage of CFAR is despite of being an ensemble, CFAR selects a very small number of rules for the classification process thus the decision process of CFAR is easily understandable.

3. We propose Deep Associative Classifier (DAC) where we implement ensemble of SigD2 to process the training data in layer wise architecture. This model mimics the model of deep neural network. In each layer, base classifiers produce class probabilities of each instance along with predicting class labels. These class probabilities are added to the original features in the next layer. We incorporate a filter layer in between two layers of the the ensemble which ensures the diversity among the

base learners. We also evaluate this model based on UCI datasets. The evaluation shows the competitive performance of the proposed model compared to associative classifiers and simple deep neural architectures.

## 1.4 Thesis Outline

The rest of the thesis is organized as follows:

1. Chapter 2 reviews the related works from the literature in the area of associative classifiers, deep learning models, ensemble models and their various applications. Section 2.1 discusses different rule based classifiers, associative classifiers, their working procedure and drawbacks. Section 2.2 describes the application of deep learning models and few proposed models which mimic the deep learning model to keep the model simple and easily tunable. Subsection 2.3 explains different ensemble models, their advantages and applications.

2. Chapter 3 provides detailed explanation for the experiment of Dynamic Ensemble Associative Learner. Section 3.1 explains the proposed architecture of the model. In section 3.2 we provide and explain the result of proposed model on UCI datasets. We also make a comparative analysis on the performance of DEAL with the other classifiers.

3. In chapter 4, we provide our experiment of Classification by Frequent Association Rules(CFAR). In Section 4.1, we explain the architecture and methodology of of the proposed model. In Section 4.2 we provide the result of our experiment. Here we provide explanation on the performance of the model with comparison to other classification models. We also provide the interpretability of the model in this section.

4. Chapter 5 contains the details of our proposed method Deep Associative Classifier. Section 5.1 explains the algorithm and architecture of Deep Associative Classifier(DAC). We provide detailed explanation on the layered architecture of DAC and the data processing procedure. Section 5.2

contains the experimental results and explanation of the result. We also made comparative analysis on the performance of DAC on Section 5.2.

5. Chapter 6 concludes this thesis with the summary of the proposed methods and results. This section also includes the future prospect of the experiments those are performed in this thesis.

# Chapter 2

# Related Work

In this chapter we discuss on related work on classification, associative classification and ensemble learning models for classification. We also discuss the associative classification process from the literature.

## 2.1 Classification

Classification is a process of labeling a given instance to a class label. In this process a model is trained on labelled training data. In the training phase the model finds patterns among the data and establish relationship between the pattern and the class label. When a new data instance is provided for labelling, the model use pre-discovered patterns to label the new data instance. Till now several state-of-the-art machine learning models has been designed for classification task. Among them support vector machine(SVM), decision tree, naive bayes classifier, logistic regressions are noteworthy. In this section we will discuss them briefly.

Cortes et al. [15] proposed support vector machine. This model is widely used in classification task. Support vector machine finds a separating hyperplane between the data points of two different class. After plotting the input vector in a data plane, support vector machine tries to find a separating plane. In this process kernel plays an important role. Sometimes data points that are not separable in one kernel might be separable in a high dimensional kernel. This model is efficient when the datapoints are linearly separable in a high dimensional plane. But if the datasets are very large. In that case SVM takes

huge time to find a hyperplane between datapoints of two classes.

Decision tress proposed by Quinlan [32] is another widely used classification model. One important feature of decision tree is the decision process is interpretable. In decision tree, data instances are splitted based on different features of the dataset. This forms a tree like structure. The intermediate nodes of the tree are features. The decision of class label is taken based on these features. The leaf nodes of the tree are class labels. There are several phase in the tree generation. First step is the branch generation where different branches of the trees are generated based on the data instances of the training data. Second step is the pruning phase where noisy branches are pruned from the tree. In the classification phase, when a new data instance is given to the tree for classification, the datapoints is considered as the intermediate nodes based on the features of the instance and mapped to a leaf node. The mapped leaf node is the assigned class label of the test instance. Decision has the advantage of being explainable over other classifiers. Quinlan also proposed C4.5 [33] as an extension of decision tree algorithm which is also very efficient and competitive. Though both decision tree and C4.5 are very competitive performance and the decision process is human understandable, they are prone to overfitting and noise in the training data affects the performance significantly.

Deep learning classification model proposed by Hagan et al. [19] follows the architecture of artificial neural network. The working procedure of this model follows layer by layer data processing like human brain. Deep learning classification model showed excellent performance in comparison to other classification models. Deep learning model consists of connected layers and in each layers there are interconnected nodes. Input data is processed in each of the nodes and they are forwarded to the next layer. Being processed in each layer and each nodes, the data is passed in the output nodes to predict the final class label. Though this architecture has superior result than other classifier there are some drawbacks. High volume of training data is required to train deep learning classifiers which can be a problem in some domain like medical data of a particular disease. Another drawback of deep learning models is the requirement of computing resources is very high. The model needs to store

the processed data from each nodes and each layer. Thus processing this huge amount of data requires high computing resources. As the data are processed and transformed at each layer of the model, the decision process is not human understandable which can be a problem in the case of working with sensitive data like financial decision making. Because if there is a bias or lots of noise in the data, in the case of deep learning models, there is no way to understand its effect from the model.

For any classification model, it is very important to understand the decision process. Associative classifiers are interpretable and human understandable as they generate rules and make the classification based on the generated rules. The rules generated by associative classifier are human understandable. We will discuss associative classifiers in the next section.

## 2.2 Associative Classification

Associative rule mining finds relations among the features in the transaction data in terms of their association and correlation. For example, association rule mining finds the relation among item 'A' and item 'B' from transactions by counting the transaction where item 'A' appears. Then it counts in how many of those transaction item 'B' appears. This simple strategy becomes very important in different scenario like the retail stores to increase their sales and profits. This strategy can be helpful to build a classification model. The rules which are generated by the association rule mining is in the form of $X \rightarrow Y$ where $X$, the antecedent, is a conjunction of co-occurring features, and $Y$, the consequent, is a class label. The relation between the features to the class label can be determined by associative rule mining approach. Further using these rules, the class label of the test data can be predicted. Being inspired from this idea, researchers studied associative classifier extensively in the last few decades.

The concept of using association rules as class association rules was first proposed by Bayardo [8]. He used apriori rule generation approach and few rule pruning technique to increase the efficiency by extracting rules with high

confidence. Liu et al. [29] proposed Classification Based on Association (CBA) which uses an Apriori based rule generation approach to generate class association rules. In their approach, all constrained association rules from the database are generated and ranked based on metrics. The highest matching rule is used for the classification task. There are three steps in associative classification process. They are rule generation, rule pruning and rule selection. Rule generation is the first approach of the associative classification process. As we stated earlier, the class association rules are in the form of $X \rightarrow Y$ where $X$, the antecedent, is a conjunction of co-occurring features, and $Y$, the consequent, is a class label. The rules are generated in such a way that the antecedent and the consequent class label co-occurs together in transactions. Two important measures of the rules are support and confidence. Support is calculated by the proportion of the dataset where the rules appear in the dataset. Confidence is the relationship between antecedent and consequent that is in fraction of total transactions containing the antecedent the antecedent and the consequent appears together. Class association rules are a special type of association rules where the antecedent are the features and the consequent is the class label. Apriori proposed by Agarwal et al. [1] is the most widely used rule mining strategy. Another famous rule mining strategy is fp-growth algorithm [22]. Almost all the classifiers use one of these two strategies in rule generation.

The next step is rule pruning strategy. In the rule generation phase many noisy rules are generated which affect the performance of the classifier. The rule pruning phase is very important as this removes the rules which does not add any additional information to the classification process. Different rule pruning strategies have been proposed by different researchers over the years. In CBA, [29] database coverage strategy has been used. In another study, Zaiane and Antonie [39] proposed pruning based on finding minimal set of class association rules. They aimed at finding minimum set of rules for classification without jeopardizing the performance of the classifier.

The final step of the classification is to select the rules for predicting the class label of a test instance. After the rule generation and rule pruning steps,

the number of rules still might be large and all of them might not be applicable to the test instance. Thus the applicable rules are to be selected from the pool of the rules. After selecting the applicable rules, in most of the cases the number of applicable rules is more than one. Among the applicable rules, the best rules are to be selected following heuristics. Several heuristics are there to select rules to apply on the test instance. Some of the heuristics are sum of confidence values, sum of support values, average of confidence values etc which are used to find most effective rules. For each of the test instances the rules are selected based on the heuristics. In the study of Sood and Zaiane [34], they used three different heuristics namely sum of ln(p-value), sum of confidence value and sum of ln(p-value)*confidence for for the rules of each of the class labels. From their experiment, they found in most of the cases the rules with the highest sum of the confidence values provide better result.

These three major steps are followed in all the associative classifiers. We will describe the associative classifiers in next section.

## 2.3 Associative Classifier

Liu et al. [29] showed it is possible to use association rule mining to build a classifier. Their proposed Classification Based on Association (CBA) uses an Apriori based approach to generate class association rules. In their approach, all constrained association rules from the database are generated and ranked based on metrics. The highest matching rule is used for the classification task. Their subsequent work [30] attempts to decrease noisy rules. They improved the performance of the classifier by choosing the most accurate class association rules for the classification task. Being inspired from this work, many researchers came forward with different approach to improve the performance of associative classifier, mainly differentiating themselves by proposing different strategies to select the applicable rules during inference. Yin and Han [38] proposed Classification based on Predictive Association Rules (CPAR). They generate association rules directly from the training data by a greedy approach. To evaluate each of the generated rules, they use an expected accu-

racy and finally select the best $k$ rules for the classification task. Li et al. [28] proposed CMAR which is Classification based on Multiple Association Rules. They extended FP-growth [22] frequent pattern mining method to construct an FP-tree for class distribution association. In this approach the class association rules are stored in a special data structure. The generated rules are pruned based on the confidence, correlation and database coverage. Using multiple strong association rules with a Weighted $\chi^2$ measure, a datapoint is labeled to the appropriate class.

Antonie and Zaiane [2] propose another Association rule based classifier that they apply for text categorization. They put forward two different approaches; ARC-AC, considering all generated rules from the whole training set, and ARC-BC, where data from each class is mined separately allowing the handling of unbalanced datasets. Indeed, being based on frequency, minority classes risk not being expressed in a predictive model when all dataset entries are mined together. Another approach, CCCS, proposed by Arunasalam and Chawla [5] introduces a measure named "Complement Class Support" (CCS). The authors claim CCS guarantees a positive correlation among the class label and the generated rules. Antonie et al. [3] also propose a two stage associative classifier where associative classification rules are discovered in a first stage and in a second stage, another algorithm learns how to use those rules for class prediction. Instead of basing the selection of rules to apply during inference on some heuristics, they use a neural network to learn how to predict the best rules to apply. This approach showed improved efficiency in terms of accuracy. One limitation of associative classifiers is the generation and the need to evaluate a huge number of rules, among them many are noisy rules. To overcome this problem Zaiane and Antonie [39] performed an extensive study with the focus of reducing the number of rules without decreasing the accuracy of the classifier. The authors propose a new pruning strategy to reduce the number of class association rules. They also propose different heuristics for selecting rules which can obtain high accuracy for a given instance. However, it remains that the proper support and confidence values have to be selected and tuned. This is one major drawback of associative classifiers inherited from association

rule mining. The performance of the model largely depends on these values. It is a tedious task to find the proper confidence and support values for each of the dataset, hampering the adoption of associative classifiers.

To solve this issue, Li and Zaiane [27] propose SigDirect to find the statistically significant rules for classification, instead of frequent ones. In their strategy they improvised the kingfisher algorithm [21]. To find out whether a rule is significant or not, they used Fisher's exact test to measure the significance of a rule. In this method they directly mined rules from the data and pruned them by following instance-centric pruning strategy. Their method solves the limitation of requiring support and confidence values. This method also solves some other limitations of the previous associative classifier. In previous classifier no statistical dependency between the antecedent and consequent was considered.Thus many rules were generated which instead of adding further information, affected the performance of the model. In addition to this, using support-confidence framework includes a large number of rules. But considering statistical significance considers only those rules which adds significant information towards the classification task. Their proposed method SigDirect increases the accuracy of the classifier. The main contribution of their work is eliminating the necessity of the annoying support and confidence thresholds. However, SigDirect also has some limitation. Sood and Zaiane [34] showed though Sigdirect produces very less number of rules comparing to other associative classifier, noisy rules still exists. In their study they showed, those noisy rules can be further eliminated and elimination of noisy rules increases the efficiency of the model. Thus they proposed SigD2, a two stage pruning technique which can reduce the number of rules without jeopardising the accuracy of the classifier and therefore making a learned model even more practically interpretable.

Though Sigdirect and its successor SigD2 shows competitive result in terms of accuracy, with the increase of the features in training dataset, memory requirement and runtime of the dataset increases rapidly. To solve this issue using ensemble models which we will discuss in next chapters.

## 2.4 Ensemble Classification Model

The concept of ensemble model was first proposed by Dasarathy and Sheela [16], where they proposed the idea of composite model. In They achieved enhanced performance by deploying two or more component classifier each trained of a separate partition of the feature domain. The optimal domain partition criteria depends on the component classifier. Authors used Sequential Weight Increasing Factor Technique (SWIFT) for partitioning the domain space. In their study, they showed when multiple classifiers are deployed together, they achieved better performance than each of the individual component classifier when they are deployed separately. Following their work, several researchers worked on this field and achieved better result using ensemble model. Bagging [9] and Boosting [17] are most popular ensemble model for classification problem. Several variation of bagging and boosting and different strategies to select the subset from the training data have been used in different literature. In bagging, a predefined number of subset of the dataset are selected and one classification model is trained on each of the subset of the dataset. These classification models which are trained on the subset of the training data are know as base classifier of the model. At the end of training models, each of the models make the prediction of the test data independently. At the end, class label with the highest vote is considered as the final class label. In case of bagging approach, all the base learners are trained and make their prediction independently. In bagging approach the base learners can be trained in parallel as each of the base learners are trained independently. In contrast to bagging, in boosting algorithm, base learners are trained in series. After training one base learner, the misclassified instances by are identified and the next base learner put extra weight to the misclassified instances. Weights if the instances which are correctly classified are decreased for the next base learners. The idea is to pay more attention on misclassified instances than the correctly classified instances on the next steps. This procedure continues until the number of base learners reaches to a predefined number or the training error reaches to a predefined threshold. At the end of training the base learners,

voting is performed. In case of boosting algorithm, each of the base learners has a weight based on their performance. This voting strategy is called weighted voting. Though this strategy is susceptible to overfitting, it shows promising result. Later Hastie et al. [23] extended this work for multi-class classification. This boosting strategy is known as adaboosting. Another boosting strategy is gradient boosting proposed by Friedman [18]. In gradient boosting, a loss function is used and the objective is to minimize the loss function using gradient descent method. Gradient boosting also have competitive performance. In [7] the authors showed that bagging and boosting improved classification performance for demographic classification of handwriting. Prusa et al. [31] evaluated 12 different techniques that improve the classification performance on 3 datasets for twitter sentiment classification. Based on their experimental findings, they concluded that bagging provided superior performance as compared to the rest of the techniques and significantly improved the classification performance. Another famous ensemble model is random forest [10]. Random forest follows the idea of bagging procedure and uses decision tree as base learner. Unlike decision trees, random forest uses a random selection of features to select each nodes for each of the trees. In this study author shows, in addition to having competitive performance, random forests are more robust to noise than boosting algorithms.

In the case of all ensemble model, subset selection procedure plays a vital role. One of the main reason of superior performance of the ensemble models is diversity among the base learners. Higher similarity among the subset results in poor performance of the model due to lack of diversity among the base learners. Thus maximizing the diversity among the base learners is one of the main goal in ensemble model. To create diversity, more emphasis should be given on the subset selection strategy. Brown [11] made extensive study to define and measure diverse error in ensemble model. He showed diversity among the base learners can be achieved using negative correlation among the base learners. His work puts emphasis on creating diversity among the base models so that the prediction of each model becomes independent and by this maximum performance from a neural ensemble model can be achieved. Brown

proved his proposition by theoretical and empirical analysis.

One drawback of the classical ensemble model is one has to fix the size of the ensemble. That is the number of the base learner in an ensemble model is predefined. But this process has a limitation. If the predefined number of base learner is large and the number of feature in the training data is small then this will result in many duplicate base learners which affects the diversity among the base learners. On the other hand if the number of feature is large and the number of predefined base learner is small then the base learners might not cover the whole feature space thus many important features might not be considered in the decision process. In classical ensemble models, the subsample to train the base learners are selected randomly. Thus some features can be selected multiple times and some features can be omitted in the process. In random subsampling procedure, there is no way to control the selection of the features. Thus this might create bias in the base learners. To address this issue Cao et al. [12] proposed a novel sampling procedure as Cost sensitive adaptive random subspace ensemble. Their proposed method automatically defines the number of base learners in the ensemble process. The proposed sampling procedure consider the overlap among the subsamples and allow a maximum amount of overlap among each subsample. In this way they ensures the diversity and coverage. When the method can not produce another new subsample with the given constrains, it stops sampling and train one base model with each of the subsample. Thus the requirement of predefined number of subsample is eliminated.

Inspired by the outstanding performance of deep learning architecture, several researchers came forward to make the ensemble of deep learning architecture and ensemble model having layered architecture like deep learning model. Kowsari et al. [25] proposed Random Multi-model Deep Learning, an ensemble of deep architecture to predict the class of different types of data like image, text and tabular data. RMDL uses varieties of optimization technique in the base learners which stabilizes the classification procedure along with using different feature extraction technique for each of the random deep models which work as a base learner for the model. This method also uses majority voting to

determine the final class label. Authors of RMDL showed their model works across different data types like text, images and videos. The main contribution of their work is it solves the problem of finding suitable hyperparameter for deep learning model as RMDL uses random parameters and feature extraction techniques for each of the base learner. Though having impressive performance, deep learning model and ensemble of deep learning model has a basic limitation that is the requirement of large training data which is not available in many cases. Small training data causes overfitting for deep learning model. Zhou et al. [40] tried to overcome some of these limitations of deep neural networks by proposing gcForest, an ensemble of decision trees that exploits the significant features of a neural network like representation learning, ability to deal with high dimensional data and model complexity. In GcForest, training data are processed in layer-wise fashion like deep neural models. Each layer of gcForest is an ensemble of decision trees and the decision trees act as a node. After processing the training data at a layer, the processed data is forwarded to the next layer forming a deep architecture. Proposed architecture have very encouraging performance in several data type like tabular data, text data and image data. After the success of gcForest, several researchers came forward to propose deep ensemble architecture. Tanno et al. [36] proposed another architecture where they used adaptive neural trees whose architecture is hierarchical and can learn the parameters of the model over the progressive growth. The parameters are auto-tuned according to the size of the dataset. Sun et al. [35] also proposed a model which is inspired by the gcForest [40] where they select important features by following some measures for classification. The subsequent phases remain the same as gcForest, but in each step, they choose the important features only. Both the work of Tanno et el. [36] and Sun et al. [35] focus more on image data. Their architecture performs well in the case of classification of an image dataset. Though our focus is on tabular data, their works provide a significant direction for our model because, in our case, we want to develop a model which is efficient for datasets of different sizes.

In chapter 3 we proposed Dynamic Ensemble Associative learners where we

propose a feature sampling technique inspired from the work of Cao et al. [12]. As we discussed in previous subsection, associative classifiers are interpretable. After making an ensemble of associative classifiers, we can still gather the rules from the base learners and interpret the decision process. To make the interpretation process simple, in chapter 4, we propose Classification by Frequent Association Rules(CFAR) inspired from the work of W. Pascal et al. [37]. They proposed Decision snippet Features where instead of classical max-vote strategy, they mined small frequent branches as decision snippets from the learned decision trees of the ensemble. They showed a linear model on top of the snippets provide competitive results and reduces the model size. Being inspired from the performance of gcForest [40], we propose Deep Associative classifier in Chapter 5.

# Chapter 3

# Dynamic Ensemble Associative Learning

In this chapter we provide the details of our proposed model Dynamic Ensemble Associative Learning (DEAL). We further evaluate our model in comparison to other classifiers using UCI datasets.

## 3.1 Methodology

In this section we describe our methodology for the bagging approach using SigD2 as a base learner. As mentioned above, SigD2 was shown to outperform other rule based classifiers and certainly all other associative classifiers in terms of accuracy and number of generated classification rules. In most cases of bagging, the dataset instances are randomly sampled. We instead sample the feature space and use all instances. In most known ensemble approaches, the number of classifiers in the ensemble is predetermined and fixed. Therefore, initially we start proceeding with the same and randomly sampling the feature space for each classifier in the ensemble. This random sampling does not guarantee completeness as some features may never be picked, and does not ensure diversity among the classifiers since different classifiers in the ensemble may end-up with exactly the same features selection. For that reason we later introduce our dynamic sampling technique that ensures completeness and diversity and automatically determines the necessary number of learners in the ensemble. The procedures are explained in detail in the next sub sections.

**Algorithm 1** Subsample generation by randomly selecting features

**Input: features:** all features of the feature space; **N:** Number of the features in each subsample

**Output:** 100 subsample of the feature space

```
all_subsample ← []
for i in range(100) do:
    new_subsample ← []
    n ← 0
    reset features
    while n < N
        f ← randomly select without replacement a feature from features
        add f to new_subsample
        n ← n + 1
    end while
    add new_subsample to all_subsamples
end for
return all_subsamples
```

### 3.1.1 Ensemble with random subsample

At first, we use 100 base learners in the ensemble model. Each of the base learners is trained on a random subsample of size $N$ of the original feature space. We show the process of creating random subsamples in Algorithm 1. Following this method, we generate 100 subsamples and train the base learners with each of the subsamples. For the random selection of all the experiments, we use *random* python library and *time* as seed. During inference, the prediction of the class label is done with each of the base learners and a vote determines the final prediction.

Generating subsample in this way presents some limitation. A feature can be selected every time or some of the features might not be selected at all. Thus many important features which are strongly correlated with the class label can be excluded from the training process. Another limitation is the need to fix the number of base learners, in our case 100, based on common practice in the literature. If the number of features is not large enough, there is no need for 100 base learners. Indeed many base learners would be trained on the same subsample of the feature space. To solve these issues we propose a new feature sampling technique which addresses both of the limitations.

### 3.1.2 Dynamic Ensemble Associative Learning

Our goal is to make sure all input features are used by at least one base learner, and the set of base learners are diverse. Therefore, to subsample from the feature vector, we follow a sequential procedure until all features are selected (ensuring completeness). To ensure diversity, we make sure the feature spaces of any pair of base learners do not overlap more than some percentage threshold. Our method to generate diverse subsamples is provided in Algorithm 2 and Algorithm 3. In the input of the algorithm, $N$ is the number of features to be selected in each of the subsamples. From previous literature we know that if the feature vector of the dataset is large, SigD2 requires high volume of memory and run time. Therefore we limited the size of the feature vector using $N$. In our experiments, we have tested across different values for $N$ but in all cases we used small value of $N$. The next parameter of the algorithm is $O_v$. $O_v$ is the maximum allowed overlap between any two subsamples. The value of $O_v$ is a percentage and ranges from 0 to 1. If we put a very small value close to 0 for $O_v$, all the subsamples would be almost unique and the number of generated subsamples small, which results in a small number of base learners, defeating the purpose of the ensemble. If we use a value close to 1, the subsamples would be almost similar to subsamples generated using a random subsampling. Thus we have to find an optimum value for $O_v$. The last parameter is T which indicates how many times we will try to get a satisfactory new subsample with our required overlap. This is to avoid blocking infinite loops. When generating a subsample by selecting features, if there is at least one previously generated subsample with which overlap of the selected features of new subsample is more than $O_v$, we discard the current subsample and generate a new one. T provides a boundary for how many times we try for a single subsample. If after trying T times we do not find any subsample with less than overlap $O_v$, we stop the sampling procedure. We also keep track of whether all the features are already covered by the subsamples or not. If all the features in feature space are covered by the generated subsamples, we stop our sampling procedure.

**Algorithm 2** Algorithm **GenerateSubsample**

**Input: Features:** all features in the feature space; **N:** Number of the features in each subsample, $O_v$**:** Maximum overlap between any two subsamples, **T:** maximum number of tries to generate a subsample

**Output:**Subsamples of the feature space

```
1: try=0;
2: all_subsamples ← []
3: while try < T do
4:    new_subsample ← subsampleGen(Features, N)
5:    if(Overlap(new_subsample)< O_v) then
6:       Add new_subsample to all_subsamples
7:       try ← 0
8:    else
9:       try ← try+1
10:   end if
11:   if (all_feature_covered(all_subsample) ==True) then
12:      break
13:   end if
14: end while
15: return all_subsamples
```

Algorithm 3 shows how a subsample is generated. Here we randomly select a feature from the feature space and add it to the new subsample. $N$ indicates the number of features in each subsample.

**Algorithm 3** Algorithm **SubsampleGen**

**Input: Features:** all features in the feature space; **N:** Number of the features in each subsample

**Output:** subsample of the feature space

```
1: subsample← []
2: n← 0
3: while n < N do
4:    new_feature ← randomly select a feature
5:    Add new_feature to subsample
6:    n ← n+1
7: end while
8: return subsample
```

After generating a new subsample we evaluate whether the features of the newly generated subsample has more overlap than the $O_v$ ratio with any of the previously generated subsamples. Between two subsample $S_1$ and $S_2$ where

the features are $F_{S_1}$ and $F_{S_2}$ respectively and the number of features in each of the subsamples is $N$ then the overlap between the subsamples is calculated using the following equation:

$$Current\_overlap = \frac{F_{S_1} \cap F_{S_2}}{N} \tag{3.1}$$

If there is at-least one previously generated subsample with which the feature overlap is greater than $O_v$, we discard the subsample. If there is no previous subsample with which the newly generated subsample has overlap greater than $O_v$, we consider the current subsample satisfies the required condition and accepts the sample. After generating the subsamples, we train one SigD2

**Algorithm 4** Algorithm **condition_satisfied**

**Input: new_subsample:** newly generated subsample, **all_subsamples:** all generated subsample, $O_v$**:** Maximum overlap between any two subsamples
**Output: True** if condition satisfied, else **False**

```
1: for each sample in all_subsamples do:
2:    current_overlap ← calculate overlap between sample and
   new_subsample
3:    if current_overlap > O_v:
4:       return False
5:    end if
6: end for
7: return True
```

model with each of the generated subsamples. The size of the output set of subsamples determines the size of the ensemble. We use the trained base learners to predict the class label of the test dataset. The class label with the highest vote of the base learners is determined as the final predicted class label. We evaluate the model using the predicted value in terms of accuracy, memory requirement and runtime.

## 3.2 Result Analysis

We use 10 different UCI datasets [6] to test our model. Before using a dataset we discretize the numerical values as stated in [13]. We convert the features to a binary feature vector. We used the same vector form of discretized values

for all our experiments will all contenders. Thus the results of mentioned algorithms can be slightly different from their original papers. For each of the datasets, we use 80% of the data as training data and the rest 20% as test data. We compare the result of our model DEAL with different values of $N$ which is the number of features for each base learner, against the result of C4.5 [33], Random Forest [10], RIPPER [14], SigD2 and the ensemble using 100 random sampling of features, we denote as $RS$. We also experiment with an ensemble of C4.5 as base learner using our proposed feature sampling method DEAL.

The reported result is the average of 20 runs for each of the datasets. We compare the performance of our proposed model in terms of accuracy, memory requirement and runtime. We also provide an analysis on the performance of DEAL for different values of $O_v$. Finally, we discuss the interpretability of the prediction by DEAL.

### 3.2.1 Classification Accuracy

We compare the performance of our proposed method with SigD2 since it was demonstrated to have better performance than other associative classifiers, rule-based classifiers [27], [34]. We include RIPPER, however, since it is a rule induction algorithm different from association rules. We also compared the performance of DEAL with C4.5 and an ensemble of C4.5. As a representative of the traditional ensemble approaches, we use Random Forest [10] to compare our result. In the case of random sampling of the features (RS), we experiment with different values of $N$ (i.e. number of features in each subsample) and different values of $O_v$ (i.e. maximum overlap among the selected features of base learners). From the experiment, we find the optimum result when the value of $N$ is between 25 and 30.

| Dataset | #cls | #rec | Feature vector Size | C4.5 | C4.5 with Dynamic sampling | RF | RIPPER | SigD2 | RS N=25 | RS N=30 | DEAL N=20 $O_v$ = .6 | DEAL N=25 $O_v$ = .6 | DEAL N=30 $O_v$ = .6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Zoo | 7 | 101 | 35 | 95.24 | 95.24 | **97.62** | 85.48 | 94.55 | 85.53 | 85.53 | 91.66 | 92.85 | **97.62** |
| Pima | 2 | 768 | 36 | 76.62 | 73.38 | 67.41 | 76.08 | 76.44 | 77.21 | 79.81 | 75.26 | **79.82** | 77.61 |
| PageBlocks | 5 | 5473 | 41 | 93.15 | 92.23 | **93.51** | 80.37 | 91.67 | 90.04 | 91.23 | 84.16 | 91.84 | 88.53 |
| Heart | 5 | 303 | 47 | 50.82 | 52.45 | 52.45 | **64.90** | 45.90 | 46.86 | 48.51 | 47.54 | 49.18 | 49.18 |
| Hepatitis | 2 | 155 | 54 | 70.97 | 74.19 | 78.38 | 78.72 | 81.93 | 80.65 | 81.29 | 78.06 | 80.64 | **82.58** |
| Wine | 3 | 178 | 65 | 91.67 | 91.67 | 77.78 | 93.33 | 92.7 | 91.57 | 93.26 | 86.51 | **94.38** | 92.13 |
| Anneal | 6 | 898 | 67 | 97.22 | 86.67 | **98.89** | 97.5 | 92.22 | 84.41 | 88.86 | 80.51 | 83.85 | 92.2 |
| Horse | 2 | 368 | 83 | 78.38 | 75.67 | 75.67 | 80.56 | 78.80 | 70.38 | 81.25 | 80.43 | **82.33** | 81.79 |
| Adult | 2 | 48842 | 95 | 84.06 | 82.31 | **84.97** | 83.91 | 84.59 | 79.67 | 81.81 | 80.52 | 84.92 | 81.51 |
| Ionosphere | 2 | 336 | 155 | 88.73 | **95.78** | **95.78** | 92.81 | 90.18 | 88.1 | 90.77 | 90.47 | 92.85 | 94.04 |
| Average | | | | 82.69 | 81.96 | 82.25 | 82.77 | 82.89 | 79.44 | 83.37 | 79.512 | 83.27 | **83.72** |

Table 3.1: Accuracy of C4.5, Ensemble of C4.5 using feature sampling of DEAL, Random Forest (RF), RIPPER, bagging using random sampling (RS), SigD2, and DEAL.

The comparison in accuracy of C4.5, ensemble of C4.5 using proposed sampling method, Random Forest (RF), RIPPER, SigD2, Random feature Sampling method (RS) with DEAL using different values of $N$ are provided in Table 3.1 and Figure 3.1. From the table we can see, among the datasets, in comparison with Random Forest and C4.5, Random Forest has better performance than DEAL in 4 of the datasets while DEAL is a close runner-up, and in 4 others DEAL has better performance, with one ex aequo with the zoo dataset. In one dataset RIPPER performed better than DEAL. On average DEAL is better. In comparison with the original SigD2, we can see that apart from the Anneal dataset, DEAL presents an improvement for the accuracy. The probable cause might be that some of the feature vectors in the Anneal dataset being highly correlated which plays an important role in the class prediction. DEAL selects a subset of the feature vectors thus the feature vectors that are needed to be together get separated. Further investigation is required for this dataset to find how the important features can be kept together. That might improve the performance of DEAL in those datasets where some feature vectors together play an important role in class prediction. The random sampling of the features with 100 base learners could not perform at the same level as DEAL. In comparison with SigD2 sometimes it improves the accuracy and in some cases the accuracy is decreased. As the random sampling is completely random, many important features might not be selected for any of the base learners at all. This is the reason of the poor performance of RS. On the average, DEAL with a feature vector size $N = 30$ and $O_v = 0.6$ has the best performance. We also compare the performance of DEAL with other models in terms of precision and recall. Figure 3.2 and 3.3 shows the precision and recall of the models for each dataset respectively. We can see DEAL, either with $N = 25$ or $N = 30$, has very competitive precision and recall compared to other models.

We can get another interesting observation from Table 3.1. With the increase of $N$, the number of features for each of the base learners, the average accuracy increases. For better understanding of this, we experimented with different values of N. The average accuracy for different values of N is plotted

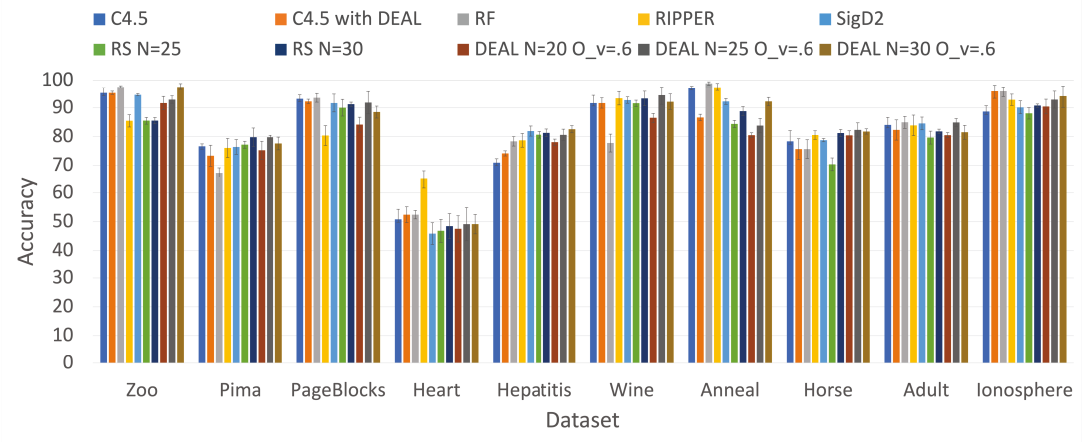


Figure 3.1: Accuracy of C4.5, C4.5 using sampling of Deal, Random Forest (RF), RIPPER SigD2, and DEAL.

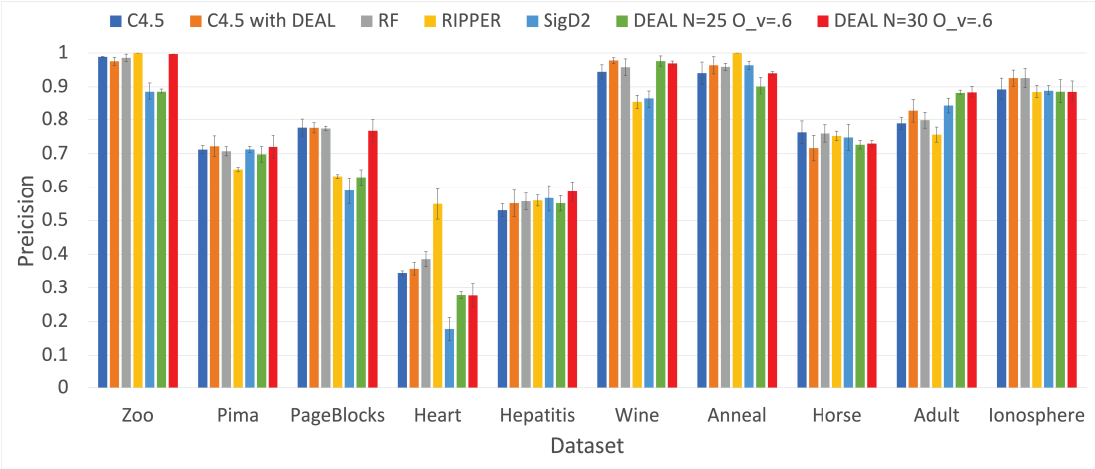


Figure 3.2: Precision of C4.5, C4.5 using sampling of Deal, Random Forest (RF), RIPPER SigD2, and DEAL.

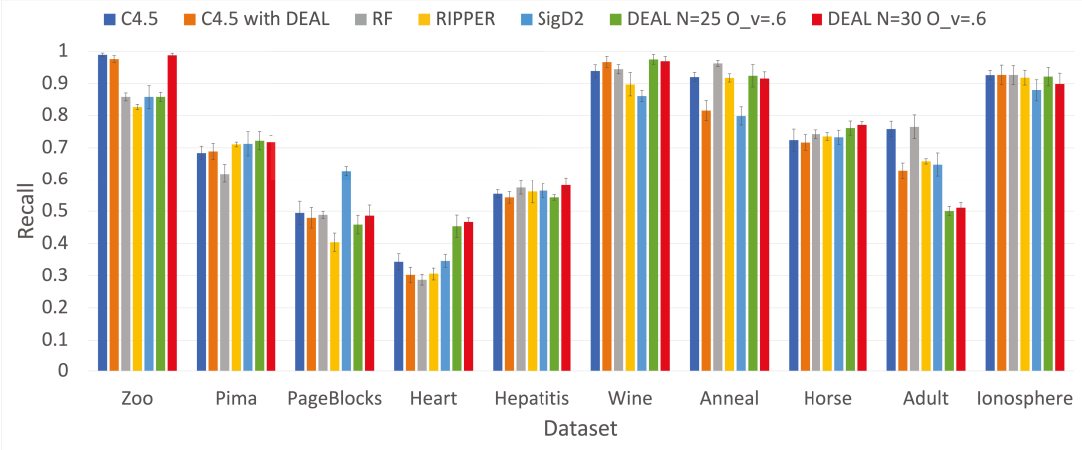


Figure 3.3: Recall of C4.5, C4.5 using sampling of Deal, Random Forest (RF), RIPPER SigD2, and DEAL.

in Figure 3.4. We can see a sharp increase when the value of N is increased from 15 to 25. After 25, the average accuracy continues to increase but not with the same steep rate.

There is a compromise to make with the required memory and runtime, particularly due to the aforementioned limitations of the used base learner SigD2. Therefore, we chose not to increase the number of features per base learner more than 30. We are also interested to know the number of base learners for each of the dataset as we mentioned the number of base learners is determined automatically by DEAL. Table 3.2 provides the number of base learners for each of the dataset. These are significantly less than the typical 100 or even 50 used in the literature for classical ensembles.

| Dataset | Feature vector Size | #Base Learner | Dataset | Feature vector Size | #Base Learner |
|---|---|---|---|---|---|
| Zoo | 35 | 4 | Wine | 65 | 13 |
| Pima | 36 | 4 | Anneal | 67 | 12 |
| PageBlocks | 41 | 8 | Horse | 83 | 19 |
| Heart | 47 | 7 | Adult | 95 | 21 |
| Hepatitis | 54 | 10 | Ionosphere | 155 | 45 |

Table 3.2: Number of base learners using $N = 30$ and $O_v = 0.6$

To understand the effect of different values of $O_v$ we also experiment with

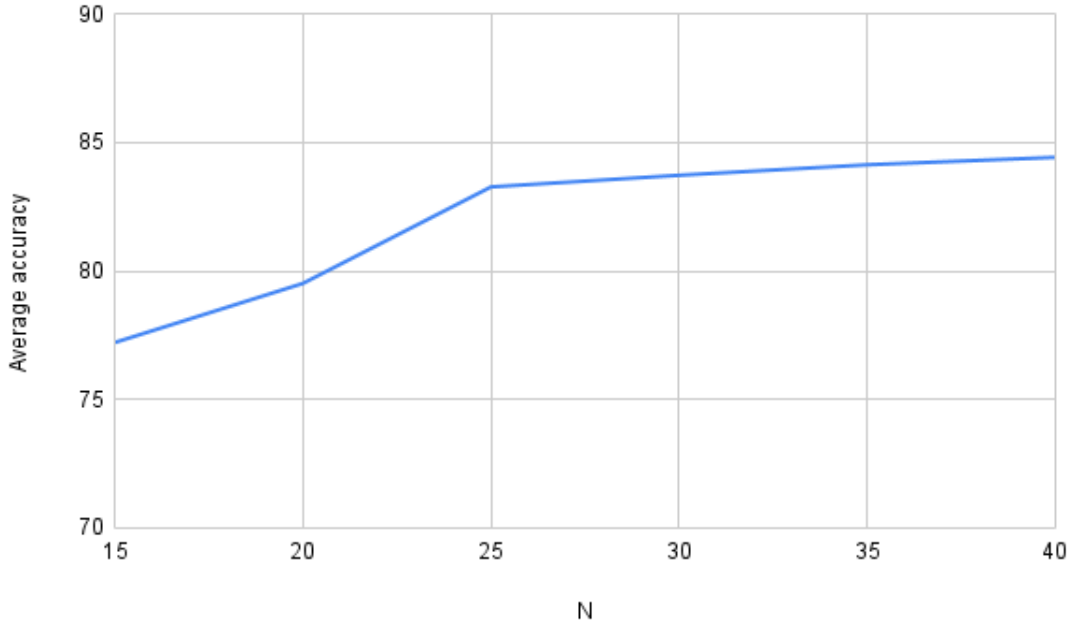


Figure 3.4: Average accuracy of DEAL for different values of N(Number of selected feature for each base learner)

values starting from 0.1 to 0.9 increasing 0.1 in each step. The result is provided in Figure 3.5. From the figure we can see, the performance of the model is best when the value of $O_v$ is in the middle. Because a small value results in almost unique base learners and a very small number of base learners. In the case of a large value of $O_v$, the results would allow repetitions of base learners like with simple random sampling procedure. Thus in both cases the performance is affected. We can see good performance when the value of $O_v$ is between 0.4 to 0.6 for N=30 and 0.5 to 0.7 when N=25 for the dataset we used in our experiments. Thus overlap between 0.4 to 0.7 provides optimum result for most cases.

To understand whether the improvement in accuracy by DEAL is significant or not, we performed a statistical significance test. We carried-out a student paired t-test with the null hypothesis, that the improvement in the accuracy of DEAL is not significant. We run the whole experiment 20 times and recorded the average accuracy each time for each model. For DEAL we

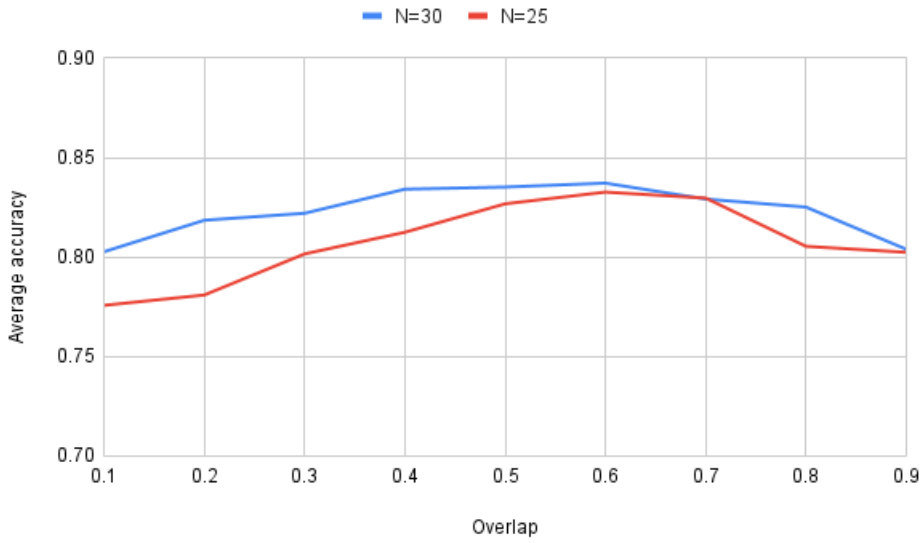


Figure 3.5: Average accuracy across different values of $O_v$

used $N = 30$, $O_v = 0.6$. The difference in the accuracy in each pair is calculated by Equation 3.2.

$$\delta = \mathcal{M}_{DEAL}(test\_data) - \mathcal{M}_i(test\_data) \tag{3.2}$$

Here, $\mathcal{M}_{DEAL}$ is our proposed model and the subscript i stands for the other models. Running the models on test data provides the accuracy of the models and $\delta$ is the difference in the accuracy.

We performed the paired t-test followed by the work of Henry et al. [24] for pairwise comparison among the models and calculated the p-value. If the calculated p-value is less than the threshold alpha(0.05) then we can say the improvement by the proposed model is significant. The calculated p-values are provided in Table 3.3.

| Algorithm | p-value |
|---|---|
| DEAL vs C4.5 | 0.0036 |
| DEAL vs C4.5 ensemble | 0.0077 |
| DEAL vs Random Forest | 0.0087 |
| DEAL vs SigD2 | 0.0046 |
| DEAL vs RIPPER | 0.1190 |

Table 3.3: Statistical result

From Table 3.3 we can see the p-value for each pair is less than the threshold alpha(0.05) except for DEAL vs RIPPER. Thus, except RIPPER, for the other models we can say, the improvement in accuracy of our proposed over other models is significant.

### 3.2.2 Memory Requirements

One of the main goals of DEAL is to reduce the memory requirements for datasets having large feature vector size. In previous sections we showed that DEAL does not reduce the accuracy in comparison to the original SigD2 rather in almost all datasets, it increases the accuracy. Using DEAL we can reduce the memory requirements for datasets having a large number of features. The memory requirements of C4.5, ensemble of C4.5 using proposed feature sampling method, Random Forest, SigD2 and DEAL using $N$ as 25 and $O_v$ as 0.6 are shown in Table 3.4. Here we did not provide memory requirement using the random sampling procedure as for any given number of feature vectors in base learners, the memory requirement of SigD2 is the same. In the memory requirements, the reported result for DEAL is the highest memory used by a base learner in the process as we run the base learners sequentially. In the case of Random Forest, the trees in the forest are executed in parallel. Thus the memory requirement reported here is the memory required for running 100 decision tree in parallel.

Running base learners in parallel has a huge advantage for runtime, which we discuss in the next subsection. Further studies are required to understand whether this trade-off between memory requirement and runtime is feasible by executing the base learners of DEAL in parallel. From Table 3.4 we can see, DEAL decreases the memory requirements even in comparison with Random Forest. The memory requirement in SigD2 increases with the increase in features. One main reason behind this is when working with a large size feature vector, SigD2 has to go through a huge number of class association rules (CAR). CAR increases with the increase of the feature vector size. But in our approach, with a dataset with large feature vector size, the number or rules for each of the base learners is reduced dramatically. In case of a dataset having

| Dataset | C4.5 | C4.5 with Deal | Random Forest | RIPPER | SigD2 | DEAL |
|---|---|---|---|---|---|---|
| Zoo | 106 | 108 | 127 | 107 | 121 | **105** |
| Pima | 103 | 109 | 128 | 102 | 102 | **101** |
| PageBlocks | 111 | 117 | 134 | 111 | 112 | **110** |
| Heart | 105 | 112 | 129 | **104** | 107 | **104** |
| Hepatitis | 103 | 111 | 127 | 102 | 113 | **100** |
| Wine | 104 | 112 | 128 | 105 | 165 | **100** |
| Anneal | 107 | 113 | 129 | 107 | 150 | **106** |
| Horse | 109 | 112 | 128 | 109 | 235 | **106** |
| Adult | 195 | 211 | 252 | **124** | 257 | 252 |
| Ionosphere | 124 | 117 | 128 | 112 | 2519 | **108** |

Table 3.4: Memory requirement(in Megabyte) by contenders and DEAL($N = 30$ and $O_v = 0.6$)

very large size of feature vector, our proposed sampling method decreases the memory requirement to a significant extent.

### 3.2.3 Runtime

We also measure the runtime of DEAL for the same datasets in comparison to Random Forest and SigD2. The runtime of the methods are provided in Table 3.5. In this table we do not include random sampling (RS) as in random sampling we have to train 100 base learners in a sequential manner which would always require a longer runtime than DEAL. From Table 3.5, we can see Random Forest is always the fastest except for Zoo dataset where C4.5 is faster. Despite the fact that DEAL is an ensemble and we are not running the base learners in parallel, DEAL is still faster than SigD2 in many cases. SigD2 beats DEAL when the input feature space is small, because for datasets with a small feature space, DEAL still has to go through the sampling and for not having enough feature, DEAL has to sample many subsample and compare with the previously generated subsamples which increases the runtime. However, with larger feature vectors, DEAL outperforms SigD2.

### 3.2.4 Interpretability of the model

One of the major advantages of our ensemble model is that the prediction of the model is interpretable. After creating subsamples of the feature space and training the base learners with each of the subsamples, base learners uncover classification rules expressed in their respective feature subspace. When predicting the class label of an instance, base learners use their own rules. There are applicable rules, applicable to the instance case, and there are applied rules, effectively used for the decision by each base learner. The applied rules that agree with the majority vote are kept. Using DEAL, we can easily gather all the applied rules selected by base learners. Those would be part of the decision explanation. In addition, the set of applied rules and the applicable rules can be used to rank features by importance vis-à-vis the final prediction. Rules gathered from the base learners can be grouped according to the class label, and ranked based on their frequency among the base learners. Features are ranked by importance for a particular class label prediction. The more a feature appears in an applied rule, the more important it is. Features appearing in applicable rules get an additional boost of importance. Table 3.6 gives a glimpse of the individual interpretable models learned by the different base learners. In the case of the Zoo dataset, there are 4 base learners. The learned models are sets of human readable classification rules with their strength measures, typically sorted by these measures). In Table 3.6 we provide a subset of these rules and the feature subspace for each base learner. At inference time, each base learner finds the applicable rules for that instance and selects the rules to apply for the decision. Table 3.7 shows the applicable rules per base learner for a given test instance as well as the decision for each learner before the vote. We can see that the consensus is for class 5 in this case and that there is agreement for the importance of features 16 and 28. What constitutes the decision explanation is the list of important features but also the list of applied rules as well as applicable rules with the final class label decision as their consequent.

| Dataset | C4.5 | C4.5 with Deal | Random Forest | RIPPER | SigD2 | DEAL |
|---|---|---|---|---|---|---|
| Zoo | **0.06** | 0.58 | 0.12 | 0.08 | 3.64 | 6.65 |
| Pima | 1.76 | 3.84 | **0.15** | 0.41 | 0.47 | 0.89 |
| PageBlocks | 21.04 | 35.83 | **0.24** | 0.46 | 10.23 | 17.86 |
| Heart | 0.69 | 3.04 | **0.13** | 0.40 | 8.55 | 3.11 |
| Hepatitis | 0.23 | 2.62 | **0.11** | 0.26 | 10.41 | 1.25 |
| Wine | 0.17 | 1.27 | **0.11** | 0.33 | 0.46 | 1.05 |
| Anneal | 2.1 | 9.46 | **0.14** | 0.56 | 7.99 | 5.38 |
| Horse | 0.92 | 6.24 | **0.13** | 0.53 | 32.49 | 2.97 |
| Adult | 13.2 | 19.8 | **3.19** | 171.56 | 263.12 | 167.16 |
| Ionosphere | 1.86 | 16.13 | **0.12** | 0.32 | 1905.44 | 6.22 |

Table 3.5: Runtime (in Seconds) by contenders and DEAL ($N = 30, O_v = 0.6$)

| Base learner 1 | | Base learner 2 | |
|---|---|---|---|
| **Selected features** | [0, 2, 4, 6, 9, 10, 11, 12, 14, 14, 15, 15, 16, 19, 19, 19, 19, 21, 21, 23, 24, 25, 26, 29, 30, 31, 33, 33, 34, 34] | **Selected Features** | [0, 1, 1, 3, 3, 3, 4, 4, 6, 6, 15, 15, 15, 18, 20, 20, 21, 23, 25, 25, 27, 27, 28, 29, 29, 31, 32, 32, 33, 34] |
| **Rules** | 6, 23 → 3;(0.1375,1.000,-29.980)<br>0, 25→ 1;(0.1625,1.000,-33.384)<br>.....<br>2, 34, 19 → 0;(0.3500,0.966,-28.991)<br>31, 2, 33, 14 → 6;(0.1125,0.750,-20.776) | **Rules** | 27 → 6;(0.0250,1.000,-4.475))<br>18, 29 → 6;(0.0750,1.000,-15.090)<br>.....<br>33, 15, 21 → 2;(0.0250,0.667,-5.185)<br>0, 29 → 6;(0.1000,0.615,-14.797) |
| **Base learner 3** | | **Base learner 4** | |
| **Selected features** | [1, 3, 6, 7, 7, 8, 10, 13, 14, 15, 16, 17, 19, 20, 21, 22, 24, 24, 24, 25, 27, 28, 29, 30, 30, 30, 31, 31, 32, 33] | **Selected features** | [0, 0, 1, 1, 1, 2, 3, 4, 5, 7, 8, 9, 9, 10, 11, 12, 12, 14, 16, 17, 17, 18, 20, 21, 22, 23, 28, 28, 30, 34] |
| **Rules** | 27 → 6;(0.0250,1.000,-4.475))<br>3 → 1;(0.1625,1.000,-33.384)<br>.....<br>15, 6, 19 → 4;(0.0375,0.600,-9.014)<br>10, 6, 17, 8 → 2;(0.0375,0.600,-7.647) | **Rules** | 7 → 0;(0.4500,1.000,-52.636)<br>3 → 1;(0.1625,1.000,-33.384)<br>.....<br>0, 14, 17 → 1;(0.1625,0.929,-30.745)<br>0, 16 → 6;(0.1125,0.750,-20.776) |

Table 3.6: Rules from the learned model in the form of "Antecedent -¿ Class label;(support, confidence, -ln(p-value))" where Antecedent is a conjunction of tokenised features. For interpretability, tokens are mapped back to features (attribute-value pairs)

| **Test instance**: [1, 2, 5, 6, 9, 10, 12, 14, 16, 19, 20, 22, 28, 29, 31, 33] | |
|---|---|
| **Vectorized test instance:** [0,1,1,0,0,1,1,0,0,1,1,0,1,0,1,0,1,0,0,1,1,0,1,0,0,0,0,0,1,1,0,1,0,1,0] | |
| **Applicable rules by BL 1** | **Applicable rules by BL 2** |
| 9, 29 → 5;(0.0250,1.000,-6.267)<br>**16**, 12 → 5;(0.0500,0.667,-11.566)<br>31, **16** → 6;(0.1125,0.750,-20.776) | **28** → 5;(0.0500,0.667,-11.566)<br>1, 20 → 0;(0.4250,1.000,-45.694) |
| **Predicted class label: 6** | **Predicted class label: 5** |
| **Applicable rules by BL 3** | **Applicable rules by BL 4** |
| 19, **28** → 5;(0.0500,1.000,-14.274) | 10, **28** → 5;(0.0500,1.000,-14.274) |
| **Predicted class label: 5** | **Predicted class label: 5** |

Table 3.7: Applicable rules from the generated rules by the base learners(**BL**) and decision process of DEAL

# Chapter 4

# Classification by Frequent Association Rules

In Chapter 3, to understand the decision process of our proposed model DEAL, we need to identify the base learners who vote for the final class label and gather the rules from those base learners. If the number of base learners is large, this process can be cumbersome. To simplify the interpretation of the ensemble model, in this chapter we propose Classification by Frequent Association Rules (CFAR) where instead of using majority voting of the base classifiers for the final class label, we gather all the rules from the base learners. After gathering all the rules we rank and select the rules based on their ranking to determine the final class label. In this chapter, we compare the performance of CFAR to DEAL and other machine learning models in terms of accuracy, runtime and required memory.

## 4.1 Methodology

In CFAR we also use SigD2 as base learner and one advantage of making an ensemble of SigD2 as well as other associative classifiers is that we can collect all the rules learned by the base learners from the training data set. These rules are used to label test data to a specified class label. However, selecting all the rules might include noisy rules which might affect the performance of the model. To solve this we select a subset of the rules which enhance the performance of the model. For this, we introduce the term Relative Frequency

Ratio (RFR). After collecting all the rules generated by the base learners, we count the frequency of the rules. We find the Rule having the maximum frequency that we note as *max_frequency*. We calculate the RFR of any rule R by Equation 4.1.

$$Relative\_frequency\_ratio(R) = \frac{Frequency(R)}{max\_frequency} \tag{4.1}$$

To predict the class label, we need to select rules from the generated rules. We consider a threshold T and select all the rules whose RFR is greater than or equal to T. The process of selecting rules is provided in Algorithm 5.

**Algorithm 5** Rules selection based on RFR

**Input: Rules:** all rules generated by base learners; **T:** Threshold value to select Rules.
**Output: Selected_rules:** rules with RFR $\geq$ T

```
Selected_rules ← []
for r in Rules do:
    if RFR(r) ≥ T:
        Selected_rules.append(r)
    end if
end for
return Selected_rules
```

In this approach we need to find an optimum value of T. But with our experiment we found this value to be different for different datasets. We show the accuracy result for our 10 datasets for different values of T in Figure 4.1 and Figure 4.2. We split the results of the 10 datasets in two figures for clarity.

Unfortunately as depicted on Figure 4.1 and Figure 4.2, there is no optimum value for T for all datasets. Therefore we design our model in such a way that the model itself can find the value of T for which we can get maximum accuracy. To achieve this goal, we outline a 3 step ensemble classifier. We describe each of the steps below:

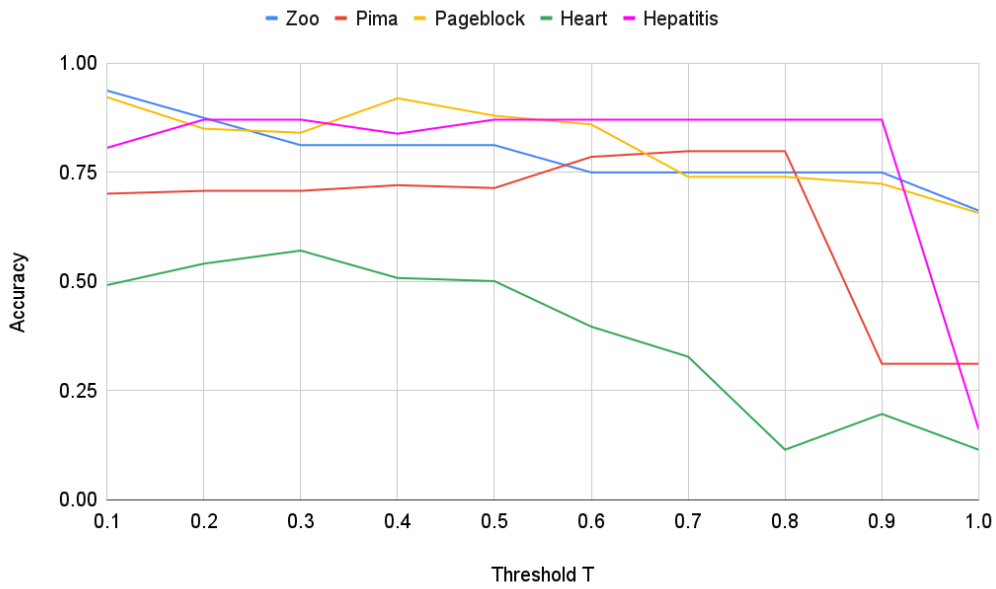


Figure 4.1: Performance of the model for different values of Threshold T

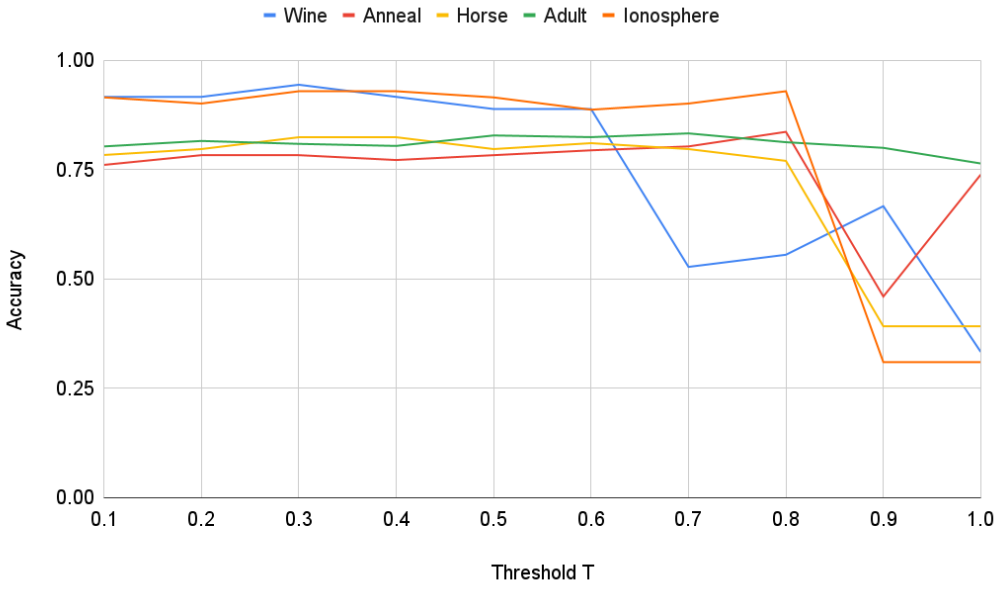


Figure 4.2: Performance of the model for different values of Threshold T

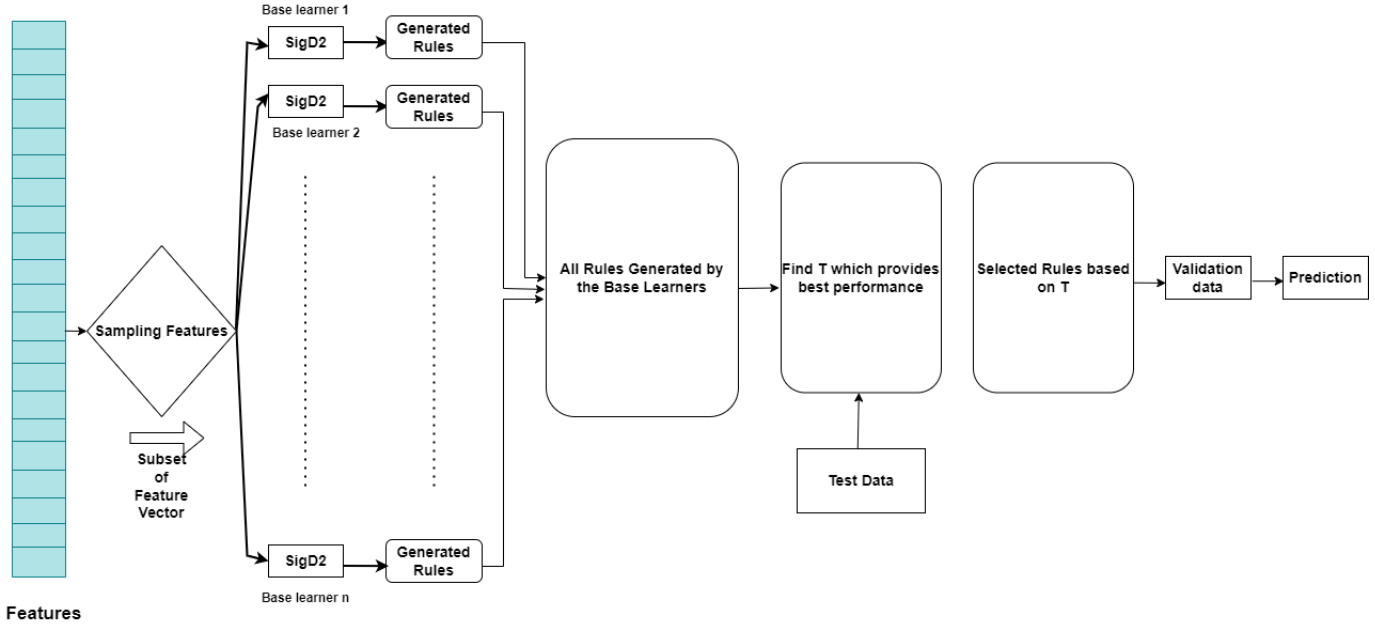


Figure 4.3: Proposed model of CFAR: We generate rules by base learners using training data. We apply generated rules on test data to find the T which provides best performance. With the help of T we select final rules and apply them on validation data to calculate performance of the model

### Rule generation

In the rule generation phase, at first we try with the dynamic subsampling procedure of DEAL but the dynamic strategy could not perform well in this case. We will discuss on this in the later section. For CFAR, we follow the random sub-sampling procedure and train 100 SigD2 base learners. We divide the dataset into train and validation data. We take 80% of the whole dataset as training data and the rest 20% as validation data. Then again we divide the training data as training and test data in the ratio of 80% and 20% respectively. To train each base learner we take a subset of the feature vector of size 30. After training 100 base learners with the training data, we gather all the rules generated by the base learners. We calculate the Relative Frequency Ratio (RFR) for each of the rules using Equation 4.1.

### Find optimum value for T

To find the optimum value for T, we try with different values in a linear search fashion. We assume, the importance of a rule depends on its frequency among the base learners. A rule being more frequent among base learners indicates more importance. Thus while experimenting with different values of T, at first we try with $T = 1$. We select rules from the generated rules using Algorithm 5 and predict the class label with the selected rules. In the next step, we decrease the value of T by 0.1 upto 1.0 and with that value we perform a rule selection and class prediction. In each step, we calculate the accuracy. From there we determine the optimal value of T.

### Prediction

The last step is the prediction. From the previous step we get the value of T which provides best result. We select rules using that value of T and predict the class label of the validation data. With the predicted class label, we calculate the performance of the model. The whole architecture is shown in Figure 4.3.

## 4.2 Result Analysis

We use 10 different UCI datasets [6] to evaluate our proposed model CFAR. Before using a dataset we discretize the numerical values as stated in [13]. We convert the features to a binary feature vector. We use the same vector form of discretized values for all our experiments with all contenders. Thus the results of mentioned algorithms can be slightly different from their original papers. As we mentioned earlier, to test our model CFAR, we use 20% of the dataset to validate our model. We further divide the rest 80% of the data into train and test data in the ratio of 80% and 20% respectively. For all other models, we use 80% of the data to train the model and rest 20% of the data is used as test data.

### 4.2.1 Performance evaluation

We compare our model with SigD2 as Sood and Zaiane [34] showed SigD2 outperforms other associative classifiers, rule-based classifiers, and other state-of-the-art machine learning models. To compare the result with another interpretable model we consider C4.5 proposed by Quinlan [33]. As we are making an ensemble architecture we considered the performance of random forest [10] which is an ensemble of decision trees. Our idea was motivated by the work of Decision snippet features (DSF) model [37], so we are also interested to compare the performance of our model with it. We also considered the classical ensemble architecture where we used 100 base learners each trained with a subset of the feature vector of size 30. As we do not control the overlap of the features between the subsamples there can be different overlap between the subsamples. We measured the overlap between the subsamples and we provide the information about overlap among the subsamples in Table 4.1. The comparison in the accuracy of SigD2, C4.5, Random forest, DSF model, SigD2, classical ensemble approach, DEAL, CFAR with dynamic ensemble and CFAR is provided in Table 4.2. We also plot the accuracy in Figure 4.5.

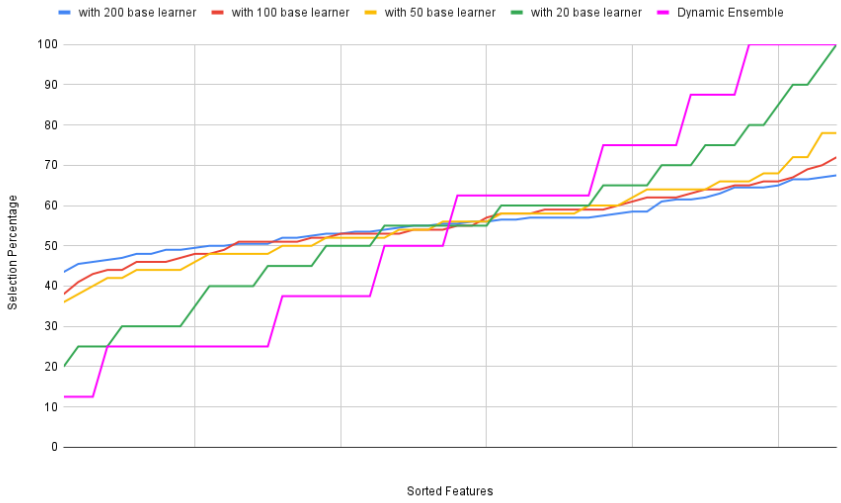


Figure 4.4: Sorted frequency of appearance of the features in CFAR

| | |
|---|---|
| Maximum Overlap | 0.67 |
| Minimum overlap | 0.03 |
| Median | 0.30 |
| St. Deviation | 0.11 |

Table 4.1: Maximum, minimum, median and standard deviation of overlap among the subsamples in CFAR

From Table 4.2, we can see CFAR strategy with dynamic sampling procedure of DEAL could not perform well in comparison to the result of CFAR where we use 100 base learners. One reason for this is, CFAR collects and learn popular rules from the base learners. In the case of dynamic sampling we can see the number of base learners from Table 3.2. The number of base learners in dynamic sampling is significantly reduced compared to 100 base learners. Thus the notion of frequency of the rules does not work for dynamic sampling procedure in CFAR. With many base classifiers, CFAR can decide which rules are important for the classification. In the case of dynamic sampling procedure CFAR does not have the capacity of deciding important rules as very few rules are generated. From empirical analysis, it can be seen that in the case of dynamic sampling strategy, CFAR ends up selecting all the rules every time. Thus training a large number of feature is a good choice for CFAR. Another issue is when the number of base learners is small then the features are not

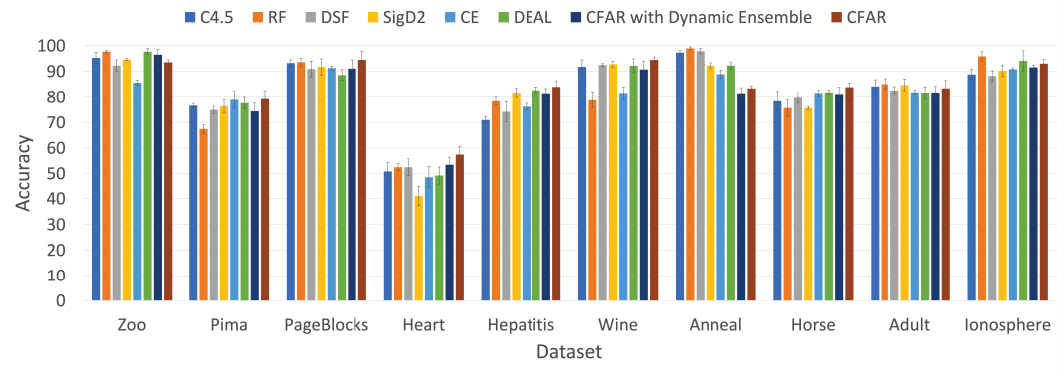


Figure 4.5: Accuracy of C4.5, Random Forest(RF), Decision Snippet Features(DSF), SigD2, Classical ensemble approach(CE), DEAL and CFAR

selected uniformly. We show the frequency of appearance of the features in the base learners in Figure 4.4. From the figure we can see with the increase of base learners the uniformity of selection of the features increases. For this reason we go for 100 base learners with random selection of the features.

From Table 4.2, we can also see in 7 among the 10 datasets CFAR has better performance than other classifiers and ensemble models. The random forest has better performance than CFAR in 3 datasets among which in Anneal dataset the margin is very large. Another interesting observation in Anneal dataset is classical ensemble approach has better performance than CFAR. That means in these two datasets the approach of CFAR where we select the rules based on the Relative Frequency Ratio (RFR) does not perform well. Further analysis is needed on this dataset to understand why CFAR is not performing well. On Average, CFAR has better performance than other models.

| Dataset | #cls | #rec | Feature Vector Size | C4.5 | RF | DSF | SigD2 | CE | DEAL | CFAR with dynamic ensemble | CFAR |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Zoo | 7 | 101 | 35 | 95.24 | **97.62** | 92.23 | 94.55 | 85.53 | **97.62** | 96.43 | 93.47 |
| Pima | 2 | 768 | 36 | 76.62 | 67.41 | 74.92 | 76.44 | 78.96 | 77.61 | 74.38 | **79.22** |
| PageBlocks | 5 | 5473 | 41 | 93.15 | 93.51 | 90.87 | 91.67 | 91.23 | 88.53 | 91.02 | **94.42** |
| Heart | 5 | 303 | 47 | 50.82 | 52.45 | 52.46 | 45.90 | 48.51 | 49.18 | 53.44 | **57.38** |
| Hepatitis | 2 | 155 | 54 | 70.97 | 78.38 | 74.19 | 81.93 | 76.21 | 82.58 | 81.13 | **83.87** |
| Wine | 3 | 178 | 65 | 91.67 | 77.78 | 92.44 | 92.70 | 81.29 | 92.13 | 90.69 | **94.44** |
| Anneal | 6 | 898 | 67 | 97.22 | **98.89** | 97.78 | 92.22 | 88.86 | 92.20 | 81.17 | 83.33 |
| Horse | 2 | 368 | 83 | 78.38 | 75.67 | 79.73 | 75.68 | 81.25 | 81.79 | 80.88 | **83.78** |
| Adult | 2 | 48842 | 95 | 84.06 | 84.97 | 82.53 | **84.59** | 81.81 | 81.51 | 81.71 | 83.31 |
| Ionosphere | 2 | 336 | 155 | 88.73 | **95.78** | 88.18 | 90.18 | 90.77 | 94.04 | 91.55 | 92.96 |
| Average | | | | 82.69 | 82.25 | 82.53 | 82.89 | 80.44 | 83.72 | 82.24 | **84.62** |

Table 4.2: Accuracy of C4.5, Random Forest(RF), Decision Snippet Features(DSF), SigD2, Classical ensemble approach(CE), DEAL, CFAR with dynamic ensemble and CFAR

We also perform a statistical test to understand whether the improvement in accuracy by CFAR is significant or not. For this, we use paired t-test from the work of Hsu and Lachenbruch [24]. In the test, our null hypothesis is the improvement in accuracy by CFAR is not significant. We repeated the whole experiment 20 times for each of the datasets and calculated the accuracy of each of the models. Then we calculated the difference in the accuracy for each pair of the model. With the difference, we calculated the p-value. This p-value signifies whether the difference is significant or not. If the p-value is less than the threshold value alpha(0.05) then we can reject the null hypothesis and say the difference in improvement by our proposed model CFAR is significant. We provide the calculated p-value in Table 4.3. From Table 4.3, we can say, in the dataset we used, except for DSF, CFAR shows significant improvement over accuracy.

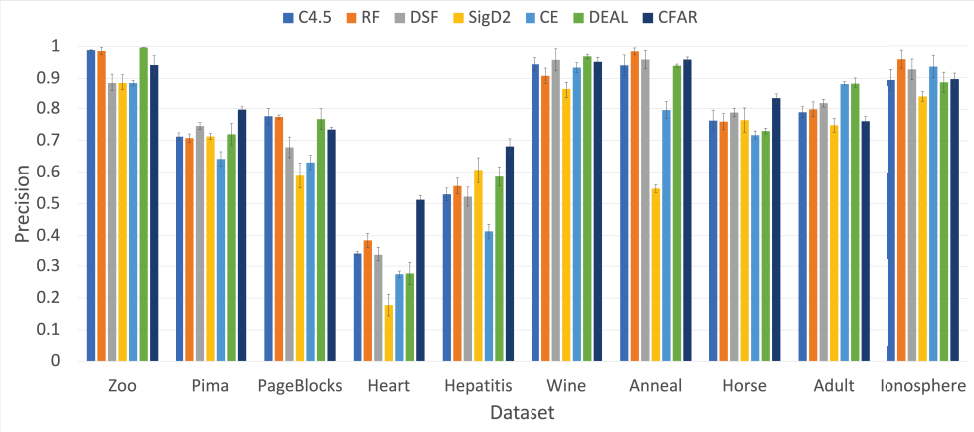


Figure 4.6: Precision of C4.5, Random Forest(RF), Decision Snippet Features(DSF), SigD2, Classical ensemble approach(CE), DEAL and CFAR

| Algorithm | p-value |
|---|---|
| CFAR vs C4.5 | 0.0036 |
| CFAR vs Random Forest | 0.0061 |
| CFAR vs DSF | 0.0007 |
| CFAR vs SigD2 | 0.0001 |
| CFAR vs Classical Ensemble | 5.3317e-11 |
| CFAR vs DEAL | 0.0426 |

Table 4.3: Statistical result

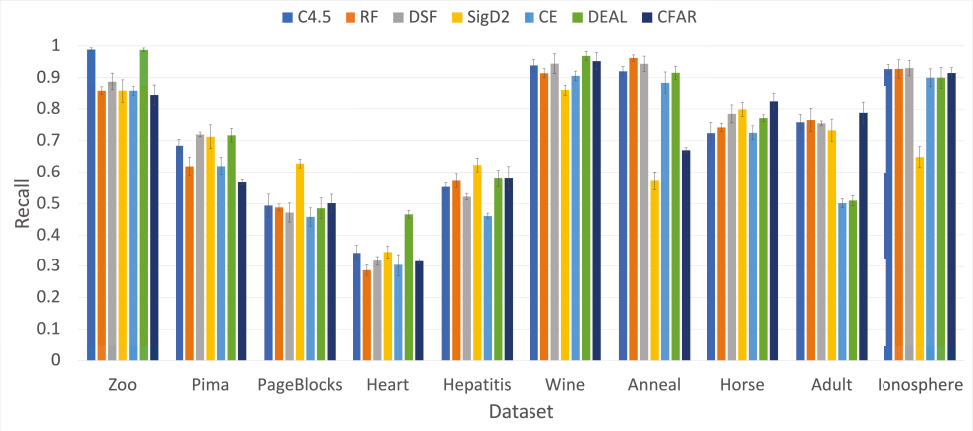


Figure 4.7: Recall of C4.5, Random Forest(RF), Decision Snippet Features(DSF), SigD2, Classical ensemble approach(CE), DEAL and CFAR

From Table 4.3, we can see in 5 of them, the p-value is less than alpha(0.05). In the case of DSF, the p-value is large than the threshold value alpha. Thus we can say, except for DSF, in the dataset we used, CFAR shows significant improvement over accuracy.

Further, we analyze the precision and recall values of the models. Figure 4.6 and Figure 4.7 show the precision and recall values of the models respectively. From the two figures, we can see CFAR has a very competitive performance compared to the other models in terms of precision and recall.

### 4.2.2 Memory Requirement

One of our main goals for making an ensemble of SigD2 is to eliminate the limitation of requiring very high amount of memory with the increase of the feature vector size. We measure the memory required by the models using the python library *psutil*. Table 4.4 shows the memory requirement for SigD2, CFAR and other contenders.

From Table 4.4, we can see, in the comparison of memory requirement between CFAR and SigD2 has two different scenarios. When the size of the feature vector is small, the memory requirement of SigD2 and CFAR are very close. However, with the increase in feature vector size, the memory requirement of SigD2 increases drastically while for CFAR the memory requirement remains stable. Thus we can say, CFAR eliminates the high memory require-

| Dataset | C4.5 | RF | DSF | SigD2 | DEAL | CFAR |
|---|---|---|---|---|---|---|
| Zoo | 106 | 127 | 112 | 121 | **105** | 106 |
| Pima | 103 | 128 | 112 | 102 | **101** | 105 |
| PageBlocks | 111 | 124 | 112 | 112 | 110 | **107** |
| Heart | 105 | 129 | 112 | 107 | **104** | 119 |
| Hepatitis | 103 | 127 | 113 | 113 | **100** | 108 |
| Wine | 104 | 128 | 112 | 165 | **100** | 107 |
| Anneal | 107 | 129 | 113 | 150 | **106** | 108 |
| Horse | 109 | 128 | 112 | 235 | **106** | 108 |
| Adult | 195 | 253 | 196 | 257 | **252** | 255 |
| Ionosphere | 124 | 128 | 113 | 2519 | **108** | 136 |

Table 4.4: Comparision of Memory requirement(MB)

ment of SigD2 in case of a dataset with a large feature vector size. In comparison to DEAL, the memory requirement of CFAR is slightly higher than DEAL.

### 4.2.3 Runtime

Another limitation of SigD2 is the runtime: it increases with the increase of feature vector space. The architecture of CFAR should solve that issue as for each of the base learners, we are taking a fixed-size subset. To understand this, we measure the runtime of CFAR, SigD2 and other contenders. The comparison is shown in Table 4.5. All the experiments are done on the machine with an Intel core i7 processor and 16 GB of RAM.

| Dataset | C4.5 | RF | DSF | SigD2 | DEAL | CFAR |
|---|---|---|---|---|---|---|
| Zoo | **0.06** | 0.12 | 0.31 | 3.64 | 6.65 | 8.23 |
| Pima | 1.76 | 0.15 | **0.10** | 0.47 | 0.89 | 28.32 |
| PageBlocks | 21.04 | **0.24** | 0.86 | 10.23 | 17.86 | 21.38 |
| Heart | 0.69 | 0.13 | **0.34** | 8.55 | 3.11 | 67.51 |
| Hepatitis | 0.23 | **0.11** | 0.08 | 10.41 | 1.25 | 26.58 |
| Wine | 0.17 | 0.11 | **0.07** | 0.46 | 1.05 | 9.61 |
| Anneal | 2.10 | 0.14 | **0.10** | 7.99 | 5.38 | 23.75 |
| Horse | 0.92 | 0.13 | **0.09** | 32.49 | 2.97 | 23.33 |
| Adult | 13.20 | 3.19 | **1.60** | 263.12 | 167.16 | 200.79 |
| Ionosphere | 1.86 | 0.12 | **0.08** | 1905.44 | 6.22 | 27.09 |

Table 4.5: Comparison of Run time(seconds)

From Table 4.5, we can see that C4.5, random forest, and DSF model are faster than SigD2, DEAL and CFAR in classification tasks. This was expected. However, between SigD2 and CFAR, the dataset having a small size of feature vector, SigD2 is faster than CFAR. This is also expected since CFAR has 100 base learners which are trained sequentially. No parallel processing is performed at this stage in the experiment. Thus for datasets having small feature vector size, SigD2 is faster than CFAR. When the size of the feature vector is large, CFAR wins by a significant amount despite the overhead of the ensemble running sequentially. In the comparison of the run time between DEAL and CFAR, CFAR requires more time than DEAL. While training CFAR, the model considers different values for RFR and choose the best one. Selecting rules using each RFR and evaluating them requires more time than the DEAL. We consider this as a trade-off to make the model more interpretable.

### 4.2.4 Performance analysis on different sizes of the subset of feature vector

In our whole experiments with CFAR we always selected a subset of the feature vector of size 30. This is because from experiments, we found, the performance of SigD2 to be affected if the feature vector size is greater than 30. In this section, we conduct an experiment for a different size for the subset of feature vector. We start the experiment by selecting subset of size 15 and in each step, we increase the size by 5 until a size of 40. In Figure 4.8, Figure 4.9, and Figure 4.10 we can see the average accuracy, average memory requirement, and average runtime of the datasets for different sizes of subset of feature vector respectively.

Figure 4.8 shows an increase of the accuracy with the increase of the size of the subset of feature vector. In the beginning, the rate of increase in accuracy is high but with the increase in the size of the subset, the rate of increase slows down. This shows that the more features we can bundle in a feature subspace for a base learner, the more CFAR can take advantage of possible feature inter-dependence. Figure 4.9 and 4.10 show the memory requirement and runtime increase with the increase of the size of the feature subspace.

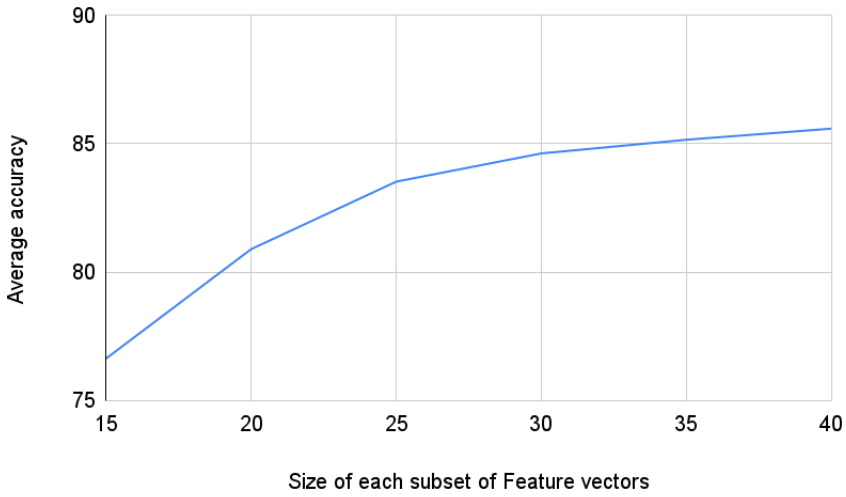


Figure 4.8: Average accuracy of 10 datasets for different size of the subset of Feature vector for CFAR

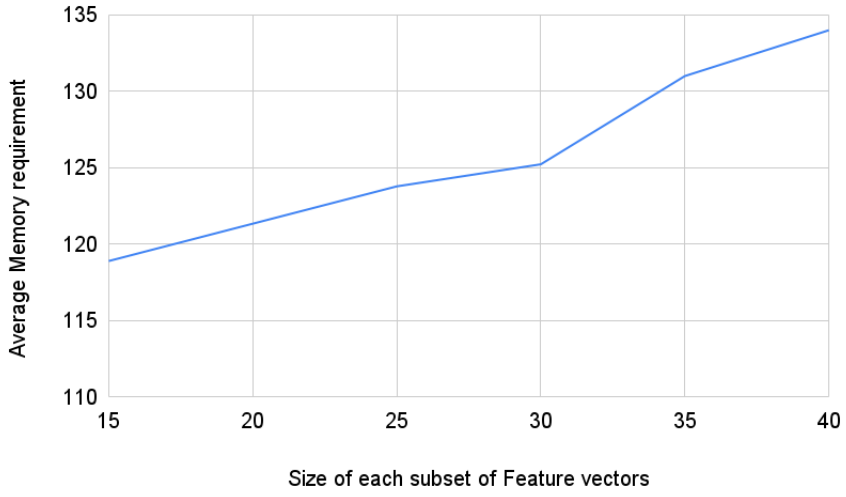


Figure 4.9: Average Memory requirement of 10 datasets for different size of subset of Feature vector for CFAR

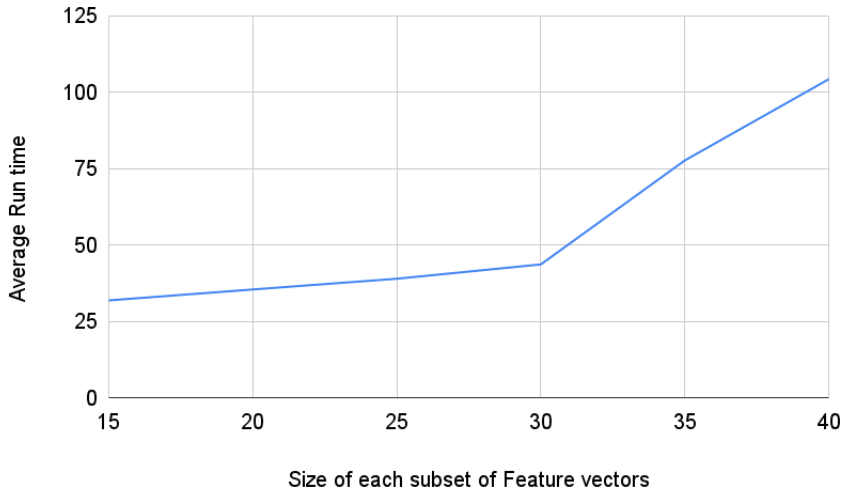


Figure 4.10: Average Runtime of 10 datasets for different size of subset of Feature vector for CFAR

Beyond 30, this increase accelerates. Therefore we conclude that 30 is a good compromise for a good accurate while considering memory requirement and runtime.

### 4.2.5 Effect of Number of base learners

In our model, we use 100 base learners. We are interested to know the effect of the number of base learners in CFAR. Thus we tested the model with a different number of base learners. In Figure 4.11 we can see the performance of CFAR for a different number of base learners. In this experiment, we consider the size of the subset of the feature vector to be 30.

From Figure 4.8, we can see when the number of base learners is reduced the average accuracy is low. With the increase of the base learners, accuracy increases. Beyond 100 base learners, the accuracy almost reaches a plateau then decreases. Considering the runtime there seems no advantage in having an ensemble larger than 100.

### 4.2.6 Interpretability and explainability

The major advantages of the associative classifier is the production of human-readable rules and by analyzing the rules it is easier to understand the decision

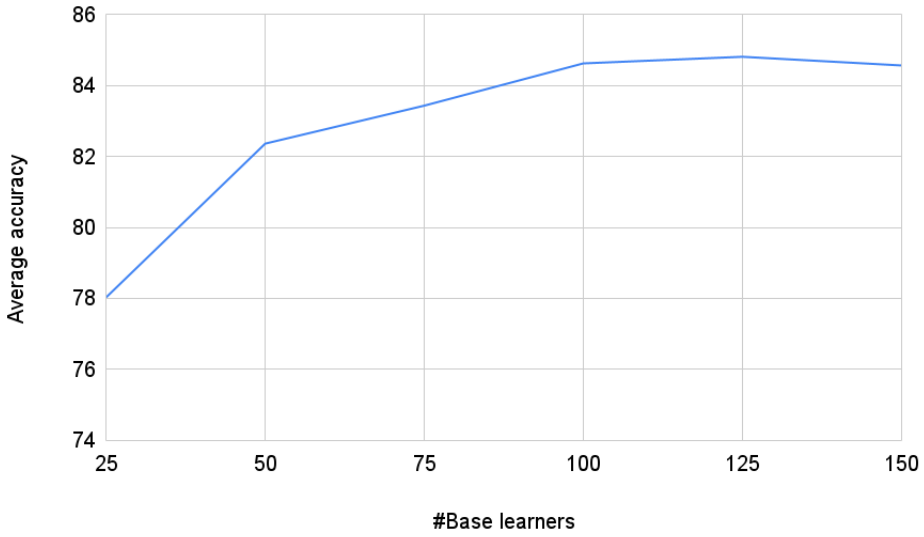


Figure 4.11: Effect of number of base learners on average accuracy in CFAR

process of the classifier. The associative classifier is therefore in itself explainable. An ensemble, however, is not straight forwardly explainable. In the classical ensemble approach, each of the base learners predicts the class label of test instances, and the final prediction is determined by a maximum vote. In the case of the ensemble of associative classifiers, to interpret the decision process we need to keep track of the base learners who voted for the final class label and then get the rules from those base learners. In this process, the number of base learners who voted for the final class label can be quite large and we need to consider the decision process of each of the base learners. Along with this, there can be some rules which have different weights in different base learners. This way of interpreting results can be difficult and time-consuming and create huge confusion.

With the CFAR approach, instead of using the max vote strategy, we collect all the rules and select rules based on their frequency. This process reduces the number of rules as well as eliminates the necessity of understanding the decision process of each of the base learners who voted for the final class label. In Table 4.6 we provide a comparative analysis of the number of total generated rules by the base learners, number of unique rules, and number of

rules selected by CFAR for the classification process.

| Stage | #Rules |
|---|---|
| Total generated rules | 1848 |
| Unique rules | 569 |
| Selected Rules | 13 |

Table 4.6: Number of rules at each stage of the CFAR

From Table 4.6 we can see, for each of the datasets, the total number of generated rules is very high compared to the selected rules. For example in Horse dataset, the total generated rules are 1848. But in the case of CFAR, only 13 rules are selected for the final prediction. Thus to understand decision process of the classifier we need to analyze only 13 rules. In Table 4.7, we provide the rules selected by CFAR for the final class prediction from the generated rules with their frequency and Relative Frequency Ratio (RFR). From Table 4.7 we can see it is very convenient to understand the decision process of CFAR.

| Selected Rules | Frequency | RFR |
|---|---|---|
| $0 \rightarrow 0$;(0.5277,0.873,-54.379) | 40 | 1 |
| $63 \rightarrow 1$;(0.0340,0.889,-6.068) | 39 | 0.975 |
| $49 \rightarrow 0$;(0.0894,0.840,-4.528) | 33 | 0.825 |
| $1 \rightarrow 1$;(0.3064,0.791,-57.129) | 31 | 0.775 |
| $64 \rightarrow 0$;(0.1447,0.944,-13.211) | 26 | 0.65 |
| $55 \rightarrow 0$;(0.1191,0.903,-8.601) | 25 | 0.625 |
| $60 \rightarrow 0$;(0.1915,0.738,-4.093) | 25 | 0.625 |
| $15 \rightarrow 0$;(0.4000,0.718,-8.093) | 25 | 0.625 |
| $48 \rightarrow 0$;(0.1957,0.821,-8.635) | 24 | 0.6 |
| $54 \rightarrow 0$;(0.1064,0.833,-5.093) | 24 | 0.6 |
| $78 \rightarrow 0$;(0.1447,0.919,-11.476) | 24 | 0.6 |
| $62 \rightarrow 1$;(0.0681,0.800,-9.304) | 24 | 0.6 |
| $12 \rightarrow 1$;(0.0936,0.595,-5.567) | 24 | 0.6 |

Table 4.7: Selected rules for final class prediction by CFAR with their count and RFR. Each rule is in the form of "Antecedent $\rightarrow$ Class label;(support, confidence, -ln(p-value))" where Antecedent is a conjunction of tokenised features. For interpretability, tokens are mapped back to features (attribute-value pairs)

# Chapter 5

# Deep Associative Classifier

## 5.1 Methodology

This section introduces the proposed model, the Deep Associative Classifier (DAC). To develop our model we followed the deep neural network architecture. In case of deep neural networks, the input data are processed at multiple layers which might be a reason for their good performance. Gcforest proposed by Zhou et al. [40] utilizes the layered architecture of deep neural networks and builds an ensemble using a decision tree as a base learner. While our architecture is different, we borrow the idea of the ensemble but use the associative classifier SigD2 as the base learner. In SigD2, an additional rule pruning technique is used to prune the noisy rules which enhances the performance of the model while significantly reducing the number of classification rules in the model. We use different ensembles, one in each layer of our architecture and another one at the start while scrutinizing the feature space. Building our layered architecture follows different steps.

In the first step, our architecture performs a scanning through the feature vectors of the dataset, this is the scanning phase of our architecture. In this phase, a fixed-size window slides over the input feature vector and selects a pre-specified number of feature vectors sequentially. In the work of Zhou et al. [40], they used 3 windows at the same time. We experimented with a different number of windows and found that only one window is sufficient for the model. This is discussed in detail in the experimental result section. As we are taking the subset of the feature vector for each base learner, our reasoning

behind adding the class probabilities created by the base learners is that they would serve as a heuristic for the base learners of the following layer to more accurately predict the class label. In the scanning phase, subsets of the feature vectors are formed and with each subset, a base learner SigD2 is trained. Each base learner provides class probabilities for the instances and the class probabilities are concatenated to the original input vector. For example, if we have a dataset with a feature vector size of 100 and the dataset consists of the instances of binary classes, with a sliding window size of 30, we will have, 71 subsets of the dataset. Each of the SigD2 base learners provides class probabilities at the end of the learning. That is as the data is of binary class, each base learner generates two-class probabilities - one for each class. Thus, with 71 base learners, we have a total of 142 class probabilities. These class probabilities are then concatenated with the original features resulting in 242 features at the end of the scanning phase. The newly formed dataset with the concatenated features is provided as an input to the next phase, the cascading phase. Algorithm 6 shows the steps performed in the scanning phase. Figure 5.1 illustrates the scanning phase of our architecture.

**Algorithm 6** Algorithm **Scanning phase**

**Input: Features:** All features in dataset
**Output:** Features concatenated with class probabilities

```
1: WindowSize=30
2: NewFeatures=[]
3: n=0
4: while n+WindowSize ≤ features.length do
5:     features_to_train ← Features[n:n+WindowSize]
6:     NewFeatures.append(Sigd2.classProbabilities(
       features_to_train)
7:     n=n+1
8: end while
9: Features ← Features.concatenate(NewFeatures)
10: return Features
```

The next step of our model is the cascade phase. Like the scanning phase, we again use Sigd2 as the base learner. In this step, the feature vector gener-

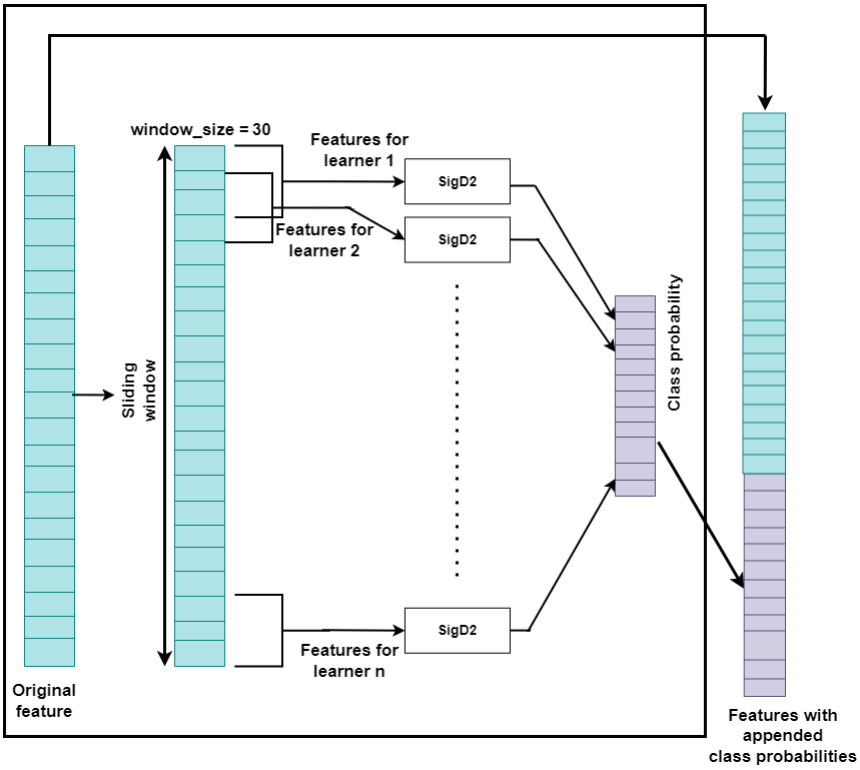


Figure 5.1: Scanning phase of Deep associative classifier

ated in the scanning phase acts as an input. The features are processed layer by layer. At each layer, we use 100 SigD2 base learners. For each base learner, 30 features are selected randomly with shuffle and replacement from the input feature vector. For the scanning phase and the cascading phase, we selected 30 features because if the feature vector size is very large, the memory requirement and run time of SigD2 increases, that is the effectiveness and efficiency of SigD2 tend to decrease with a dimension larger than 30 in terms of memory requirement and run time. Thus we decided to run the experiment with 30 features, a hyper-parameter in our model. After training, each of the base learners calculates the class probabilities. For example, if we consider our previously mentioned example of a dataset with binary classes, each of the base learners produces two class probability. With 100 base learners, we have 200 class probabilities which we append to the features after the filtering process and weight assignment. In the cascading phase, we also predict the class label of the test dataset with each of the base learners to evaluate the accuracy at

the current layer. The goal is to have an increase in accuracy from one layer to the other. We perform a max vote strategy to find the final class label of the instances and calculate the accuracy. Figure 5.2 shows the architecture of a single layer in the model, in fact, the very first layer of the cascading phase. The difference is that for the first layer the input feature vector comes from the scanning phase while for any other layer its input feature vector is the output vector of the previous layer. For example, at the end of the first layer, new class probabilities generated by the base learners of the first layer are appended with the input feature vector of the first layer. This updated feature vector is the input for the second layer. The output feature vector of the second layer acts as an input to the third layer and so on. Figure 5.3 shows the input feature vector and output feature vector at each layer.

Before forwarding the data to the next layer, we append the class probabilities from the base learners as a feature, similar to the scanning phase. But before appending, we introduce a filter to assign weights to the repetitive class probabilities. From our analysis, we found there are many base learners which produce the same class probability for each of the classes for each of the instances. Adding them directly to the output vector fed to the next layers produces more identical base learners and reduces the diversity in the ensemble. Thus instead of appending the identical class probabilities, we append the matched class probabilities multiplied with a weight factor. The weight factor is simply the count of learners generating the same class probabilities. In other words, If there is a set of identical class probabilities repeated $n$ times, we only append the class probabilities multiplied by $n$ to the output vector, making the feature vector increase with a smaller extension. While comparing the class probabilities, we compare them up-to 7 digits precision. Algorithm 7 shows the ensemble procedure and Algorithm 8 reveals the filtering procedure for the new features.

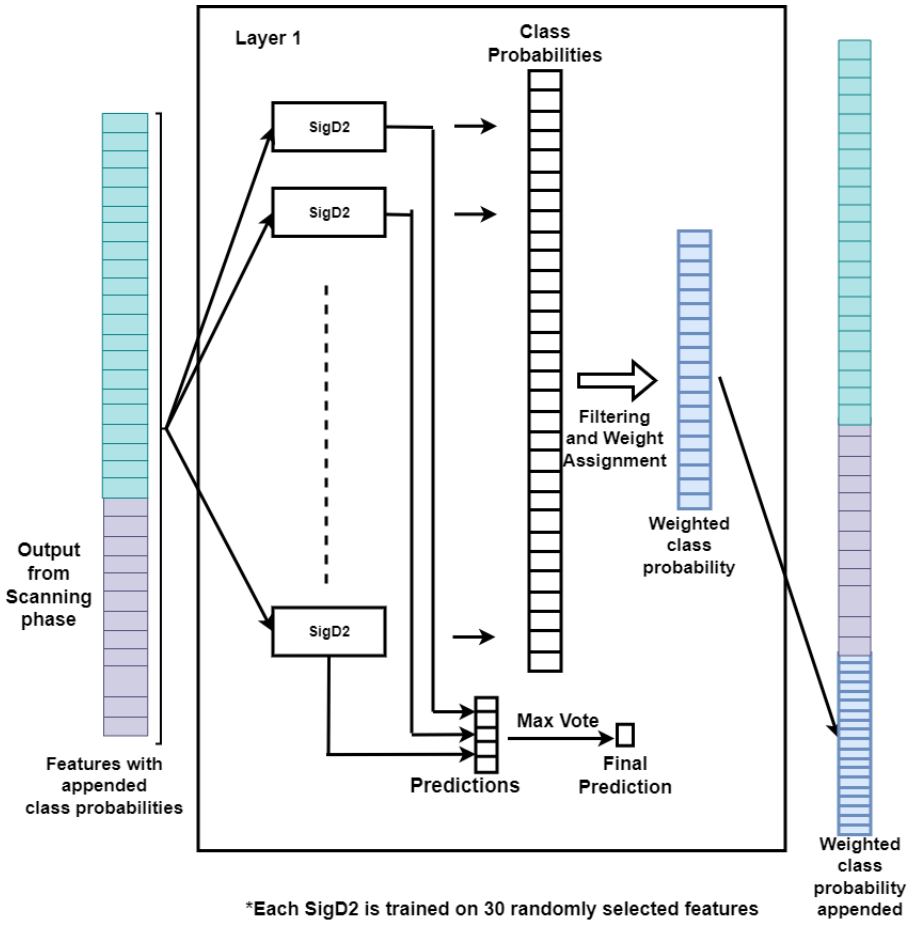


Figure 5.2: Architecture of one single layer in the cascade phase

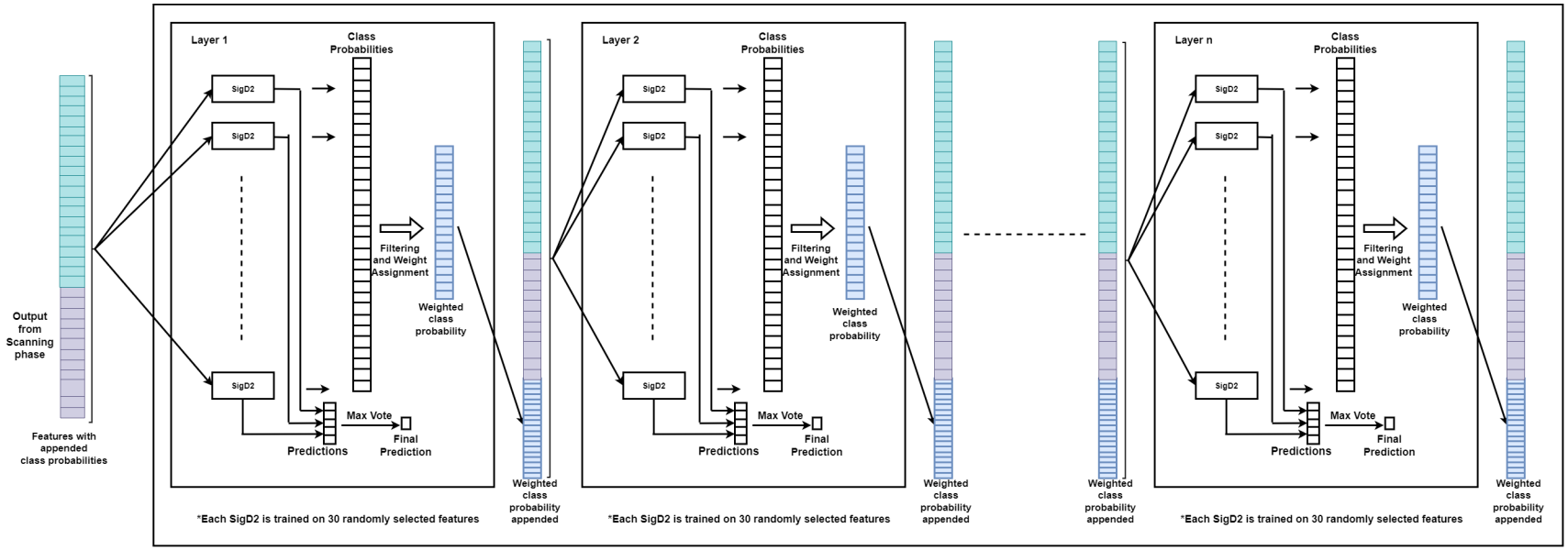


Figure 5.3: Cascade phase of Deep associative classifier

**Algorithm 7** Algorithm **Cascade**

**Input: Features**: Concatenated features from scanning phase:
**Output:** Accuracy, Concatenated features for next layer

```
NewFeatures=[]
prediction=[]
for i in range 1 to 100 do
    subsample → randomly select 30 features from Features
    Sigd2.train(train_data)
    predict ← Sigd2.predict(Test_data)
    prediction.append(predict)
    NewFeatures.append(Sigd2.classProbabilities(subsample)
end for
finalprediction ← max_vote(predictions)
accuracycalculate_accuracy(test_label, finalprediction)
if (accuracy > previous_layer_accuracy) do
    NewFeatures← weightedNewFeature(NewFeatures)
    Features ← Features.concatenate(NewFeatures)
else
    accuracy ← previouslayer_accuracy
end if
return Features, accuracy
```

**Algorithm 8** Algorithm **weightedNewFeature**

**Input: NewFeatures:** generated class probabilities
**Output: weightedfeature** weighted class probabilities as features

```
unique_features ← unique value set from NewFeatures
for unique in unique_features do:
    count ← unique in NewFeatures
    weightedfeature.append(unique * count)
end for
return weightedfeature
```

The number of layers in our architecture is not preset but determined automatically. At each layer, a prediction on test data is performed. The goal is to have an improvement of the prediction accuracy from one layer to the other. If no improvement is noted at a given layer compared to the previous layer, the creation of a new layer is halted and the layer with the best accuracy is considered the final layer. Since each layer is an ensemble of SigD2 classifiers,

the prediction of a layer is a max voting among base learners. This prediction is not appended in the output vector but its sole purpose is to decide whether to continue creating another layer if the accuracy improves or to halt and ignore the last layer if no improvement is noted. Because we cease adding new layers when there is no progress in the accuracy, the model will converge to a final output at the end of a particular layer. No improvement means that we cease adding additional layers even if the accuracy is the same as the layer before. This prevents the model from running indefinitely. The highest level of accuracy we can get is 1 if performance keeps improving in additional layers. As a result, the model will eventually converge to the desired result. Figure 5.3 shows the architecture of the cascading phase of the model and the whole architecture is provided in Figure 5.4.

## 5.2 Experimental Results

In this section, we provide an in-depth analysis of our experimental findings.

### 5.2.1 Dataset and Performance Measure

We use 10 different datasets from the UCI repository [6] to evaluate the performance of our proposed model. Before applying the algorithms to the datasets, we discretize the numerical attributes of the dataset as stated in [13]. We then vectorize the features. In the result, the reported feature vector size is the size of the feature after discretization and vectorization. For all the experiments, we use the same discretized values so that the performance is measured across the same format of the dataset. We select these datasets so that we can evaluate the performance of our model on different types of datasets i.e., datasets having a different number of records and a different number of features. By evaluating the performance of our proposed model on these datasets, we try to ensure that our proposed approach works well across datasets with different sizes in records and features.

### 5.2.2 Results

Our proposed model is assessed in terms of accuracy and memory requirement. For this comparison, we select random forest, gcForest proposed by Zhou et al. [40], SigD2 proposed by Sood and Zaiane [34] and three different deep neural network architectures. From the rule-based classifier, we select only SigD2 because it was shown to outperform other existing rule-based classifiers. We also calculate the memory requirements of the model and show a comparative analysis. Finally, we conduct a statistical analysis on the accuracy to show the significance of our results. In our experiments, we divide the dataset into training and testing data, 80% and 20% respectively. For all datasets, we use a window size of 30. In each layer of the cascade phase, we use 100 SigD2 base learners. In the case of gcForest, we use the implementation [26] of gcForest proposed by Zhou et al. [40]. As we use discretized datasets, the reported results can be slightly different from the results reported in the mentioned papers.

Among the deep neural network models, The first deep neural network (DNN1) has one hidden layer and the second deep neural network (DNN2) has two hidden layers and DNN_N where we incorporated N hidden layers. In the case of DNN_N, N represents the number of layers equal to the number of layers determined by our proposed model DAC for each dataset. DNN1 consists of one hidden layer with 256 nodes in the layer with a dropout of 0.3 and ReLU activation function. In the output layer, we use the softmax activation function. We use the Adam optimizer with a learning rate of .0001. In DNN2 and DNN_N we use the same parameters with additional hidden layers with 256 nodes. With DNN1 in DNN2, one more hidden layer is added, and with DNN1 in DNN_N, N-1 more hidden layers are added. For each dataset, we utilise a different number of layers in the DNN_N model, which is dataset-specific and the number of layers are determined by our proposed model. We execute the model for 100 epochs for each dataset. We take these models as a general representation of the deep neural network family.

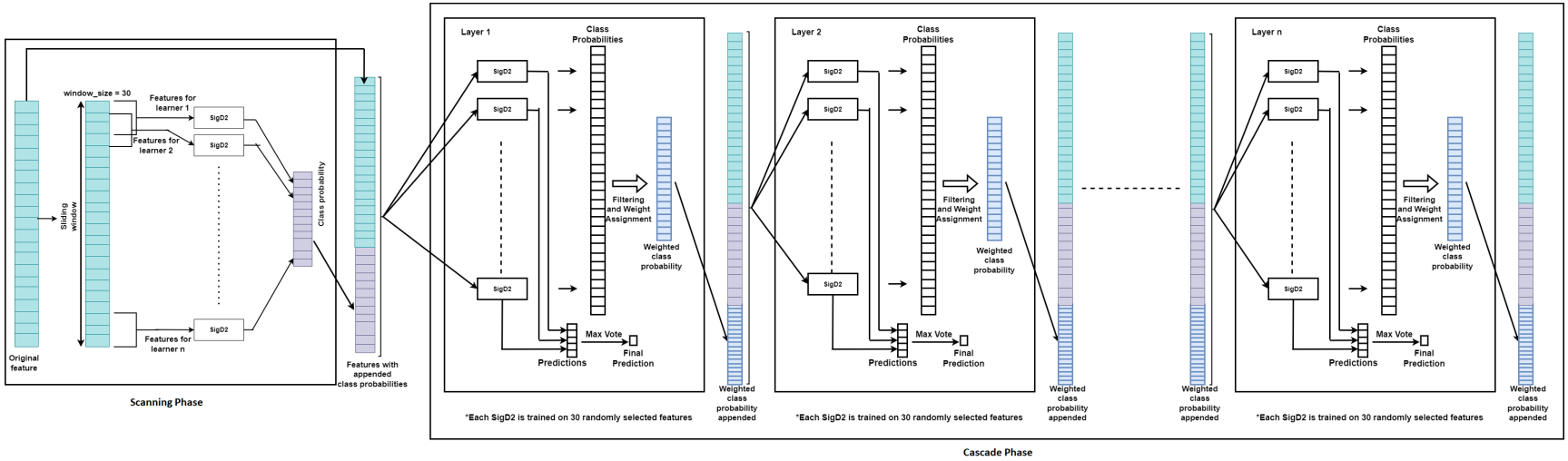


Figure 5.4: Architecture of Deep associative Classifier

With different combination of hyper-parameters for the deep neural network architectures, the performance may outperform the classifiers mentioned in this paper including our proposed model. But the hyper-parameter tuning might be dataset-specific and it is not an easy task to find proper set of hyper-parameters. We consider this as a limitation of the deep neural networks. For this reason, we do not perform hyper-parameter tuning for each dataset for deep neural networks rather use simple architectures for all the datasets.

Among the deep neural network models, The first deep neural network (DNN1) has one hidden layer and the second deep neural network (DNN2) has two hidden layers and DNN_N where we incorporated N hidden layers. In the case of DNN_N, N represents the number of layers equal to the number of layers determined by our proposed model DAC for each dataset. DNN1 consists of one hidden layer with 256 nodes in the layer with a dropout of 0.3 and ReLU activation function. In the output layer, we use the softmax activation function. We use the Adam optimizer with a learning rate of .0001. In DNN2 and DNN_N we use the same parameters with additional hidden layers with 256 nodes. With DNN1 in DNN2, one more hidden layer is added, and with DNN1 in DNN_N, N-1 more hidden layers are added. For each dataset, we utilise a different number of layers in the DNN_N model, which is dataset-specific and the number of layers are determined by our proposed model. We execute the model for 100 epochs for each dataset. We take these models as a general representation of the deep neural network family. With different combination of hyper-parameters for the deep neural network architectures, the performance may outperform the classifiers mentioned in this paper including our proposed model. But the hyper-parameter tuning might be dataset-specific and it is not an easy task to find proper set of hyper-parameters. We consider this as a limitation of the deep neural networks. For this reason, we do not perform hyper-parameter tuning for each dataset for deep neural networks rather use simple architectures for all the datasets.

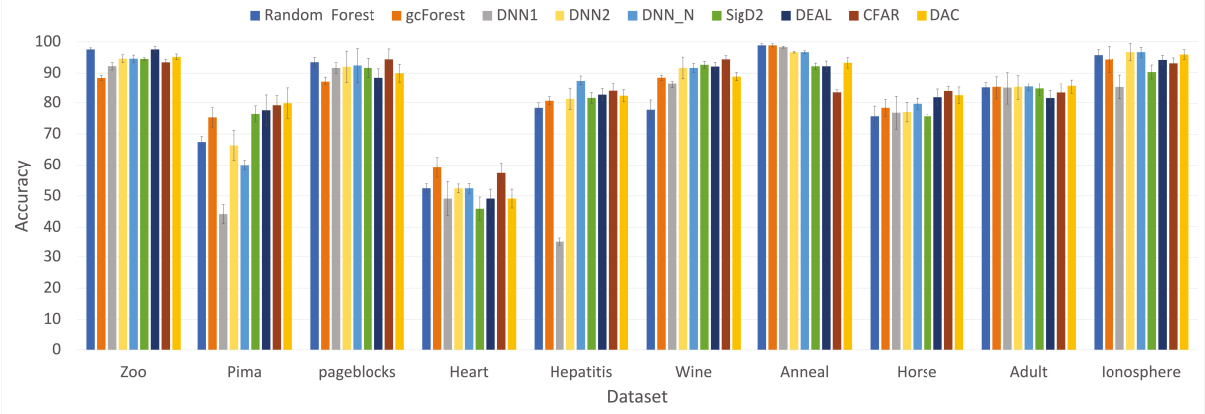


Figure 5.5: Comparison in Accuracy

### 5.2.3 Accuracy

A comparison of the accuracy of our model compared to other models is given in the Table 5.1 and Figure 5.5

From Table 5.5 we can see in 2 datasets DAC outperforms all other models. Only in 1 datasets, dataset specific model DNN_N outperforms all other models and in 1 dataset DNN2 and DNN_N has same performance and they outperform all other models. For the Hepatitis dataset, the accuracy of the associative classifier and the gcForest model is quite low as compared to the dataset specific model DNN_N. In this dataset, the decision tree-based architecture gcForest, rule-based classifier SigD2 and our proposed model could not perform well. The core architecture of the deep neural network models might have an advantage over the rule-based models and the gcForest model for this dataset. In the case of the Anneal dataset, we can see that gcForest architecture outperforms all other models. In this dataset, the Random forest also has the same performance as the gcForest. As the base learner for both models is the decision tree, in the case of this dataset, the method of the decision tree has an advantage over the other classifiers. In 3 datasets, CFAR outperforms other classifiers. In 4 datasets DAC outperforms all DNN models and in 8 datasets DAC outperforms its base classifier SigD2. DAC also has better average accuracy than the other models except CFAR. We also

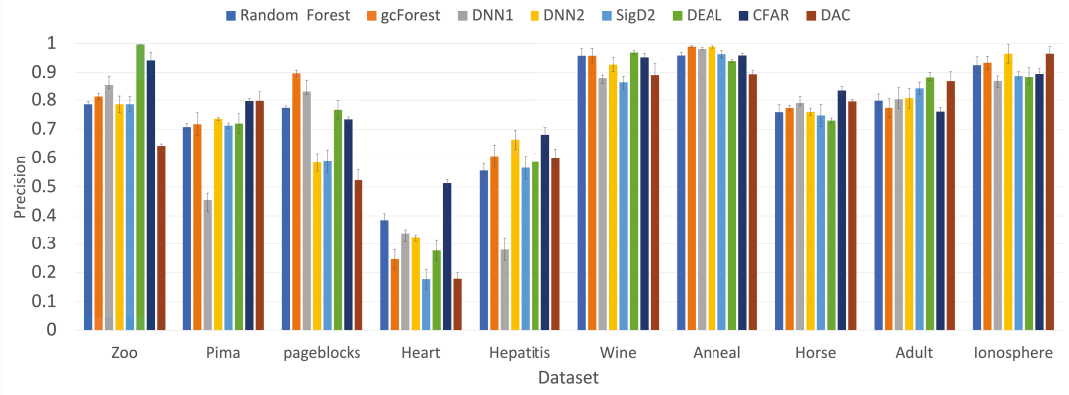


Figure 5.6: Comparison of Precision

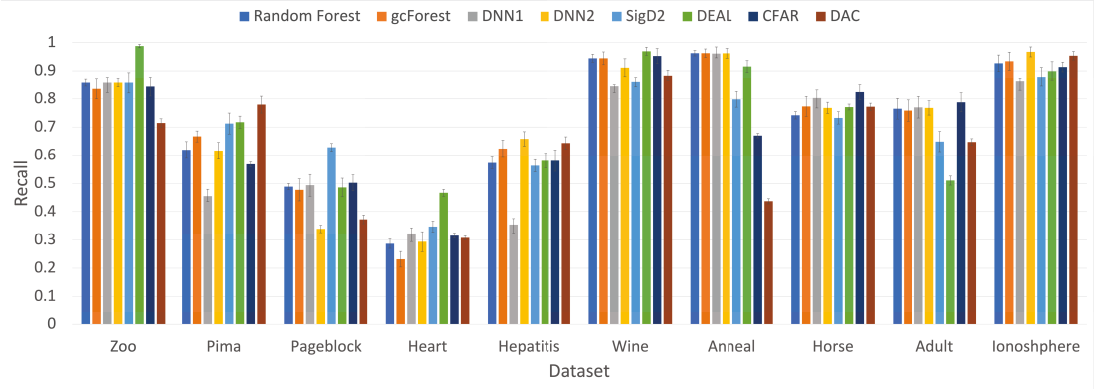


Figure 5.7: Comparison of Recall

compare the performance of DAC with other models in terms of precision and recall. Figure 5.6 and 5.7 shows the precision and recall of the models for each dataset respectively. We can see DAC has very competitive precision and recall compared to other models.

To see the effect of filtering and weighting of the augmented feature, we tested the performance of our model by removing the filtering phase. We only generated the class probabilities and without further analysis, we added all the class probabilities in the dataset as features. Similarly, we also tested the performance of our model without the scanning phase. In this case, we used the same parameters and settings as mentioned above but just removed the scanning phase. The results of these tests are provided in Figure 5.8.

From Figure 5.8, we can see that removing the filtering or the weighting

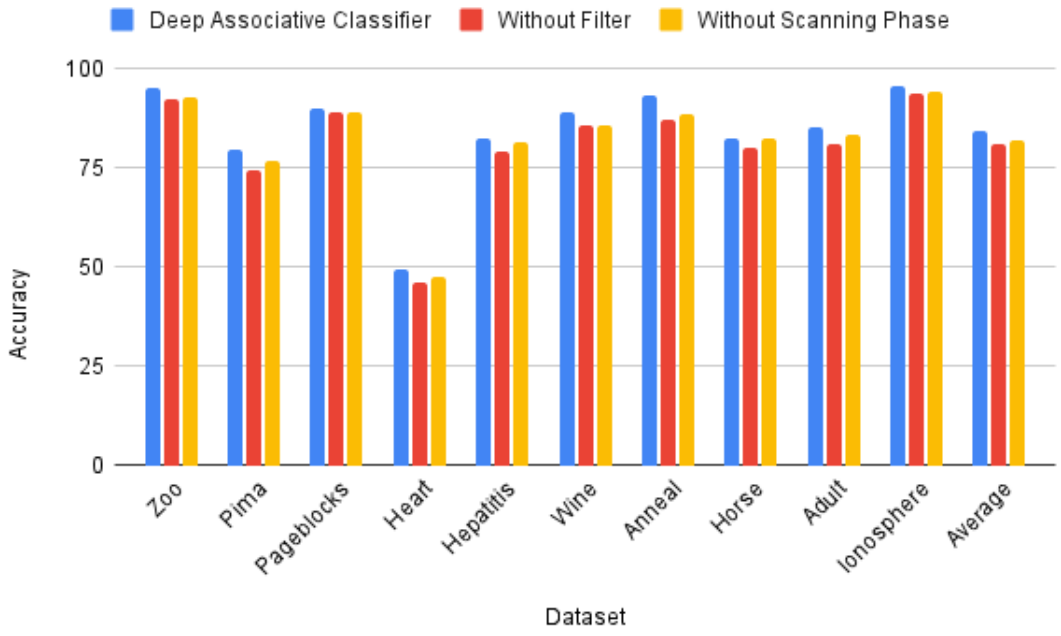


Figure 5.8: Comparison of accuracy with and without different component of the proposed model

of the augmented feature components from the model has an impact on the performance of the model. Among the two modules, filtering has more impact on the results. In all the datasets removing the filtering step reduces the accuracy more than removing the scanning step. In the case of implementing the model without filtering, when the model generates class probabilities, we found many base learners produce the same class probabilities for all the instances. To make a good ensemble we know diversity among the base learners is a crucial factor. But without filtering many of the base learners produce duplicate results. With those results, in the subsequent layers, they produce even more duplicate base learners which hampers the model.

We analysed the performance of our model when using a different number of windows in the initial scanning step scrutinising the feature space. We experimented with two, three and four windows. The average accuracy for the different windows is provided in Figure 5.9.

In Figure 5.9, we can see, that increasing the number of windows in the

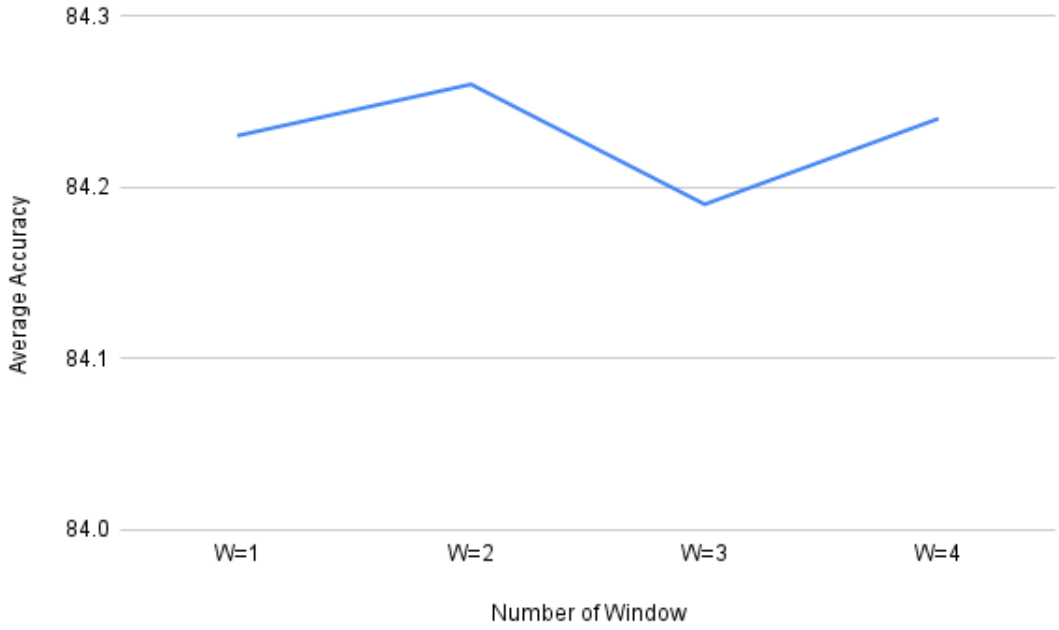


Figure 5.9: Effect of number of windows in the scanning phase

scanning does not have a significant impact on the performance of the model in terms of accuracy. However, it impacts the execution time of the model. In our model, the layers and the base learners are trained sequentially. Thus increasing the number of windows increases the execution time of the model.

We also analyzed the different window sizes by experimenting with window sizes from 15 to 40 increasing by 5 at each step. We plot in Figure 5.10 the average accuracy of the 10 datasets with the increase of the window size. From the figure we can see that the accuracy increases with the increase in the window size. However, there is a compromise to make. As the window size increases, so does the feature vector output in each layer and therefore the memory requirement.

Finally, we analyse the effect of the number of base learners in each cascading layer. Since we use a majority voting scheme, the number of base learners can have a big impact on the performance of the model. This is a very important hyper-parameter for our model. In the case of the number of base learners, we perform the test varying the number of base learners from 25 to

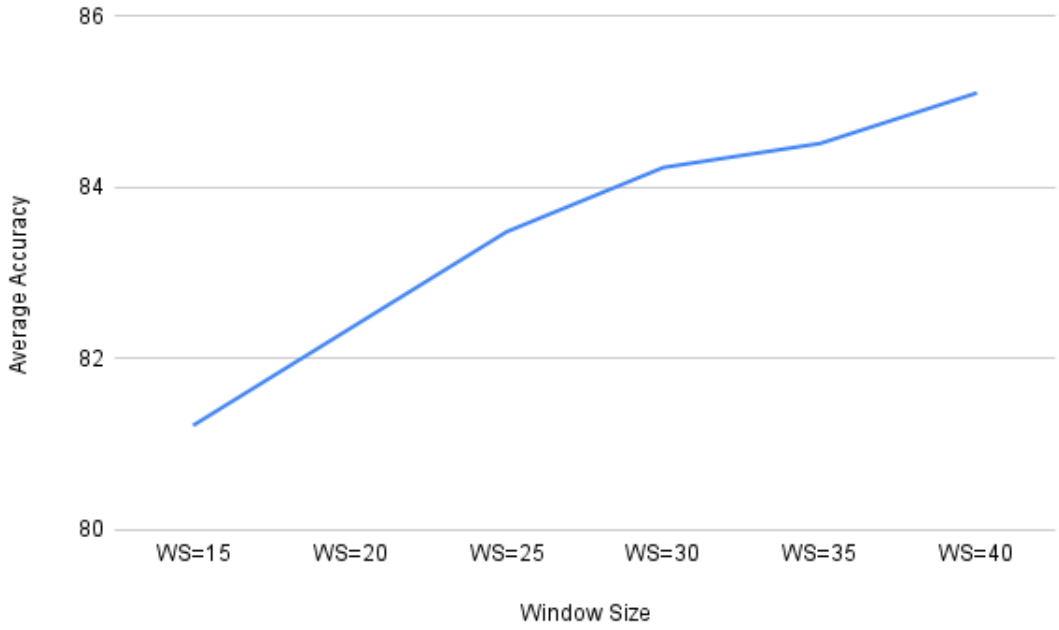


Figure 5.10: Effect of different window size on accuracy

200 increasing by 25 base learners in each step. Figure 5.11 shows the effect of the number of base learners on accuracy.

From Figure 5.11, we can see with the increase in the number of base learners, the accuracy increases sharply then plateaus beyond 100. In our model, we introduce filtering at each layer of the cascading layer which keeps only one of the base learners and increases redundancy by assigning some weight to that one learner. Thus when we increase the number of base learners, many redundant base learners are produced and most of them are filtered out. By increasing the number of base learners beyond 100 there is no significant change in the accuracy of the model.

### 5.2.4 Memory Requirement

To see how efficient our proposed model is in terms of memory requirement, we also measure the memory needed for each model. Deep neural networks require a high amount of memory for processing data at each layer. SigD2 also

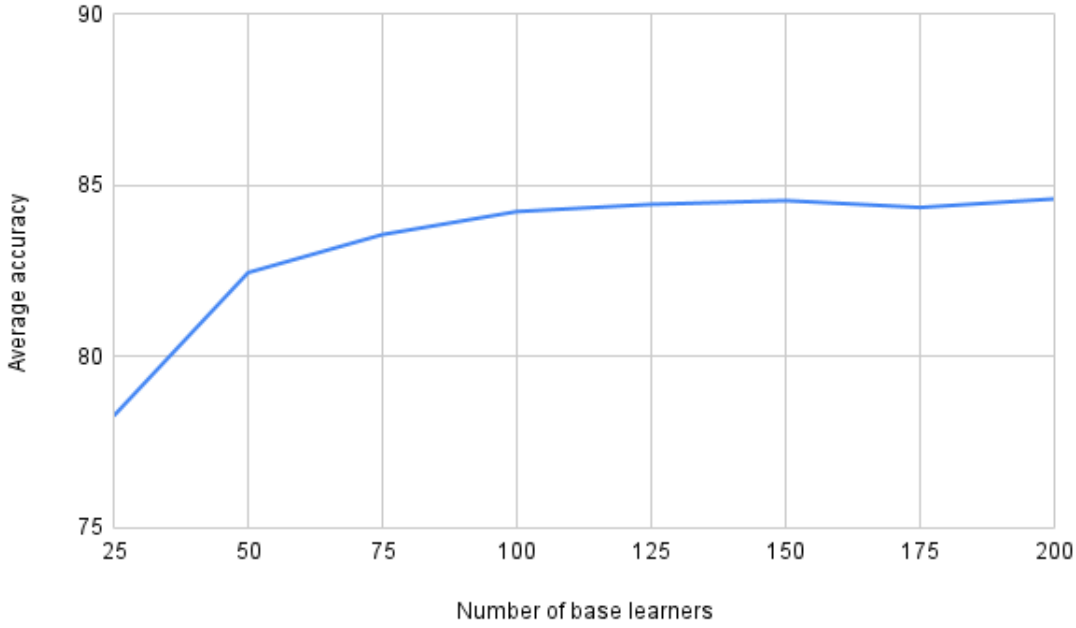


Figure 5.11: Effect of number of base learners on accuracy

requires a high amount of memory with the increase of the feature vector size. We can see the memory requirements of the different models in Table 5.2. To calculate the memory requirement for each of the model we used the python library *psutil*.

From the table, we can observe the high memory requirement of deep neural networks. The memory requirement is also high for SigD2 for a dataset with a large feature vector. The gcForest architecture largely decreases the memory requirements for all types of datasets. Our proposed model further decreases the memory requirements. In the case of a dataset having a very small number of features, SigD2 requires less memory than our proposed model. But when the feature vector size increases, we can see a decrease in the memory requirement by our model in contrast to SigD2. Our model also shows competitive performance in comparison with random forest.

To get a better understanding of the memory requirement of our model, we tried different window sizes since SigD2 is sensitive to the size of feature vectors. Thus we tested different window sizes from 15 to 40 increasing by

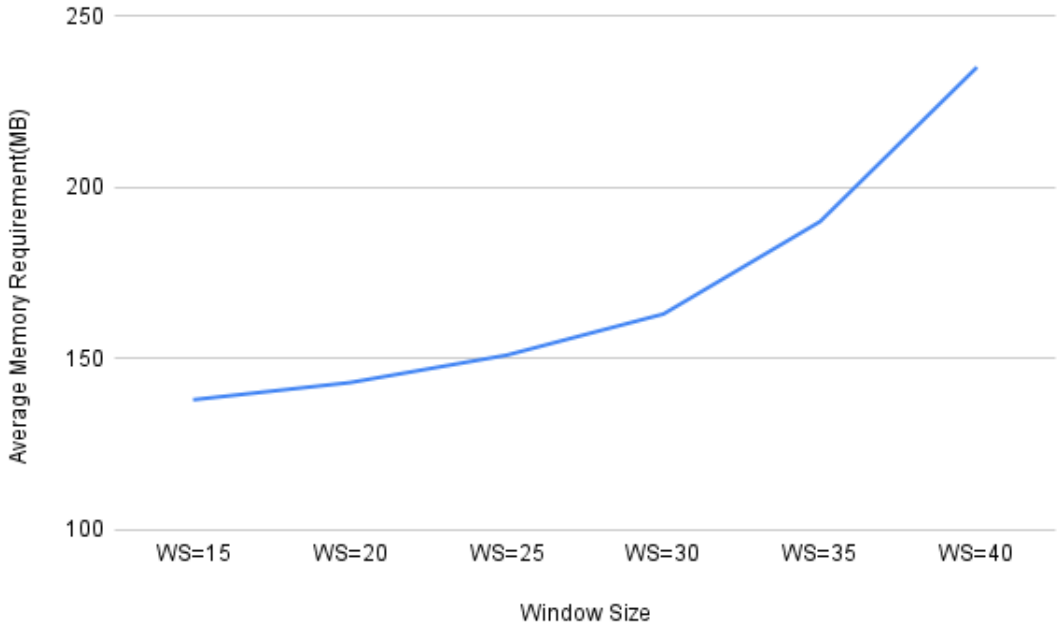


Figure 5.12: Effect of window size in the memory requirement of DAC

5 at each step. Figure 5.12 shows the average memory requirement by our proposed model as we increase the window size. By analyzing Figure 5.10 and Figure 5.12, we can see the improvement in the accuracy and higher memory requirement with the increased window size. After window size 30 though the accuracy increases but the memory requirement also rises and the trend of increasing memory requirement is much higher than the accuracy. For this reason, window size 30 can be considered as an optimum window size for using our proposed model.

### 5.2.5 Number of hyper-parameter and tuning feasibility

In previous sections, we mentioned, that deep neural network models have a complicated hyper-parameter tuning process and they differ for different datasets, In our experiment, we used a simple version of the deep learning model and we used the same model for all the datasets, these models of deep neural networks could not perform well in most of the datasets. But with proper hyper-parameters, deep neural networks might perform better. Most

of the time the hyper-parameter tuning for deep neural networks is dataset-specific. That is, for each of the datasets the hyper-parameters could be different. Tuning the hyper-parameters is a tedious procedure. Moreover, in the case of deep neural network models, it is very hard to understand the effect of each hyper-parameter on the performance of the model.

On the other hand, gcForest architecture requires fewer hyper-parameters than the deep neural network models. Still, several hyper-parameters are present in their model. For the experiment, the authors found a setting where they showed their model achieved a better performance on different types of datasets, There can be some other settings of the hyper-parameter by which the performance of their model can be further improved. In their case, tuning 4 hyper-parameters to find a suitable setting is also a complicated and time-consuming process and like deep neural networks, there is no way to assume the hyper-parameters without testing. In our model, we further reduced the number of hyper-parameters. We have two hyper-parameters which are the number of base learners in each of the cascading layers, and the window size. The number of layers is determined automatically. With experiments, we found a window size 30 is a very good choice for our model as decreasing from 30 decreases the accuracy of the model. Increasing the window size highly increases the memory requirement which we can see from Figure 5.10 and Figure 5.12. Still, for some other datasets, different window sizes may improve the performance of the model. Table 5.3 shows the number of hyper-parameter in the case of the deep neural network models, gcForest and our proposed model.

Considering the complexity of tuning hyper-parameters for deep neural networks and gcForest, changing the hyper-parameters for our proposed model is quite an easy task. In the case of our model, if we increase the number of base learners at each layer, the effect of the change that is whether there is an improvement in the performance can be directly observed. In the case of changing the window size, the change in the accuracy and memory requirements is also directly discernable from the result. Decreasing the window size makes the model faster but decreases the accuracy, and increasing the window

size requires much memory while still improving the accuracy. By changing the window size it is easy to determine a good window size for a dataset which is an easier task than testing the model with different settings of many hyper-parameters in the case of deep neural networks and gcForest.

From Table 5.3, we can see that gcForest requires 4 different parameters to be tuned. Finding a proper settings by tuning 4 different parameter is not an easy task as different combination can work for different dataset. But in our model,if we consider the window size is fixed then, we have one hyper-parameter which is the number of base learners in each of the cascading layers. Tuning this one parameter is very easy to increase or decrease the number of base learners in each of the layers. In case we want to know how our model performs in case of a different number of base learners, changing only the number of base learners is sufficient and we can directly observe the effect of changing the number of base learners from the result. Side by side, if we consider the window size as a hyper-parameter,finding a combination of two hyper-parameter will not be a difficult task. Changing both of the hyper-parameter, the effect of the change is directly observable. Thus having these two parameter makes our model more suitable for tabular dataset.

### 5.2.6 Statistical Analysis

Finally, we conducted a statistical analysis to show the significance of the improvements in our proposed model. For this, we performed the student paired t-test with the null hypothesis, that the improvement in the accuracy of our model is not significant. We run the whole experiment 20 times and calculated the average accuracy for each model. In this case, we used a window size of 30 and 100 base learners in each layer of cascading phase. Then we calculated the differences in accuracy by $\delta = \mathcal{M}_{DAC}(test\_data) - \mathcal{M}_i(test\_data)$. Here $\mathcal{M}_{DAC}$ is our proposed model and the subscript i stands for the other models. Running the models on test data provides the accuracy of the models and $\delta$ is the difference in the accuracy. Then with the differences, we performed the paired t-test which is inspired by the work of Henry et al.**15**. We used their model to calculate p-values for each of the mentioned classification models

against our proposed model. The calculated p-values are provided in Table 5.4.

| Algorithm | p-value |
|---|---|
| DAC vs Random Forest | 0.0235 |
| DAC vs gcForest | 0.0030 |
| DAC vs SigD2 | 0.0114 |
| DAC vs DNN1 | 3.2084e-10 |
| DAC vs DNN2 | 0.0209 |
| DAC vs DNN_N | 0.0268 |

Table 5.4: Statistical result

From Table 5.4 we can see the p-value for each pair is very low. Thus we can reject the null hypothesis and conclude that the improvement in accuracy of our proposed model is significant.

| Dataset | #cls | #rec | Feature vector size | Random Forest | gcForest | DNN1 | DNN2 | DNN_N | SigD2 | DEAL | CFAR | DAC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Zoo | 7 | 101 | 35 | **97.62** | 88.46 | 92.24 | 94.64 | 94.64 | 94.55 | **97.62** | 93.47 | 95.23 |
| Pima | 2 | 768 | 36 | 67.41 | 75.32 | 44.21 | 66.32 | 60 | 76.44 | 77.61 | 79.22 | **79.87** |
| pageblocks | 5 | 5473 | 41 | 93.51 | 87.28 | 91.67 | 92.05 | 92.45 | 91.67 | 88.53 | **94.42** | 89.95 |
| Heart | 5 | 303 | 47 | 52.45 | **59.21** | 49.18 | 52.45 | 52.45 | 45.9 | 49.18 | 57.38 | 49.18 |
| Hepatitis | 2 | 155 | 54 | 78.38 | 80.65 | 35.42 | 81.25 | **87.50** | 81.54 | 82.58 | 83.87 | 82.26 |
| Wine | 3 | 178 | 65 | 77.78 | 88.55 | 86.11 | 91.67 | 91.67 | 92.7 | 92.13 | **94.44** | 88.89 |
| Anneal | 6 | 898 | 67 | **98.89** | **98.89** | 98.33 | 96.74 | 96.74 | 92.22 | 92.2 | 83.33 | 93.33 |
| Horse | 2 | 368 | 83 | 75.67 | 78.38 | 76.82 | 77.03 | 79.7 | 75.68 | 81.79 | **83.78** | 82.43 |
| Adult | 2 | 48842 | 95 | 84.97 | 85.12 | 84.86 | 85.13 | 85.25 | 84.59 | 81.51 | 83.31 | **85.41** |
| Ionosphere | 2 | 336 | 155 | 95.78 | 94.37 | 85.34 | **96.55** | **96.55** | 90.14 | 94.04 | 92.96 | 95.78 |
| Average | | | | 82.25 | 83.62 | 74.42 | 83.38 | 83.70 | 82.54 | 83.72 | **84.62** | 84.23 |

Table 5.1: Accuracy of proposed method compared to other models

| Dataset | #cls | #rec | Feature vector size | Random Forest | gcForest | DNN1 | DNN2 | DNN_N | SigD2 | DEAL | CFAR | DAC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Zoo | 7 | 101 | 35 | 127 | 110 | 1715 | 1721 | 1721 | 121 | 105 | 106 | **102** |
| Pima | 2 | 768 | 36 | 128 | 119 | 1774 | 1792 | 660 | **101** | **101** | 105 | 105 |
| pageblocks | 5 | 5473 | 41 | 134 | 170 | 1729 | 1797 | 1797 | 112 | 110 | **107** | 279 |
| Heart | 5 | 303 | 47 | 129 | 115 | 1734 | 1742 | 1742 | 107 | **104** | 119 | 105 |
| Hepatitis | 2 | 155 | 54 | 127 | 111 | 1786 | 1807 | 688 | 113 | **100** | 108 | 104 |
| Wine | 3 | 178 | 65 | 128 | 110 | 1715 | 1763 | 1763 | 165 | **100** | 107 | 102 |
| Anneal | 6 | 898 | 67 | 129 | 117 | 1802 | 1827 | 1827 | 150 | **106** | 108 | 116 |
| Horse | 2 | 368 | 83 | 128 | 137 | 1813 | 1833 | 717 | 235 | **106** | 108 | 112 |
| Adult | 2 | 48842 | 95 | 253 | 374 | 1871 | 1876 | 1992 | 825 | **252** | 255 | 492 |
| Ionosphere | 2 | 336 | 155 | 128 | 276 | 1838 | 1859 | 1859 | 2519 | **108** | 136 | 113 |

Table 5.2: Memory requirement (in MB) of proposed method compared to other models

| Deep neural networks | gcForest | DAC |
|---|---|---|
| • Activation functions<br>• Number of hidden layer<br>• Number of Nodes in hidden layer<br>• Number Feature maps<br>• Kernel size<br>• Learning rate<br>• Dropout ratio<br>• Momentum<br>• weight regularization penalty<br>• Weight initialization<br>• Batch size | • Number of Forests<br>• Number of Trees in each forest<br>• Tree growth<br>• Sliding window sizes (3 different sizes) | • Number of base learners<br>• Window size |

Table 5.3: Hyper-parameters

# Chapter 6

# Conclusion

## 6.1 Summary

In this study, we propose Dynamic Ensemble Associative Learning (DEAL) where we use SigD2 as a base learner and present a feature sampling method which eliminates the need to fix the size of ensemble while ensuring diversity and feature space coverage. Evaluating the model over different datasets, reveals an increase of the accuracy. DEAL solves the limitation of SigD2 requiring huge memory and runtime if the feature vector space is large. Memory requirement and runtime of DEAL does not increase with the increase in feature vector like SigD2. Along with decreasing the runtime and memory requirement, we designed DEAL in such a way that the decision process of DEAL is human readable and explainable. The sampling method of DEAL provides promising results when using SigD2 as a base learner since it ensures diversity and feature space coverage.

Further, we propose another ensemble technique Classification by Frequent Association Rules (CFAR) using SigD2 as base learner where instead of the classical max voting strategy we aggregate all the rules generated by the base learners and select frequent rules for the classification task. An experiment with 10 different datasets shows CFAR increases the accuracy in most cases. CFAR also eliminates the limitation of high memory requirement and runtime of SigD2 in case of a dataset having a large feature vector size. In CFAR, instead of the classical ensemble process, we design the ensemble in such a way that the decision process remains human-readable and explainable, and can be

understood by analyzing a very small number of rules. Along with providing promising performance in terms of accuracy, memory requirement, and run time, CFAR also preserves interpretability. Understanding the decision process of CFAR is easier than any other ensemble model.

In addition to this, we propose Deep Associative Classifier(DAC) where we present an attempt at exploit one of the main reasons behind the success of deep learning: the layered and deep representation learning. Our proposal expands on the idea of the gcForest by using a rule-based classifier, SigD2, as a base learner, introduces a filtering process to tackle diversity in ensembles, and further reducing hyper-parameters. One of the main questions we try to answer while building DAC is *Can we have a deep architecture with fewer hyper-parameters, low memory requirement and exploit a layer-wise data processing architecture of deep neural networks while making use of a rule-based base learner?*. To answer this question and overcome some of the limitations of deep neural networks, gcForest, and some hindrances of associative classifiers, we designed a deep associative classifier architecture using SigD2 as a base learner. We compared the performance of our approach with the other state-of-the-art models in terms of accuracy, memory requirements and hyper-parameter tuning feasibility on 10 different UCI datasets. We explored different sampling and tuning methods and added a filtering phase along with scanning and cascading phase to the ensemble architecture to improve the accuracy of our proposed model. From our experiments, we show that our model performs better than other existing models including gcForest and deep neural networks for different datasets not only in terms of accuracy but also in terms of memory needs. We also show that our model reduces the memory requirement to a great extent in comparison to deep neural networks and SigD2 which makes our model suitable to run in low-resource hardware. Our proposed model DAC has only two hyper-parameters to tune as compared to gcForest which has four to tune and deep neural networks which have several hyper-parameters. Further, our model achieves excellent performance across various datasets in terms of accuracy.

## 6.2 Future Research Direction

- The sampling method of DEAL provides promising results when using SigD2 as a base learner since it ensures diversity and feature space coverage. Random Forest also uses a subset of the features and the subsets are chosen randomly. It would be interesting to use our sampling method in the case of Random Forest. This would eliminate the need to predetermine the number of estimators in Random Forest.

- DEAL is a framework where the base learner SigD2 could be replaced. A better rule based learner that is not as constrained by the dimensionality of the feature space could allow experimenting with a larger sampled feature subspace.

- In our proposed model CFAR, we use SigD2 as base learner. We can also use other associative classifiers as base learners in this model. It would be interesting to use other associative classifiers as base learners and compare their performance. our study comprises applying CFAR only to tabular data. In the future, we want to deploy CFAR on text and image data. In our model, CFAR is a framework where we replaced the max voting strategy with aggregating the association rules for final prediction. It would be interesting to explore the same strategy in other interpretable ensemble models.

- While selecting rules in CFAR we use a simple criteria which we call Relative Frequency Ration (RFR) which we calculate based on the frequency of appearance of a rule. It would be an interesting idea to investigate other measures to select rules.

- In DAC, in the cascading step, we employ a filtering process to ensure that identical class probabilities are not appended more than once in the output feature vector. Our current filtering strategy simply avoids repetition by multiplying the probabilities by a weight equal to the count of repetitions of identical predictions. The performance of our model

could be improved by exploring other filtering methods that still ensure diversity in the subsequent ensemble and safeguard the feature vector size in the layer output. Moreover, the random sampling of features for the base learners in the layer ensemble during the cascading step makes the model less stable across different runs. A better strategy of feature sampling that guarantees diversity and coverage could not only ensure stability but also eliminate the need to select the size of the ensemble as a hyper-parameter but instead automatically determine this size per layer.

- In DAC, the number of layers is determined automatically by the model. While determining the number of layers, we stop execution if we do not find any improvement at a layer from its previous layer. From our experiment we find in all the cases the execution stops after 3 to 4 layers which is not very deep compared to the deep learning models. It would be an interesting idea to see the performance of the model if we allow the model to add few more layers even if the accuracy does not increase because it might be the case for some dataset that the accuracy might go down and then go up again.

- While sampling the features in DEAL, CFAR and DAC, we select features with random selection. Even for dynamic sampling we select the features randomly and then compare the subsample with other subsamples. It would be worthwhile to investigate other criteria for selecting features for instance keeping the most correlated features together or the least correlated features together or other feature selection approaches..

- Another important direction for DAC to explore is to investigate how we can exploit the interpretability of the rule-based classifier, SigD2, to make DAC explainable. Understanding the important features in a decision is tricky with a cascade of layers each with an ensemble of deciders voting by majority. Moreover, the salient features composing the vectors in each layer are made up not only of the initial input features but

also collected weighted probabilities from the ensemble of the previous layers. Finding the salient features is a problem in itself but translating those back to the original features is another open problem worth investigating.

# References


[1] R. Agrawal, R. Srikant, *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, Citeseer, vol. 1215, 1994, pp. 487–499.

[2] M.-L. Antonie and O. R. Zaiane, "Text document categorization by term association," in *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, IEEE, 2002, pp. 19–26.

[3] M.-L. Antonie, O. R. Zaiane, and R. C. Holte, "Learning to use a learned model: A two-stage approach to classification," in *Sixth International Conference on Data Mining (ICDM'06)*, IEEE, 2006, pp. 33–42.

[4] A. B. Arrieta, N. Dıaz-Rodrıguez, J. Del Ser, *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information fusion*, vol. 58, pp. 82–115, 2020.

[5] B. Arunasalam and S. Chawla, "Cccs: A top-down associative classifier for imbalanced class distribution," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 517–522.

[6] A. Asuncion and D. Newman, *Uci machine learning repository*, 2007.

[7] K. R. Bandi and S. N. Srihari, "Writer demographic classification using bagging and boosting," in *Proc. 12th Int. Graphonomics Society Conference*, 2005, pp. 133–137.

[8] R. J. Bayardo Jr, "Brute-force mining of high-confidence classification rules.," in *KDD*, vol. 97, 1997, pp. 123–126.

[9] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[10] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[11] G. Brown, "Diversity in neural network ensembles," Ph.D. dissertation, Citeseer, 2004.

[12] P. Cao, D. Zhao, and O. R. Zaiane, "Cost sensitive adaptive random subspace ensemble for computer-aided nodule detection," in *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*, IEEE, 2013, pp. 173–178.

[13] F. Coenen, *The lucs–kdd software library*, 2004.

[14] W. W. Cohen, "Fast effective rule induction," in *Machine learning proceedings 1995*, Elsevier, 1995, pp. 115–123.

[15] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[16] B. V. Dasarathy and B. V. Sheela, "A composite classifier system design: Concepts and methodology," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 708–713, 1979.

[17] Y. Freund, R. E. Schapire, *et al.*, "Experiments with a new boosting algorithm," in *icml*, Citeseer, vol. 96, 1996, pp. 148–156.

[18] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[19] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural network design*. PWS Publishing Co., 1997.

[20] M. Hagan, H. Demuth, and M. Beale, "Neural network design (pws, boston, ma)," *Google Scholar Google Scholar Digital Library Digital Library*, 1996.

[21] W. Hämäläinen and M. Nykänen, "Efficient discovery of statistically significant association rules," in *2008 Eighth IEEE international conference on data mining*, IEEE, 2008, pp. 203–212.

[22] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *ACM sigmod record*, vol. 29, no. 2, pp. 1–12, 2000.

[23] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[24] H. Hsu and P. A. Lachenbruch, "Paired t test," *Wiley StatsRef: statistics reference online*, 2014.

[25] K. Kowsari, M. Heidarysafa, D. E. Brown, K. J. Meimandi, and L. E. Barnes, "Rmdl: Random multimodel deep learning for classification," in *Proceedings of the 2nd international conference on information system and data mining*, 2018, pp. 19–28.

[26] Lamda-Nju, *Lamda-nju/deep-forest: An efficient, scalable and optimized python framework for deep forest (2021.2.1)*. [Online]. Available: `https://github.com/LAMDA-NJU/Deep-Forest`.

[27] J. Li and O. R. Zaiane, "Exploiting statistically significant dependent rules for associative classification," *Intelligent Data Analysis*, vol. 21, no. 5, pp. 1155–1172, 2017.

[28] W. Li, J. Han, and J. Pei, "Cmar: Accurate and efficient classification based on multiple class-association rules," in *Proceedings 2001 IEEE international conference on data mining*, IEEE, 2001, pp. 369–376.

[29] B. Liu, W. Hsu, Y. Ma, *et al.*, "Integrating classification and association rule mining.," in *Kdd*, vol. 98, 1998, pp. 80–86.

[30] B. Liu, Y. Ma, and C. K. Wong, "Improving an association rule based classifier," in *European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, 2000, pp. 504–509.

[31] J. Prusa, T. M. Khoshgoftaar, and A. Napolitano, "Utilizing ensemble, data sampling and feature selection techniques for improving classification performance on tweet sentiment data," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2015, pp. 535–542.

[32] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[33] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[34] N. Sood and O. R. Zaiane, "Building a competitive associative classifier," in *International Conference on Big Data Analytics and Knowledge Discovery*, Springer, 2020, pp. 223–234.

[35] L. Sun, Z. Mo, F. Yan, *et al.*, "Adaptive feature selection guided deep forest for covid-19 classification with chest ct," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 10, pp. 2798–2805, 2020.

[36] R. Tanno, K. Arulkumaran, D. Alexander, A. Criminisi, and A. Nori, "Adaptive neural trees," in *International Conference on Machine Learning*, PMLR, 2019, pp. 6166–6175.

[37] P. Welke, F. Alkhoury, C. Bauckhage, and S. Wrobel, "Decision snippet features," in *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, 2021, pp. 4260–4267.

[38] X. Yin and J. Han, "Cpar: Classification based on predictive association rules," in *Proceedings of the 2003 SIAM international conference on data mining*, SIAM, 2003, pp. 331–335.

[39] O. R. Zaiane and M.-L. Antonie, "On pruning and tuning rules for associative classifiers," in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, Springer, 2005, pp. 966–973.

[40] Z.-H. Zhou and J. Feng, "Deep forest," *National Science Review*, vol. 6, no. 1, pp. 74–86, 2019.