A Novel Framework for Dynamic Ensemble Selection Using Weighting Approach

Author

Aiman Qadeer

Fall 2017-MS-17(CSE)00000204282

MS-17(CSE)

Supervisor

Dr. Usman Qamar

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

January, 2020

A Novel Framework for Dynamic Ensemble Selection Using Weighting Approach

Author

Aiman Qadeer

Fall 2017-MS-17(CSE)00000204282

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

MS Computer Software Engineering

Thesis Supervisor:

Dr. Usman Qamar

Thesis Supervisor Signature:____

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY, ISLAMABAD

January, 2020

Declaration

I certify that this research work titled "*A Novel Framework for Dynamic Ensemble Selection Using Weighting Approach"* is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.

Signature of Student

Aiman Qadeer

2017-NUST-MS-Computer Software-00000204282

Plagiarism Certificate (Turnitin Report)

This thesis has been checked for Plagiarism. Turnitin report endorsed by Supervisor is attached.

Signature of Student

Aiman Qadeer

Registration Number

00000204282

Signature of Supervisor

Dr. Usman Qamar

Language Correctness Certificate

This thesis has been read by an English expert and is free of typing, syntax, semantic, grammatical and spelling mistakes. Thesis is also according to the format given by the university.

Signature of Student

Aiman Qadeer

00000204282

Signature of Supervisor

Copyright Statement

Acknowledgements

I am thankful to my Creator Allah Subhana-Watala for guiding me throughout this work, for every new idea which You setup in my mind to improve my work. Indeed, I could have not done anything without Your priceless help and guidance. Whosoever helped me throughout the course of my thesis, whether my parents or any other individual was Your will, so indeed none be worthy of praise but You.

I am profusely thankful to my beloved parents who raised me when I was not capable of walking and continued to support me throughout in every aspect of my life.

I would also like to express special thanks to my supervisor Dr. Usman Qamar for his help throughout my thesis and also for teaching data engineering and web engineering courses. I can safely say that I haven't learned any other engineering subject in such depth than the ones which he has taught. I appreciate his patience and guidance throughout the whole thesis

I would also like to pay special thanks to Dr. Saad for his tremendous support and cooperation. Each time I got stuck in something, he came up with the solution. Without his help I wouldn’t have been able to complete my thesis. I appreciate his patience and guidance throughout the whole thesis.

I would also like to thank Dr. Wasi Haider and Dr. Muhammad Abbas my GEC committee for their support and cooperation.

Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

Dedicated to my exceptional parents, husband and adored siblings whose tremendous support and cooperation led me to this wonderful accomplishment

Abstract

Dynamic Classifier Selection (DCS) techniques classifies the test sample only by the most competent classifiers. Hence, the major problem in DCS is to find the measures by which competence of classifiers in a pool can be calculated to find out the most competent classifiers. To tackle these issues, we suggest a Framework for Dynamic Ensemble Selection (DES) that uses more than one criterion to calculate the base classifier's competence level. The framework has three major steps. In first step, training data is used to create a pool consisting of different classifiers. In second step meta-classifier training is performed by extracting meta-features from training data. In third step meta-classifier uses meta-features extracted from test sample to perform an ensemble selection and to predict the final output. We have suggested some improvements in second step (training) and last step (generalization) of the framework. In training phase, four different models are used as meta-classifiers. While in generalization phase, dynamic weighting scheme is used where meta-classifiers will dynamically assign weights to selected competent classifiers based on their competence level and final decision will be aggregated using a weighting voting scheme. The modifications proposed in this paper altogether enhance performance and accuracy of the framework in contrast with other dynamic selection techniques in literature.

Key Words: *Multiple Classifier system, Dynamic Classifier Selection, Dynamic Weighting*

Table of Contents

Declaration i

Plagiarism Certificate (Turnitin Report) ii

Language Correctness Certificate iii

Copyright Statement iv

Acknowledgements v

Abstract vii

Table of Contents viii

List of Figures 1

List of Tables 2

CHAPTER 1: INTRODUCTION 4

1.1 Dynamic Ensemble Selection 4

1.2 Competence of Classifier for Dynamic Selection 6

1.2.1 Local Accuracy of a Classifier 7

1.2.2 Decision Templates 7

1.2.3 Confidence of the Classifier 7

1.3 META-DES Framework 8

1.3.1 Overproduction 8

1.3.2 Meta-Training Phase 8

1.3.3 Generalization Phase 10

1.4 Problem Statement 10

1.5 Scope 10

1.6 Relevance to National Needs 10

1.7 Advantages 11

1.8 Areas of Applications 11

CHAPTER 2: LITERATURE REVIEW 13

2.1 Publishers 13

2.2 Quality Assessment 14

2.2.1 Effective Technique Proposed 14

2.2.2 Results Validation 14

2.2.3 Repetition 14

2.2.4 Recent Researches 14

2.3 Related Work 15

2.3.1 Dynamic Classifier Selection 16

2.3.2 Supervised Clustering Technique 19

2.3.3 Pool Generation 19

2.3.4 Ensemble Construction Techniques 19

2.3.5 Meta-Features Selection.......... 21
2.3.6 Unsupervised Clustering Technique.......... 22
2.3.7 Machine Learning Classification Techniques.......... 23
CHAPTER 3: PROPOSED METHODOLGY.......... 28
3.1 Overproduction Phase.......... 30
3.2 Meta-Training Phase.......... 31
3.2.1 Sample Selection.......... 31
3.2.2 Meta-Feature Extraction.......... 31
3.2.3 Training.......... 33
3.3 Generalization Phase.......... 34
CHAPTER 4: IMPLEMENTATION AND EXPERIMENTS.......... 36
4.1 Proposed Framework.......... 36
4.2 Datasets.......... 43
4.2.1 Heart Dataset.......... 43
4.2.2 Diabetes Dataset.......... 43
4.2.3 Blood Transfusion Dataset.......... 44
4.2.4 Sonar Dataset.......... 45
4.2.5 Cardiotocography (CTG) Dataset.......... 45
4.2.6 Phoneme Dataset.......... 46
4.2.7 Breast Cancer Dataset.......... 47
4.2.8 Thyroid Dataset.......... 47
4.3 Training and Testing.......... 49
4.4 Evaluation Measures.......... 50
4.5 Confusion Matrix.......... 50
4.5.1 Accuracy.......... 50
4.5.2 Precision.......... 51
4.5.3 Recall.......... 51
4.5.4 F-Measure.......... 51
4.6 Parameter Setting.......... 52
4.7 Experiment Protocol.......... 52
CHAPTER 5: RESULTS AND COMPARISONS.......... 54
5.1 Classification Results.......... 54
5.1.1 Classification Results on Heart Dataset.......... 54
5.1.2 Classification Results on Diabetes Dataset.......... 54
5.1.3 Classification Results on Blood Transfusion Dataset.......... 55
5.1.4 Classification Results on Sonar Dataset.......... 55
5.1.5 Classification Results on Cardiotocography (CTG) Dataset.......... 56
5.1.6 Classification Results on Phoneme Dataset.......... 56
5.1.7 Classification Results on Breast Cancer Dataset.......... 57

5.1.8 Classification Results on Thyroid Dataset 57
5.2 Comparison with State of Art Techniques 59
5.2.1 Comparison with Dynamic Classifier Selection Techniques........ 59
5.2.2 Comparison with Static Classifier Selection Techniques........ 61
5.3 Results Evaluation 63
5.4 Discussion........64
CHAPTER 6: CONCLUSION AND FUTURE WORK 66
References........ 67

List of Figures

Figure 1.1-1: The Stages of Multiple Classifier System 4
Figure 2.1-1: Papers Selected from Scientific Databases........ 13
Figure 2.2-1: Papers Selected Per Year 2015-2019........ 15
Figure 2.3-1: The Classical DCS Process 16
Figure 3.1-1: Overview of the Proposed Framework 28
Figure 3.2-1: Meta-Feature Vector that Predicts Base Classifier Behavior 33
Figure 4.1-1: Defining Pool of Classifiers........ 36
Figure 4.1-2: Pre-Processing of Dataset using PCA........ 36
Figure 4.1-3: Training of Decision Tree 36
Figure 4.1-4: Training of Naïve Bayes 37
Figure 4.1-5: Training of K-Nearest Neighbor........ 37
Figure 4.1-6: Training of Support Vector Machine........ 37
Figure 4.1-7: Training of Logistic Regression 37
Figure 4.1-8: Training of Neural Network........ 38
Figure 4.1-9: Making Clusters by Using K-Means........ 38
Figure 4.1-10: Method for Forming Output Profile of Instances........ 39
Figure 4.1-11: Method for Calculating Meta-Features 40
Figure 4.1-12: Training of Meta-Classifier: Random Forest........ 41
Figure 4.1-13: Training of Meta-Classifier: Support Vector Machine 41
Figure 4.1-14: Training of Meta-Classifier: Multilayer Perceptron........ 41
Figure 4.1-15: Training of Meta-Classifier: Logistic Regression........ 41
Figure 4.1-16: Method for Dynamically Calculating Weights for Classifiers........ 42
Figure 4.1-17: Method for Predicting Class Labels........ 42
Figure 4.3-1: 10-Fold Cross Validation........ 49
Figure 5.3-1: Comparison of Different Models Results........ 64

List of Tables

Table 2.3-1. Comparison Between State-Art Techniques 24
Table 3.1-1. Attributes of the Classifiers in the Pool 30
Table 3.2-1. Criterion and Paradigm for Meta-Features........ 32
Table 3.2-2. Attributes of the Meta-Classifiers........ 33
Table 4.2-1. Description of Heart Dataset........ 43
Table 4.2-2. Description of Diabetes Dataset 44
Table 4.2-3. Description of Blood Transfusion Dataset........ 44
Table 4.2-4. Description of Sonar Dataset 45
Table 4.2-5. Description of Cardiotocography Dataset........ 46
Table 4.2-6. Description of Phoneme Dataset 46
Table 4.2-7. Description of Breast Cancer Dataset........ 47
Table 4.2-8. Description of Thyroid Dataset 48
Table 4.2-9. Important Features of Datasets........ 48
Table 4.5-1. The Confusion Matrix 50
Table 5.1-1. Classification Results on Heart Dataset........ 54
Table 5.1-2. Classification Results on Diabetes Dataset 55
Table 5.1-3. Classification Results on Blood Transfusion Dataset 55
Table 5.1-4. Classification Results of Sonar Dataset 56
Table 5.1-5. Classification Results of Cardiotocography Dataset........ 56
Table 5.1-6. Classification Results of Phoneme Dataset........ 57
Table 5.1-7. Classification Results of Breast Cancer Dataset........ 57
Table 5.1-8. Classification Results of Thyroid Dataset........ 58
Table 5.1-9. Summarized Classification Results 58
Table 5.2-1. Comparison with State-of-Art Dynamic Classifier Selection Techniques........ 60
Table 5.2-2. Comparison with State-of-Art Static Classifier Selection Techniques 62

Chapter 1

Introduction

CHAPTER 1: INTRODUCTION

This thesis is about dynamic ensemble selection framework that dynamically selects classifiers for an ensemble formation. Different techniques and approaches regarding to dynamic selection of classifiers are also discussed for accurate predictions.

1.1 Dynamic Ensemble Selection

Multiple Classifier System (MCS) is a combination of classifiers functioning in parallel to form an ensemble that gives an accumulated output. In past years, many different techniques for designing and integrating multiple classifiers have been proposed. Although results showed that multiple classifiers have good performance as compared to single classifier where outputs from multiple classifiers are combined to give an aggregated result. Hence, Multiple classifier system (MCS) aims to integrate classifiers to improve accuracy, resilience, robustness and performance in pattern Reorganization systems [1]. MCS has three stages [2]: (i) Generation (ii) Selection (iii) Integration as illustrate in Figure 1.1-1. Firstly, a pool of different classifiers is created; then a single or group of best classifiers from the pool is selected. This group of classifiers is called Ensemble of Classifiers. In last stage, final prediction is obtained by aggregating the outputs of the selected classifiers. Recent work in literature had illustrated that the Dynamic Classifier Selection (DCS) methods in MCS accomplish higher accuracy than static ones [3]. This is also valid for the problems having small size of training data that is not enough for the training of classifiers [4][5].

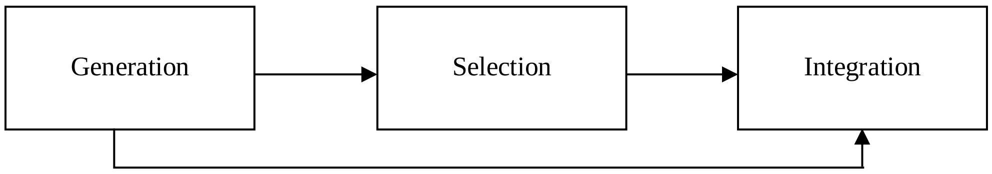


Figure 1.1-1: The Stages of Multiple Classifier System

For Selection phase, two types of selection techniques can be used i-e dynamic and static. In static technique, classifiers are selected during the training phase and then unseen test samples are classified by using these selected classifiers or Ensemble of Classifiers (EoC). In contradistinction, dynamic ensemble selection technique (DES) for each new test sample different ensemble of classifiers is used for classification. DES techniques assume that only a single classifier is expert in certain feature space of local region. So, when a new test sample arrives, most competent classifiers from the pool are selected to form an EoC depending upon the local region of feature space of test sample. Classifiers that have some competence level according to selection criteria are selected. Recent studies in dynamic ensemble selection framework show that dynamic selection techniques are effective for ill-defined classification problems i-e where the training data length is inadequate for the accurate selection of classifiers.

The main problem in DES is to define a method for the measurement for the competence of the classifiers. Different DES techniques use different selection criterion like local accuracy of the classifier in feature space, posterior probability, output profile and confidence of the classifier. However, from the previous studies, we collaborated that local accuracy itself is not enough to achieve good results that are near to oracle performance. Oracle is a conceptual model that always selects the best classifier that had correctly identified the label of test sample. Hence, it is regarded as an ultimate classification scheme. But oracle is very complex as compared to other classification schemes.

Other methods of DES like overproduce-and-choose strategy encounter problems when a fixed subset of classifiers is defined using training/optimization data and is not adapted for the classification of test data. This problem is similar as finding a universal best classifier that can classify each test sample accurately." No Free Lunch" theorem states that no algorithm is best over all the classification class problems. Using just a single technique for the measuring the competence of classifier is fallible. Thus, combining different methods can result in achieving good result from dynamic ensemble selection methods.

To tackle these limitations, we suggest a Dynamic Ensemble Selection (DES) Framework, that uses more than one criterion to calculate the base classifier's competence level. The framework has two main environments: (a) Classification Environment that map features of the input to class label, (b) Meta-Classification Environment that extracts certain properties of classifier such as accuracy from training data and encode them as meta-features. When a new test sample arrives, the meta classifier uses meta-features extracted from test sample as an input. The meta-classifier then decides which classifier is competent for the test sample classification. The framework has three major steps: (a) Overproduction, in which a pool consisting of different classifiers is created; (b) Meta-Training, in which meta-classifier training is performed by extracting meta-features from training data; (c) Generalization, in which meta-classifier uses meta-features of the test sample to construct an ensemble model. In this paper, we suggest improvements of DES framework. First, we will improve training method of meta-classifier. The improvement suggested is inspired by the fact that there exists a solid relationship between the performance of meta-classifier for the choice of competent classifiers, i.e., classifiers that anticipate the right label for a given test sample and the accuracy of the DES framework. Thus, we believe that the proposed framework will help us to achieve a good performance by improving the meta classification level. Four models i-e Support Vector Machine (SVM), Random Forest, Naïve Bayes and Neural Network (NN) are considered for the meta-classifier. Second, we propose a dynamic weighting scheme for generalization phase where meta classifiers will dynamically assign weights to selected competent classifiers base on their competence level and final decision will be aggregated using a weighting voting scheme. Thus, the classifier having higher competence level will have greater effect on the final decision. The proposed framework is unique from other techniques as it is based on the dynamic selection of ensemble rather than the static selection. The modifications proposed altogether enhance performance and accuracy of the framework in contrast with other dynamic selection techniques in literature.

1.2 Competence of Classifier for Dynamic Selection

Classifier competence is defined as the tendency of classifier to perform a classification assignment. The term competence is extensively used in machine learning as the method of selecting a classifier or an ensemble of classifier that can accurately solve the problem. Let C = {c_1, c_2, ………c_N} having size N and c_i is the base classifier of this pool. The aim of dynamic selection technique is to select a subset of classifiers to form an ensemble that can best classify the test sample xi. It is different from static selection technique in which classifier or an ensemble of classifier is selected in training phase and remains the same for every test sample, whereas in dynamic selection technique selection is performed in testing phase and varies according to the test sample.

The main problem in dynamic selection is that for a test sample how to calculate the competence of the base classifier. In previous work, we have seen that different criteria's have been used for the selection for the calculation of classifier's competence local accuracy of the classifier in feature space i-e the region in feature space around the test sample, decision templates i-e methods that's are used in decision space and confidence. These techniques have been described in following sub-sections.

1.2.1 Local Accuracy of a Classifier

Local accuracy of a classifier in feature space is the mostly commonly used technique for the calculation of classifier competence. A small region in the feature space around the test sample is defined in this technique. This region can be calculated either by using supervised learning technique i-e KNN algorithm in which k nearest neighbors are included in this region by using Euclidean distance or by using unsupervised learning technique i-e K-Means clustering algorithm in which local accuracy of a classifier is calculated over a cluster. Based on the number of samples included in local region, classifier competence is computed. However, using local accuracy alone is not enough. Other techniques can be combined with local accuracy to achieve good results.

1.2.2 Decision Templates

In this technique, samples that are very close to test sample are selected. However, to calculate similarity decision space around a classifier is used. Output profiles of both training set and test sample are formed that shows the decision of classifier c_i for that sample. Based on the information from decision space, nearest output profiles to the test sample are selected to form an output profile best. The base classifier that achieves good accuracy in output profile set is selected. The advantage of this technique is that it is not restricted by the quality of region of competence in feature space, as similarity is computed from decision space. But the disadvantage is that global information is considered, so local competence of the classifier is neglected.

1.2.3 Confidence of the Classifier

In this technique, rather than generating a pool of classifiers, pool of EoC i-e C ={C_1, $C_{2,}$………C_N} having size N has been populated by using greedy search or genetic algorithm. Then for each test sample, competence level of EoC C_i is equal to its consequences among base classifiers. Ambiguity among the base classifiers of an ensemble is computed. Ambiguity is the ratio among the base classifiers that agrees with decision to the base classifiers that disagrees. The advantage of this technique is that it is not dependent on the information from region of competence of feature space or decision space. However, the disadvantage of this technique is that search is unable to find an acceptable EoC with good confidence level.

1.3 META-DES Framework

META-DES is a framework for dynamic ensemble selection. The framework has three major steps:

1. Overproduction
2. Meta-Training
3. Generalization

1.3.1 Overproduction

The first phase of META-DES is overproduction phase in which a pool of classifiers is created by using different techniques. The pool can be created by using bagging, boosting, choose and overproduce strategy, random subspaces and genetic algorithm. The size of pool can be defined by the user. An appropriate pool size depends upon the number of base classifiers chosen. Best pool size for the datasets can be obtained by using brute force approach. However, it is an expensive approach that can affect the efficiency of the DES framework. Other techniques like intrinsic classification complexity of the problem can be used to predict accurate pool size.

1.3.2 Meta-Training Phase

The second phase of META-DES is meta-training phase in which meta-classifier is trained by computing the meta-features. Different approaches are used for selecting the meta-classifier. There can be a single meta-classifier or more than one meta-classifier depending upon the approach used. Meta- Training phase is further divided into three phases:

- Sample Selection
- Meta-Feature Extraction
- Training

Sample Selection

In this phase of meta-training, samples that are used for the training of meta-classifier are selected. These samples can be selected by using different techniques i-e by calculating the consensus of the pool or by setting the threshold. One of the key issues of DES arises when consensus in the pool is low i-e winning class votes are very close to the votes of other class. To deal with this issue meta-classifier is trained in such a way that it can handle such cases. This is done by setting threshold *t* called consensus threshold. For all training sample $x_{j,train}$ the degree of consensus for pool is calculated as $H(C\ ,\ x_{j,train})$. If it is less than *t* i-e $H(C\ ,\ x_{j,train})<t$, then the training sample $x_{j,train}$ is passed to next phase.

Meta Feature Extraction

In this phase of meta-training, meta-features are extracted for each training sample. Different sets of meta-features can be extracted. Each feature set is used to calculate the competence level of the base classifier. Different property about the behavior of the base classifier is measured by each feature set. For the measurement of the meta-features, region of competence is defined for each training sample. Region of competence can by defined by using supervised learning techniques like KNN where k nearest neighbors of each samples are considered to be in the region of competence or by using unsupervised technique like K-Means in which the all the samples in the cluster in which the sample lies is considered to be in its region of competence. Euclidian distance can be used for the measurement in the case of KNN. A set of meta-features is obtained at the end of this phase. For each training sample, number of meta-features vectors obtained is equal to the size of pool of classifiers each vector corresponds to the single classifier in pool. After this phase, meta-features dataset is obtained.

Training

In this phase of meta-training, meta-classifier is trained over meta-features dataset. If there are more than one meta-classifiers, each of them is trained over the meta-features dataset by splitting it into testing and training data.

1.3.3 Generalization Phase

The third phase of META-DES is generalization phase, in which when a test sample arrive, it region of competence is computed and samples in its region of competence are selected. After that, for base classifier in pool meta-feature extraction process is called for the test sample. The meta-feature vectors dataset obtained after this process is then fed to meta-classifier to compute the competence level of the base classifier. Threshold value can be set for the competence level, that if the competence level of classifier is above the threshold then it should be selected for the ensemble model. After the formation of an ensemble model, majority voting rule or weighted voting approach can be used for the prediction of final output.

1.4 Problem Statement

Dynamic Ensemble Selection is required to

1. Select most competent classifiers from the pool of classifiers to classify the test sample.
2. Automatically construct an ensemble of competent classifiers.

This method is proposed to dynamically form an ensemble such that the overall system accuracy is higher than individual classifiers accuracies.

1.5 Scope

In this research work we will focus on:

1. Formulation of a novel dynamic ensemble selection framework that can achieve a higher overall accuracy then the accuracies separately recorded by single classifiers.
2. Evaluation of the proposed framework on datasets from different repositories.

1.6 Relevance to National Needs

This algorithm will facilitate the computerized storage of data in various fields like in health, business, education and government sectors by performing an automatic classification thus helps in more accurate predictions.

1.7 Advantages

- The framework helps to choose classifiers with high competence level to have greater impact on the final prediction and to deal with the case of noisy samples where all classifiers from pool will be considered as competent for an ensemble and incompetent classifiers will be having equal impact on final output as that of competent ones.
- Thus, framework suggested will enhance performance and accuracy of the framework in contrast with other dynamic selection techniques in literature.

1.8 Areas of Applications

- Health
- Government
- Education
- Business

Chapter 2

Literature Review

CHAPTER 2: LITERATURE REVIEW

2.1 Publishers

This part of document briefly describes the previous state of arts techniques that are used for dynamic ensemble selection of classifiers. Methodology and results of previous studies in literature are discussed here.

Research process is carried out in a systematic manner. Data is collected by a search process that includes finding researches relevant to the work and selecting the most relevant among them. These selected researches are analyzed for quality and data is extracted from them.

Following research publishers are considered for this research.

- IEEE
- Springer
- Elsevier

Figure 2.1-1 represents the papers selected from different databases.

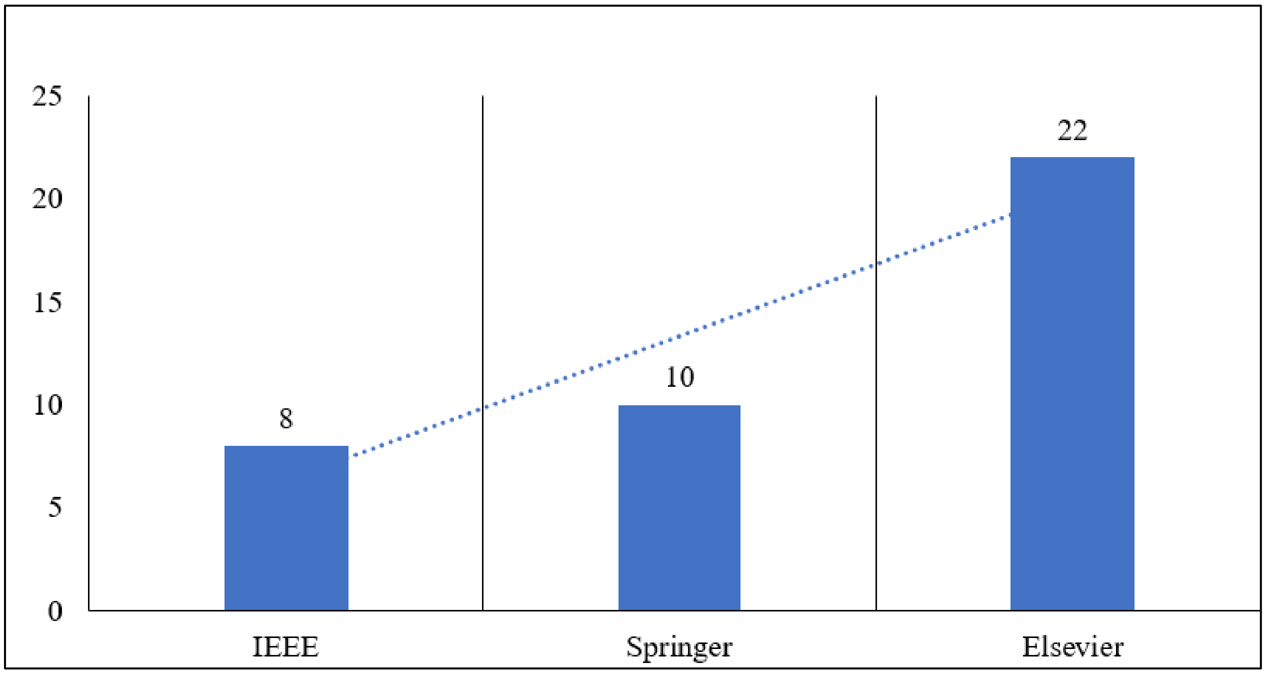


Figure 2.1-1: Papers Selected from Scientific Databases

2.2 Quality Assessment

There were certain quality factors that were used for carrying out the research for the research papers selected from above databases. These quality factors are described below:

2.2.1 Effective Technique Proposed

It is most important factor about the selected paper was that it should propose some technique, method or model regarding dynamic classifier selection. All remaining papers were eliminated from the search results.

2.2.2 Results Validation

All those papers that do not give any assessment of the results supported by the validation of some dataset are excluded from the search results.

2.2.3 Repetition

Only those papers that contain some new and unique researches are considered. Those representing same new methodology or models are included.

2.2.4 Recent Researches

Most of the papers from the recent 5 years i-e 2015-2018 and the current year (2019) researches are collected for analysis as they are the most updated ones. Figure 2.2-1 represents the papers selected per year.

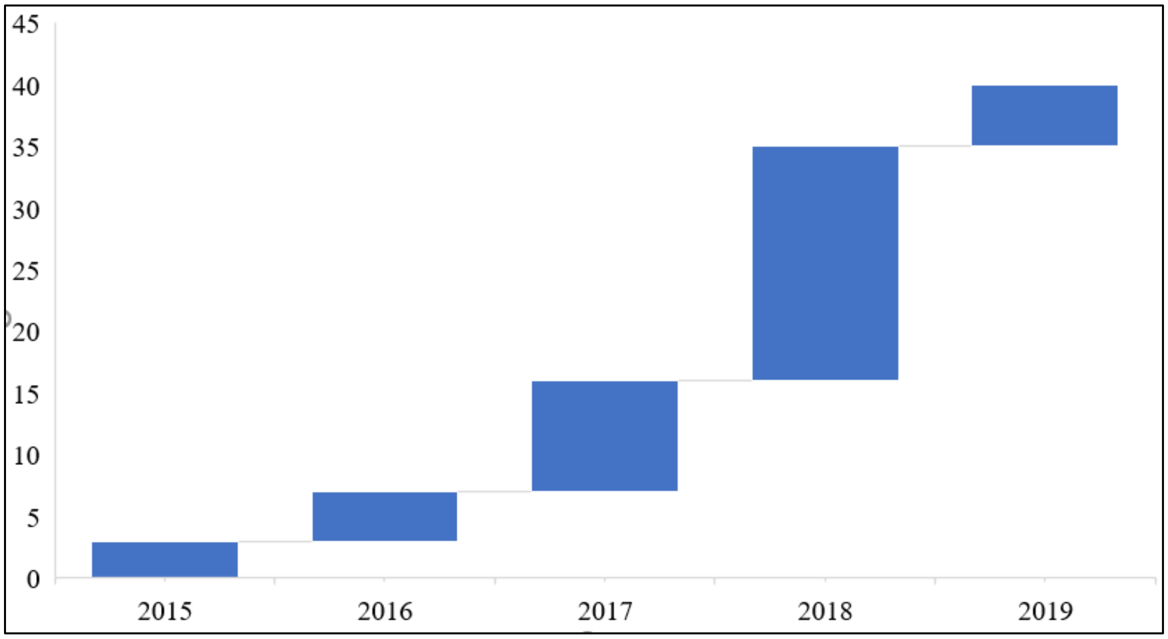


Figure 2.2-1: Papers Selected Per Year 2015-2019

2.3 Related Work

Multiple Classifier Systems (MCS) uses either Static Classifier Selection (SCS) methods in which classifiers are defined during training phase and are used to predict test samples labels in the generalization phase, or Dynamic Classifier Selection (DCS) methods that selects the single or subset of competent classifiers as an ensemble from the pool of classifiers based on test sample. This property had made DCS technique a robust approach. Classical DCS had mainly three phases, as shown in Figure 2.3-1. First is Classifier Generation Phase in which training data (T) is used to produce a pool of classifiers, second is Region of Competence Generation Phase in which T is used to produce the competence region (Ry) and third is Dynamic Selection Phase that selects the classifiers that are competent based on (Ry). The selection phase is dynamic in DCS as it is based on the test sample. However, in literature many DCS methods are described for selection of the most competent classifiers. The difference between DCS methods described in literature is based on the technique used for the selection of competent classifiers that can classify the test sample more accurately.

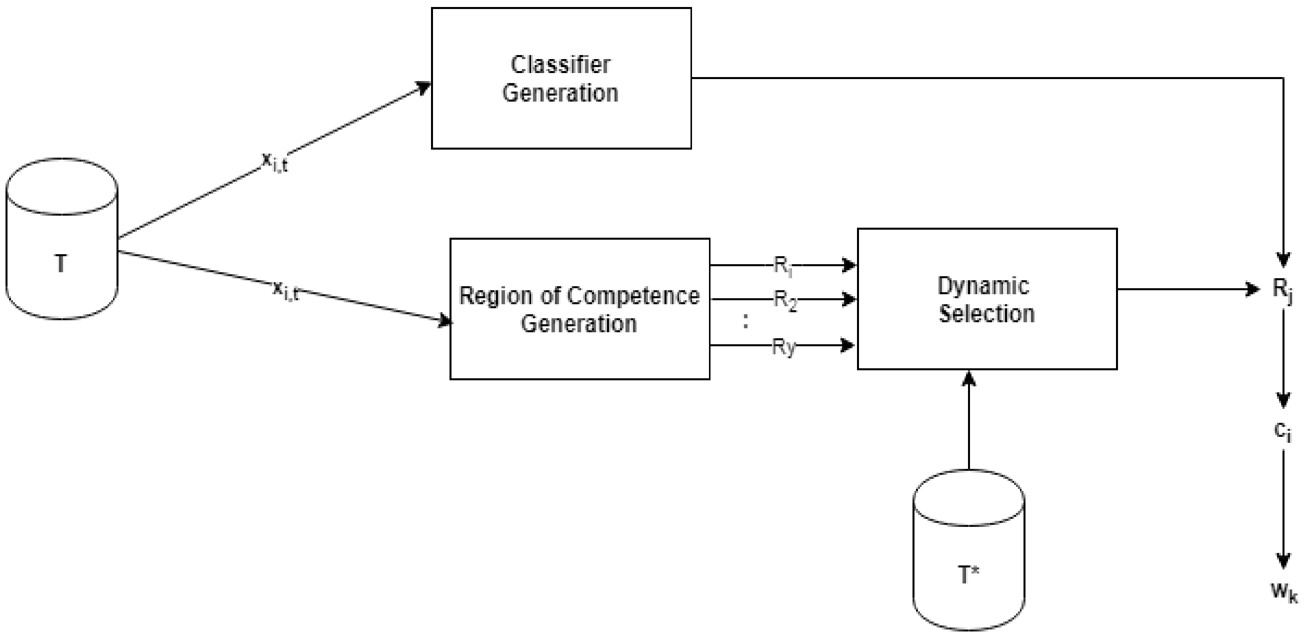


Figure 2.3-1: The Classical DCS Process

2.3.1 Dynamic Classifier Selection

In 2015, Rafael M. O. Cruz et al. [6] proposed a technique for the dynamic selection of classifier by using meta-learning. Five different sets of meta-features were proposed where each feature correspond to some criteria that is used to find out the competence of the classifier. These meta-features are extracted from training data and are used to train the meta-classifier. Experiments were conducted on different datasets and results were compared with other state-of-art of techniques. Highest accuracy achieved by using technique was *96.21%.* Results showed that this technique outperforms from other state-of-art techniques. Tomasz Woloszynski et al. [7] proposed a random classification selection strategy based on competence measure used majority voting rule that dynamically selects a set of classifiers that can perform better than a randomly selected classifier for every test sample and eliminates the weak (incompetent) classifiers. The strategy developed above achieved the highest classification accuracy 87.04%.

In 2016, Rafael M. O. Cruz et al. [8] proposed a prototype selection technique that reduced the amount of overlap between the classes in validation data and produced smooth decision borders. A nearby versatile K-Nearest Neighbor calculation was utilized to limit the impact of noisy sample in the region of competence. DES techniques over 30 classification problems were used for experimentation. The results showed this technique improves accuracy of dynamic selection method and it achieves the best accuracy of 85%. Dhouha Mejri et al. [9] employs aggregation of different classifiers to improve the performance of single classifier by using batch learning process and reducing the chance of selecting non-competent classifier by monitoring shifts in data samples. Accuracy achieved by using this method was up to 89%.

In 2017, Andrel Brun et al. [10] described a technique for Dynamic Classifier Selection (DCS) by using certain features that address the classification difficulty in terms of pool creation and selection of classifier. This difficulty of classification was described by using complexity measures of meta-features calculated from problem data. Experimentation was performed on 30 datasets. Results showed that the proposed technique using complexity measures achieved better classification accuracy of 63.3% when compared to other similar methods in the literature. Rui Ye et al. [11] combined path -relinking, variable neighborhood search and random adaptive search algorithms for dynamic selection of an ensemble to increase the accuracy of DES.Bartosz et al. [12] conducted a survey on different techniques of static and dynamic selection of ensemble and concluded that DES are techniques are not only better in terms of accuracy but also in terms of computational requirements i-e memory and time.

In 2018, techniques were proposed for imbalance datasets. Pablo Perez-Gallego et al. [13] proposed a selection criterion for problems in quantification, where two of them were for dynamic ensemble selection while one was for static ensemble selection. The experiments showed that using above criteria achieves the performance of 70% as compared to the other ensembles where all the models are averaged. Salvador Garcia et al. [14] proposed a novel Dynamic Ensemble Selection (DES) technique to deal with imbalanced datasets which consists of two components i-e balancing of training dataset with some data pre-processing method, and the selection of competent classifiers, in which the classifiers competence level is compared through region of competence weightage. The proposed method aims to improve the classification performance in case of imbalance datasets. Experiments were performed using different imbalanced datasets. Results showed that of the proposed technique is effective for multi-class imbalanced datasets at a level of 5% significance. Cheriguene et al. [15] suggested the use of diversity and accuracy measures with greedy search algorithm for determining the optimal set of classifiers. Q-statistic and Disagreement measures are used to calculate the diversity among members. The experiments were performed on 24 different datasets showed that the proposed method had higher performance as compared to the other ensembles selection methods and it has achieved the high diagnosis sensitivity of 79.27%, specificity of 94.03%, and F - measure of 82.73%. as compared to other methods of selections. Paulo R.L. Almeida et al. [16] demonstrate that with drift phenomenon DCS approach can be more powerful, especially in cases where some areas do not change. In such cases area/local dependency measure is not enough, thus a time dependency measure must also be used. The experimental results showed that this approach achieves an accuracy of 92.2% with no parameter tuning. Xiaodong Feng et al. [17] employed classification ability and relative cost of classifier in validation set for the selection of classifiers. After selection, classifiers were combined for samples by using the probability. Experiments conducted on real datasets revealed that the proposed method achieved and accuracy of 95%. Rafael M.O. Cruz et al. [18] proposed FIRE-DES++, an enhanced version of FIRE-DES in which equal number of classes sample are used to define region of competence by which noise and overlapping between classes can be reduced. Experiments were performed on 64 datasets. Results showed that FIRE-DES++ outperforms on 7 out of 8 datasets as compared to other state of art techniques. Highest accuracy achieved by using this technique was 85.17%. One of drawback of this technique was that it does not handle multi-class classification problem. Rafael M. O. Cruz et al. [19] proposed an improvement in generalization phase of DES technique by reducing the overlap between classes through prototype technique and smother decision borders are produced. Noisy samples are reduced by using KNN algorithm. Experiments were conducted on 30 datasets and results were compared with 10 state of art techniques. Hence, it was proved that using this technique improves the accuracy. Highest accuracy achieved was *96.52%*.

2.3.2 Supervised Clustering Technique

In 2016, Hongshan et al. [20] proposed supervised clustering technique in which original dataset is partitioned into subsets such that each subset has same class members. Different class subsets are then combined to form training data and base classifier is selected in each subset. Weighted voting approach is used for combining the outputs of different classifiers. Accuracy of 90% was achieved by using this method. However, small-sized and heavily imbalanced datasets don't give significant results by using these techniques.

2.3.3 Pool Generation

In 2017, Yufei Xia et al. [21] integrated the stacking method with bagging algorithm. The proposed method differs from the other ensemble in terms of pool generation, base learner's selection, and trainable fuser. The Bstacking model uses XGBoost, SVM, RF and GPC classifiers as base learners and use bagging approach to train these models. Area Under the Curve (AUC), Accuracy, AUC-H and Brier score measures were using to calculate the performance. Bstacking showed good performance over other models and achieved best accuracy of 74%. Dayvid et al. [22] employs a method to detect the location of classifier in indecision region and prunes the pool of classifiers accordingly. Experiments conducted on different datasets showed that proposed method had same performance as compared to other methods in literature.

In 2018, Mariana A. Souza et al. [23] proposed an online learning generation approach for dynamic classifier selection. This technique generates local pool based on different regions of feature space for test data and help the DCS to select best classifier from the local pool that will have less chances to misclassify the data. Experimental results showed that more competent classifier can be selected from local pool than from global generated pool and give more accurate results as compared to other state of art techniques.

2.3.4 Ensemble Construction Techniques

In 2015, Saba Bashir et al. [24] presents a framework for ensemble selection by using weighted voting scheme with bagging approach for the prediction of heart diseases. Five

Classifiers Naïve Bayes, quadratic discriminant analysis, linear regression, support vector machine and instance-based learner are used as based classifiers. Experiments were performed on fived datasets and results were accessed using 10- fold validation. Highest accuracy of *84.16%* was achieved by using this framework.

In 2016, Saba Bashir et al. [25] proposed a framework that employs an ensemble of seven i-e Naïve Bayes (NB), Linear Regression (LR), Quadratic Discriminant Analysis (QDA), K Nearest Neighbor (KNN), Support Vector Machine (SVM), Decision tree using Information Gain (DT-IG) and Decision tree using Gini Index (DT-GI). The framework consists of three modules. The first module collects data from different resources and pre-process them. Then the second module performs classifiers training on the pre-processed data. The third module predicts the labels for test instances. The framework is tested upon different datasets from public repositories. Results shows that the framework has achieved highest accuracy in prediction of diseases on all medical datasets.

In 2017, Xiaoqian Liu et al. [26] gave the query analysis statistics for constructing decision trees and proposed algorithm that use multiple private decision trees built by using bootstrapped samples to construct the ensemble model and showed that the proposed ensemble improves accuracy and stability on real datasets. The most effective accuracy achieved the proposed ensemble is of *82%*. Jan N. van Rijn et al. [27] proposed a strategy that dynamically assigns weights to the classifiers of ensemble by using an internal evaluation criterion on training data, that measures the performance of an ensemble on this and dynamically update weights of classifiers. Experiments had showed that the accuracy achieved is *86%* as compared to other ensemble techniques like Online Bagging and Leveraging Bagging.

In 2018, Anandarup Roy et. al [28] suggested that the dynamic ensemble of classifiers gives promising results in the case of imbalance datasets. These ensembles were designed with aim to apply under-sampling and/or oversampling for balancing class proportions. Experiments were performed using various multi-class datasets. Results showed that by combining pre-processing technique with dynamic selection achieves higher performance as it increases the F-measure and the G-mean up to 2.58 and 2.38 respectively.

In 2019, Zhong-Liang Zhang et al. [29] improved dynamic weighting scheme for the classification ability of dynamic ensemble selection systems by using neighborhood of every class instance that assigns weights to dynamically adjust the decision boundary for correct class label. Results were observed on many real-world applications. Highest accuracy achieved by using this technique was 96.59% that outperforms other state-of-art techniques.

2.3.5 Meta-Features Selection

In 2017, Rafael M. O. Cruz et al. [30] proposed a scheme for meta-feature selection that uses discrete decision to improve meta classifier performance. The results of meta-classifier were supposed to be like the Oracle. 30 classification problems were used for experimentation. The proposed framework obtained an accuracy of *97%*, which is closer to the oracle results.

In 2018, Mahardhika et al. [31] proposed a technique called pENsemble in which classifiers are evolved from data streams by using pruning and online feature selection strategies. 10 % improvement was observed as compared to single selection classifiers techniques and other dynamic selection techniques i-e up to 80%. Khamar et al. [32] proposed an algorithm in which original data is projected into new space, to preserve its locality and different rotation matrix was constructed. Results showed that the proposed algorithm was better in terms of performance and complexity as compared to other algorithms. Best accuracy achieved by it was 91%. Carine A. Dantas et al. [33] proposed a technique that uses Dynamic Feature Selection in Dynamic Ensemble Selection (DES) methods. Analysis was performed on DES methods i-e KNORA-U and META-DES. Initial classifier pool contains 90 classifiers i-e 30 SVMs, 30 MLPs and 30 k-NNs. Experiments were conducted on 10 datasets from different repositories. Results revealed that the use Dynamic Feature Selection with DES methods improves performance and accuracy.

2.3.6 Unsupervised Clustering Technique

In 2018, Sarfaraz Hussein et al. [34] compared supervised and unsupervised approaches for the detection of pancreatic and lungs tumor. In supervised learning deep learning algorithms convolutional neural network and transfer learning were used. While in unsupervised learning proportion SVM was used. The proposed algorithms were evaluated on 1018 CT and MRI scans of lungs and pancreases tumor. Results showed that the proposed approach of unsupervised learning showed more good results than supervised learning. The best accuracy achieved in case was *78%*. Wenming Cao et al. [35] proposed multi-task unsupervised learning method with hierarchal data structure. The benefit of this approach was that it increases the similarity between the instances of same cluster and diversities between different clusters. Both feature space and sample space capabilities of clusters are used concurrently. Experiments were performed against state-of-art clustering techniques. Results revealed that the proposed method outperforms in efficiency and accuracy. Peijie Lin et al. [36] proposed a hybrid model by combining K-means clustering, grey relational analysis (GRA) and Elman neural network (Hybrid improved K-means-GRA-Elman, HKGE). Model was trained my using historical power datasets and multivariate meteorological factors. Improved K-means was used the clustering of historical power datasets. GRA was used for the finding the similarities between days. Elman neural network was used for finding the non-linear relationship between meteorological factors. Results compared with previous prediction models revealed that the proposed model outperforms in terms of accuracy. Highest accuracy achieved by using this model was *77.81%*. Hui Gu et al. [37] proposed a new clustering technique by combing K-means and fuzzy C-means. This technique is superior to fuzzy C-means clustering as it helps to find out the number of clusters without using prior knowledge. Results revealed that the proposed method outperforms in efficiency and accuracy. Minghu Wu et al. [38] proposed K-means clustering algorithm for global motion estimation by which motion vectors due to unstable and shaking environment can be removed. Feature points were matched using SURF algorithm. Proposed algorithm was applied on three videos and results were compared with other techniques in literature that showed the superiority of algorithm based on effectiveness and accuracy. Christian Lopez et al. [39] proposed an unsupervised machine learning technique that cluster patients based on their genomic features. Internal validity metrics are used to identify the number of clusters. Relationships between clusters are identified by using gene pathway. The technique was tested on other genomic datasets in literature. Results revealed that the technique has highest performance in comparison of other state-of-art techniques. Highest accuracy achieved was 78.57%.

In 2019, Sherrir et al. [40] used random forest with unsupervised clustering model along with aggregated statistics of crops by preventing the models from labels. Experiments were performed on crop data from US Department of Agriculture's Cropland. It was found that crop labels can be separated more accurately by harmonic coefficient using unsupervised clustering model and random forest when neighbor geographies and crop conditions are same. Results showed that the proposed algorithm exceeds the accuracy by *80%*. Neo Christopher Chung et al. [41] used deep learning approaches long-short term memory and deep convolutional embedded clustering for unsupervised classification of metabolites and proteins during cardiac remodelling in mice. Moreover, K-Means clustering and hierarchical clustering were also performed. Dataset used for training and testing used temporal trends images. Results showed that deep convolutional embedded clustering give more accurate results than conventional convolutional clustering and achieved highest average ranking of *1.72*. Haidong Zhong et al. [42] proposed an improved Reversible Image Transformation (RIT) that minimizes the additional information. Target images were divided into non-overlapping blocks and used K-means to classify them into K classes. Patching blocks were used to hide the secret segment. Results showed that proposed technique improved the quality of images effectively.

2.3.7 Machine Learning Classification Techniques

In 2015, Zahra Nematzadeh et al. [43] presents the effect of using K-fold cross validation on accuracy by applying different algorithms of machine learning. Neural Network, Decision Tree, Naïve Bayes and Support Vector Machine algorithms were used with different kernel values to classify Wisconsin Diagnostic Breast Cancer (WDBC). Different datasets from UCI were used for comparison. Results were tested on different values of K for K-fold cross validation. Study revealed that by increasing value K, computational cost increases as more folds are required for training and it does not have a significant impact on accuracy i-e by using higher value K does not mean that accuracy will be increased.

In 2017, R. Ani et al. [44] investigated that better accuracy can be provided for predicting diseases by using machine learning algorithms and proposed a model that uses random forest as base classifier and for feature projection Linear Discriminant Analysis was used. Results showed that Linear Discriminant Analysis gives better results than Principle Component Analysis. Highest accuracy achieved by using this model was 95% that outperforms other techniques in state-of-art.

In 2018, David A. Omondiagbe et al. [45] investigated machine learning techniques with other feature reduction methods and proposed a method that uses Linear Discriminant Analysis to reduce features dimensionality. This reduced feature dataset was fed to Support Vector Machine for classification. Wisconsin Diagnostic Breast Cancer (WDBC) Dataset was used training and validation. An accuracy of 98.82% was achieved by using this technique.

In 2019, Quinlan D. Buchlak et al. [46] presented a systematic review on different machine algorithms and their usage in machine learning applications. Systematic study provided 6866 results by using accuracy, specificity and sensitivity as performance statistics. Results showed that mostly Neural Network, Support Vector Machine and Linear Regression are used. Out of which Neural Network have sufficiently higher accuracy then Support Vector Machine and Support Vector Machine have sufficiently higher accuracy then Linear Regression. Neural Network outperformed other supervised learning techniques.

The comparison between these techniques is shown in Table 2.3-1.

Table 2.3-1. Comparison Between State-Art Techniques

Author	**Year**	**Technique**	**Accuracy**
Rafael et al. [6]	2015	Meta-Learning	*96.21%*
Tomasz et al. [7]	2015	Majority Voting Rule	*87.04%*
Rafael et al. [8]	2016	KNN	85%

Dhouha et al. [9]	2016	Batch Learning Process	89%
Andrel et al. [10]	2017	Complexity Measures	*63.3%*
Pablo et. al. [13]	2018	Both Dynamic and Static Methods	70%
Soraya et al. [15]	2018	Greedy Search Algorithm	79.27%
Paulo et al. [16]	2018	Drift Phenomenon	92.2%
Xiaodong et al. [17]	2018	classification ability and relative cost	95%
Rafael et al. [19]	2019	Prototype and Smoother Decision Boundary Technique	*96.52%*
Hongshan et al. [20]	2016	Supervised Clustering	90%
Yufei et al. [21]	2017	Stacking Method with Bagging Algorithm	74%
Xiaoqian et al. [26]	2017	Multiple Decision Tree	82%
Jan et al. [27]	2017	Internal Evaluation Criterion	86%
Rafael et al. [30]	2017	BPSO	97%
Mahardhika et al. [31]	2018	pENsemble	80%
Khamar et al. [32]	2018	Rotation Matrix Construction	91%

This research study evaluates different models available in literature for dynamic classifier selection. The major problem in DCS is to find the measures by which competence of classifiers in a pool can be calculated to find out the most competent classifiers. Another issue discovered was to find out the measures to create combined output by aggregating different classifiers results. Following research gaps have been deduced from above research study.

- Most of the approaches discussed in literature uses single meta-classifier.
- No aggregation function has been used in Generalization Phase.
- Multi-Class datasets are not handled.

Hence, to overcome these shortcomings a dynamic ensemble model has been proposed in this thesis.

Chapter 3

Proposed Methodology

CHAPTER 3: PROPOSED METHODOLGY

An abstract level diagram of the proposed methodology is shown in Figure 3.1-1.

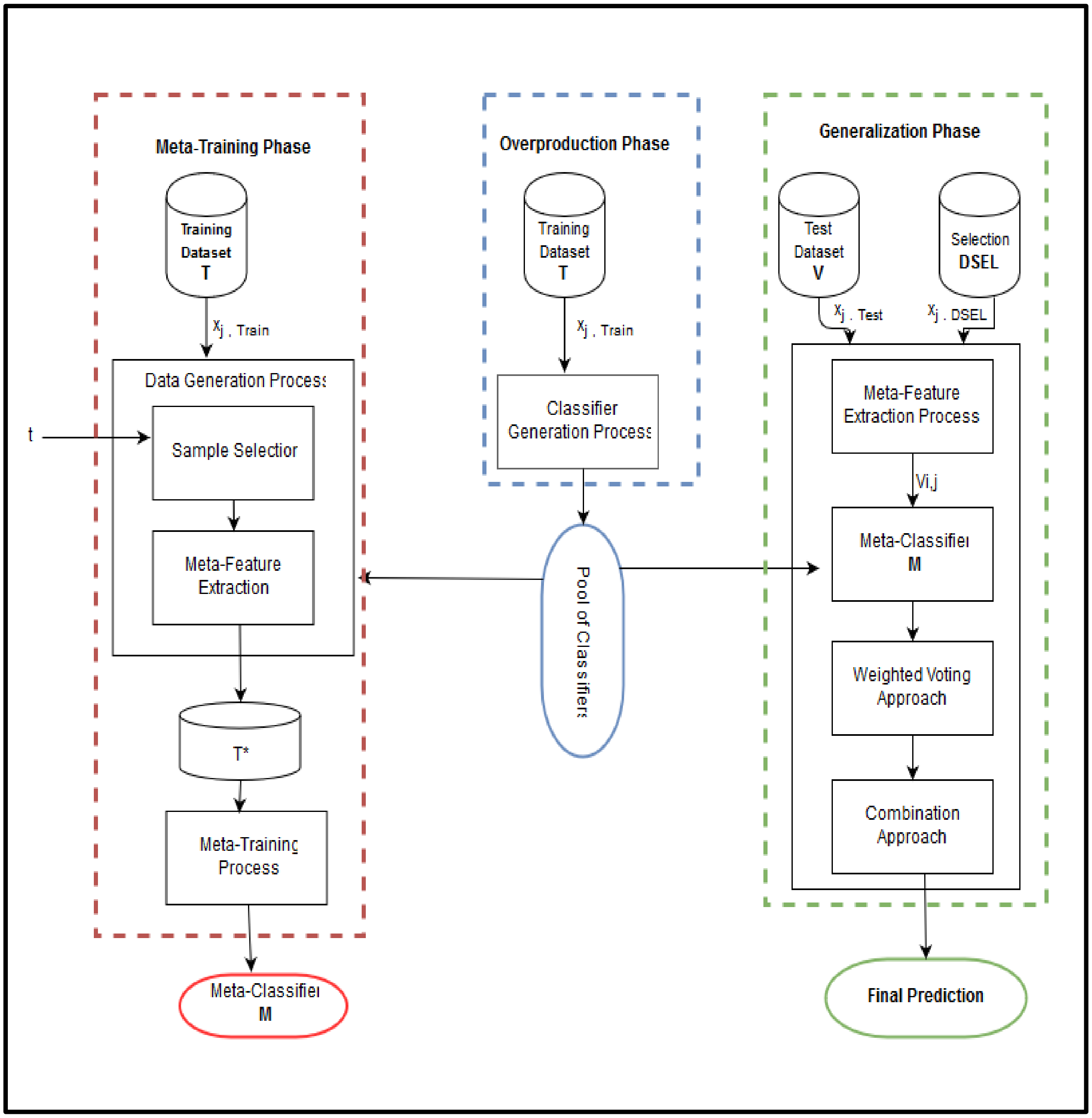


Figure 3.1-1: Overview of the Proposed Framework

1. Input training and test sets are first passed through an initial pre-processing step that removes the duplicate records and finds the relationships between attributes.
2. Training set is then used to train all the classifiers in the pool.
3. K-Means is used to make clusters of training instances and these clusters are used to find out the region of competence of each instance.
4. Output profile of each training instances is formed and stored in a vector that contains the decision of each classifier in pool for that instance.
5. After that meta-features are calculated for each instance in training set and a meta-features dataset is formed.
6. Meta-Features dataset is then used to train the meta-classifiers.
7. Then for each instance of test set steps from step 3 are repeated.
8. Meta-Features vector for each test instance is passed to the meta-classifiers that will compute the compete level for each classifier and weights are dynamically assigned to the competent classifiers.
9. After that using weighting voting approach, final class label for that instance is computed.

The dynamic classifier selection is a classification dilemma that uses various techniques to calculate the competence level of each classifier in the pool and determine whether the classifier is competent for the test sample or not. There are two basic environments for this framework: (i) Classification Environment that maps class label to the input features, (ii) Meta-Classification Environment that extracts meta-features from the training dataset. The proposed framework is defined as:

- For each test sample, classifiers can be regarded as “competent” or “incompetent”.
- To calculate the competence level of base classifiers different meta-features are computed.
- Vectors are formed from these meta-features.
- These vectors are then used for the training of meta-classifiers that predicts whether classify is competent and depicts its label.

The framework is divided into three main steps i-e Overproduction Phase, Meta-Training Phase and Generalization Phase that are explained as follows:

3.1 Overproduction Phase

The first phase of META-DES is overproduction phase in which a pool of classifiers is created by using different techniques. The pool can be created by using bagging, boosting, choose and overproduce strategy, random subspaces and genetic algorithm. The size of pool can be defined by the user. An appropriate pool size depends upon the number of base classifiers chosen. Best pool size for the datasets can be obtained by using brute force approach. However, it is an expensive approach that can affect the efficiency of the DES framework. Other techniques like intrinsic classification complexity of the problem can be used to predict accurate pool size. In proposed methodology different classifiers are included in the pool. The pool contains Decision Tree Classifier, Gaussian NB, K Nearest Neighbors Classifier, SVM, Logistic Regression and Neural Network. The attributes of classifiers are defined in Table 3.1-1.

Table 3.1-1. Attributes of the classifiers in the pool

Sr. No.	Classifier Name	Attributes
1.	Decision Tree	Random State = 80 criterion = "entropy" Max Depth=200 Min Samples Leaf=22
2.	Gaussian NB	Priors=None Var Smoothing=1e^{-09}

3.	K Nearest Neighbors	Neighbors=5
		Weights='uniform'

4.	SVM	Gamma='auto'
		Probability = True
		Random State = none

5.	Logistic Regression	Max Iterations = 100 N Jobs = none Random State = none

6.	Neural Network	Solver='lbfgs'
		Alpha=1e^{-5}
		Hidden Layer Sizes=(10, 10)
		Random State=10

3.2 Meta-Training Phase

The second phase of META-DES is meta-training phase in which meta-classifier is trained by computing the meta-features. Different approaches are used for selecting the meta-classifier. More than one meta-classifier used in this approach. Meta-Training phase is further divided into three phases:

1. Sample Selection
2. Meta-Feature Extraction
3. Training

3.2.1 Sample Selection

In this phase of meta-training, samples that are used for the training of meta-classifier are selected. These samples can be selected by using different techniques i-e by calculating the consensus of the pool or by setting the threshold. One of the key issues of DES arises when consensus in the pool is low i-e winning class votes are very close to the votes of other class. To deal with this issue meta-classifier is trained in such a way that it can handle such cases. This is done by setting threshold *t* called consensus threshold. For all training sample $x_{j, train}$ the degree of consensus for pool is calculated as $H(C , xj,train)$. If it is less than *t* i-e $H(C , x_{j,train})<t$, then the training sample $x_{j,train}$ is passed to next step.

3.2.2 Meta-Feature Extraction

In this phase of meta-training, meta-features are extracted for each training sample. Different sets of meta-features can be extracted. Each feature set is used to calculate the competence level of the base classifier. Different property about the behavior of the base classifier is measured by each feature set. For the measurement of the meta-features, region of competence is defined for each training sample. Region of competence is defined by using K-Means in which the all the samples in the cluster in which the sample lies is in its region of competence. For each training sample $x_{j,train}$ region of competence represented by $\theta_j = \{x_1, x_2, \ldots x_k\}$ is calculated and defined in T by using K-Means algorithm. Then the output profile, $\tilde{x}_{j,train}$ is formed from $x_{j,\ train}$, that shows the decision of classifier c_i for $x_{j,train}$.The closest output profiles are chosen for each $x_{j,train}$ and each output profile has a label $w_{i,k}$. After that for each $c_i \in C$, following meta-features are calculated.

- **Neighbors' Classification - f_1:** First, a vector of zeroes having size k is created. After that for each x_k, having region of competence θ_j, if c_i accurately identifies x_k then kth position of above created vector is set to *1*.
- **Posterior Probability - f_2:** First, a vector of zeroes having size k is created. After that for each x_k, the posterior probability of c_i is calculated and inserted into kth position.
- **Accuracy - f_3:** The accuracy of c_i on the region of competence θ_j is calculated.

Criterion and paradigm for meta-features is represented in Table 3.2-1. In the end of the process a meta-vector of meta-features i-e $V_i = \{f_1, f_2, f_3\}$ is obtained. If c_i correctly predicts the label of x_k, then the class label of V_i is set to *1* else to *0*. Meta-features dataset T^* is used to store V_i that will be used training the meta-classifier *M*. Meta-Feature vector is represented in Figure 3.2-1.

Table 3.2-1. Criterion and Paradigm for Meta-Features

Meta-Feature	Criterion	Paradigm
f_1	Local accuracy of a classifier in competence region.	Accuracy of a classifier over local region
f_2	Consensus extent in competence region	Consensus of a classifier
$f3$	Overall accuracy of the classifier in competence region	Local region accuracy

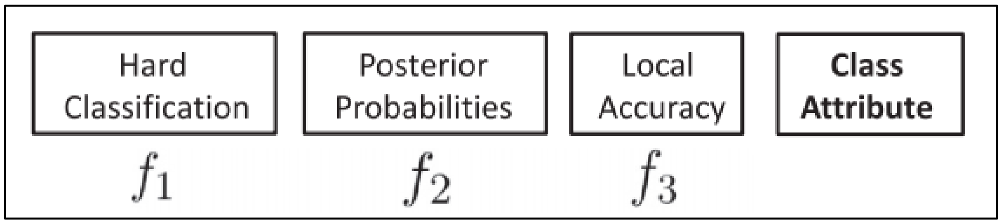


Figure 3.2-1: Meta-Feature Vector that Predicts Base Classifier Behavior

3.2.3 Training

The last step is to train *M*. The dataset *T** is divided into partitions i-e *75%* as training data and *25%* as testing data. The training phase of META-DES is improved by considering four models i-e SVM, Random Forest, Naïve Bayes and NN for the meta-classifier *M*. These models are selected because they are ranked as best models according to the study [47]. The attributes of meta-classifiers are defined in Table 3.2-2.

Table 3.2-2. Attributes of the Meta-Classifiers

Sr. No.	Classifier Name	Attributes
1.	SVM	Gamma='auto'
2.	Random Forest	N Estimators=100 Maximum Depth =10 Random State=0

3.	Logistic Regression	Max Iterations = 100
		c = 1.0
		Solver = ‘warn’

4.	MLP Classifier	Solver='lbfgs'
		Alpha=1e^{-5}
		Hidden Layer Sizes=(100, 100)
		Random State=0

3.3 Generalization Phase

When a test sample $x_{j,Test}$ arrives, its output profile $\tilde{x}_{j,Test}$ and region of competence are computed by using selection dataset *DSEL* and the closest output profiles for $x_{j,Test}$ are selected. After that for each c_i meta-feature extraction process is called that will return a vector V_i of meta-features. The V_i is used as an input for each M model, that will compute support s_i as the competence level for each c_i.

A support μ will be set as a threshold for the selection of the c_i. If support s_i of classifier ci is greater than μ i-e $s_i > \mu$, then classifier ci will be selected as "competent" by M. An Ensemble E is formed by selecting classifiers that are considered as "competent" by all models of M. If a model produces output "competent" for c_i, it will earn one vote.

After the formation of Ensemble E, weights will be assigned to the selected classifiers based on their competence level such that classifier having higher level of competence will get more weight and have more effect on the final prediction of output. Weight of the c_i is the number of votes c_i earned. Final output will be then obtained by using weighted voting scheme.

Hence, the performance and accuracy is increased by using proposed framework as we have used four different models for meta-classifier and only those classifiers are selected for ensemble that are considered as "competent" by all models of meta-classifier. After the ensemble is formed weights are assigned to classifiers based on their competence level i-e number of votes a classifier earned. Classifier having more competence level will have more weight and higher impact on output prediction. This will increase the both accuracy and performance of the system.

Chapter 4

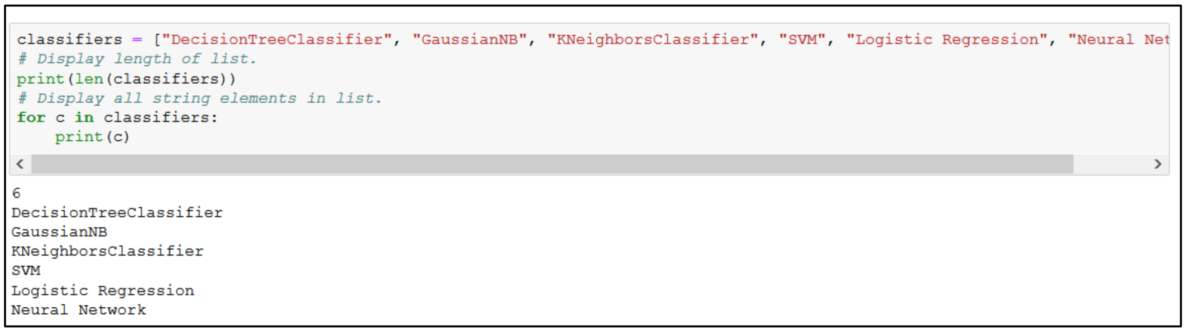
Implementation and Experiments

CHAPTER 4: IMPLEMENTATION AND EXPERIMENTS

4.1 Proposed Framework

The proposed framework is written in python language and IDE Anaconda 2018 is used. Following steps define the working of algorithm.

1. A pool of classifier was defined that contains all weak classifiers. These classifiers are used as base classifiers.

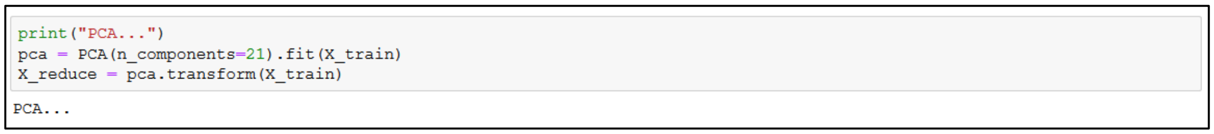


```
classifiers = ["DecisionTreeClassifier", "GaussianNB", "KNeighborsClassifier", "SVM", "Logistic Regression", "Neural Net
# Display length of list.
print(len(classifiers))
# Display all string elements in list.
for c in classifiers:
    print(c)
```

```
6
DecisionTreeClassifier
GaussianNB
KNeighborsClassifier
SVM
Logistic Regression
Neural Network
```

Figure 4.1-1: Defining Pool of Classifiers

2. Principle Component Analysis (PCA) was applied on dataset after data cleaning for pre-processing.

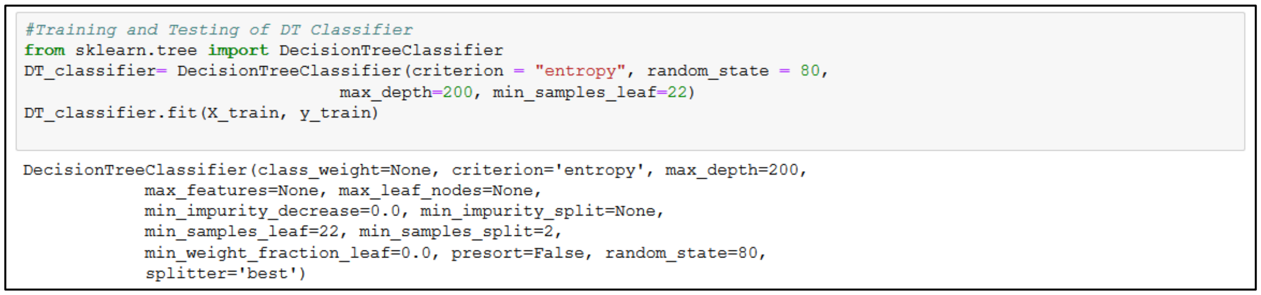


```
print("PCA...")
pca = PCA(n_components=21).fit(X_train)
X_reduce = pca.transform(X_train)
```

```
PCA...
```

Figure 4.1-2: Pre-Processing of Dataset using PCA

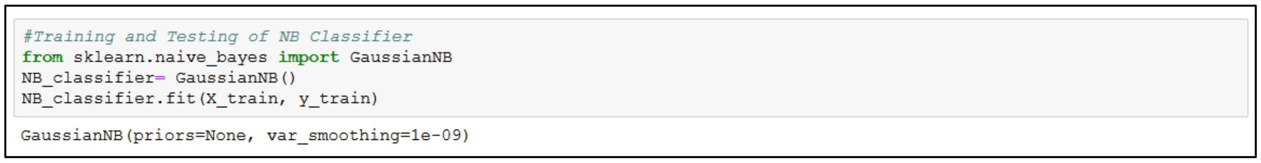
3. All the classifiers in the pool were trained on the training data.



```
#Training and Testing of DT Classifier
from sklearn.tree import DecisionTreeClassifier
DT_classifier= DecisionTreeClassifier(criterion = "entropy", random_state = 80,
                                      max_depth=200, min_samples_leaf=22)
DT_classifier.fit(X_train, y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=200,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=22, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=80,
            splitter='best')
```

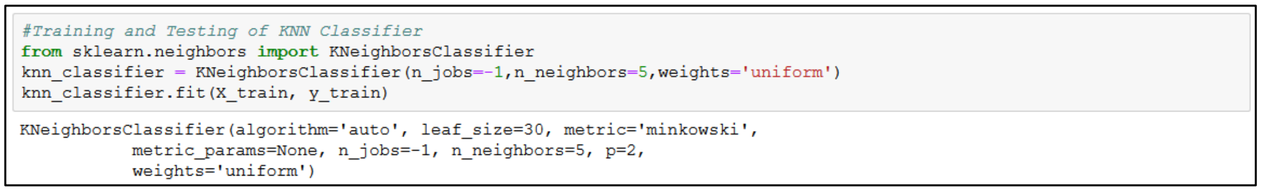
Figure 4.1-3: Training of Decision Tree



```
#Training and Testing of NB Classifier
from sklearn.naive_bayes import GaussianNB
NB_classifier= GaussianNB()
NB_classifier.fit(X_train, y_train)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

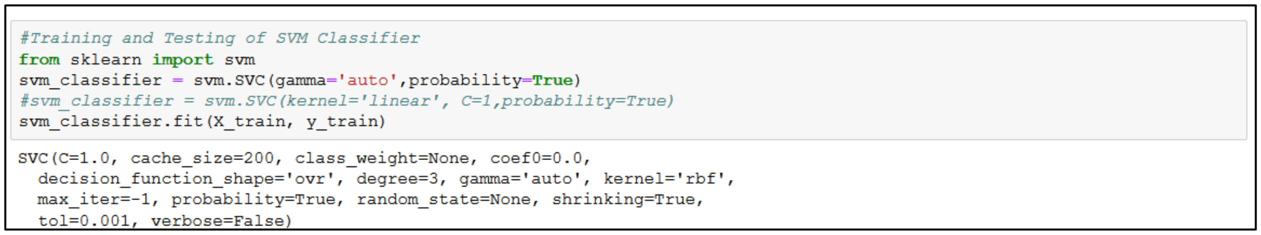
Figure 4.1-4: Training of Naïve Bayes



```
#Training and Testing of KNN Classifier
from sklearn.neighbors import KNeighborsClassifier
knn_classifier = KNeighborsClassifier(n_jobs=-1,n_neighbors=5,weights='uniform')
knn_classifier.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=-1, n_neighbors=5, p=2,
           weights='uniform')
```

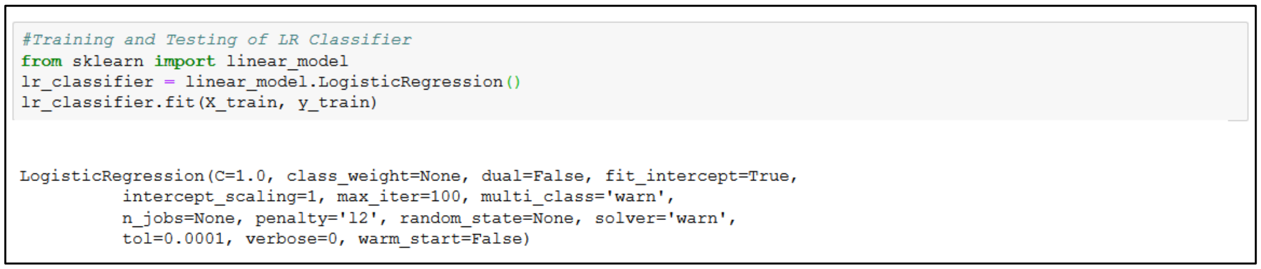
Figure 4.1-5: Training of K-Nearest Neighbor



```
#Training and Testing of SVM Classifier
from sklearn import svm
svm_classifier = svm.SVC(gamma='auto',probability=True)
#svm_classifier = svm.SVC(kernel='linear', C=1,probability=True)
svm_classifier.fit(X_train, y_train)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=True, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

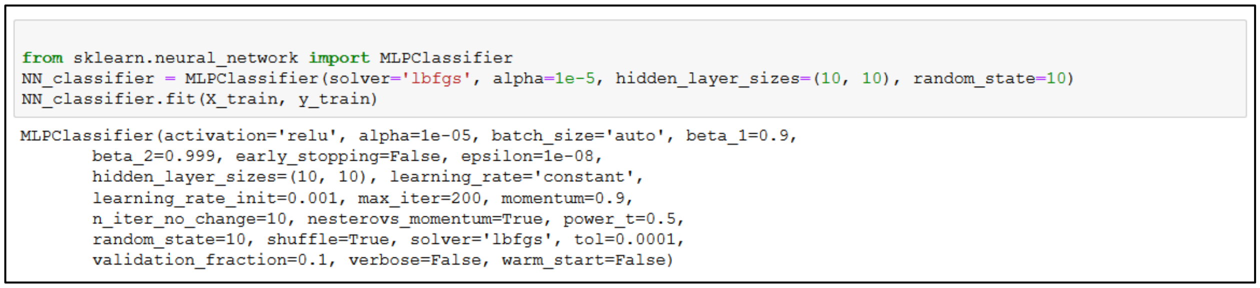
Figure 4.1-6: Training of Support Vector Machine



```
#Training and Testing of LR Classifier
from sklearn import linear_model
lr_classifier = linear_model.LogisticRegression()
lr_classifier.fit(X_train, y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

Figure 4.1-7: Training of Logistic Regression

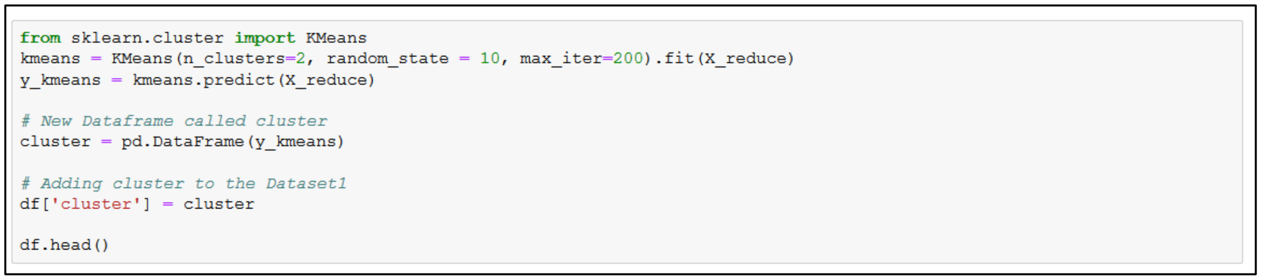


```
from sklearn.neural_network import MLPClassifier
NN_classifier = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(10, 10), random_state=10)
NN_classifier.fit(X_train, y_train)
```

```
MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(10, 10), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=10, shuffle=True, solver='lbfgs', tol=0.0001,
       validation_fraction=0.1, verbose=False, warm_start=False)
```

Figure 4.1-8: Training of Neural Network

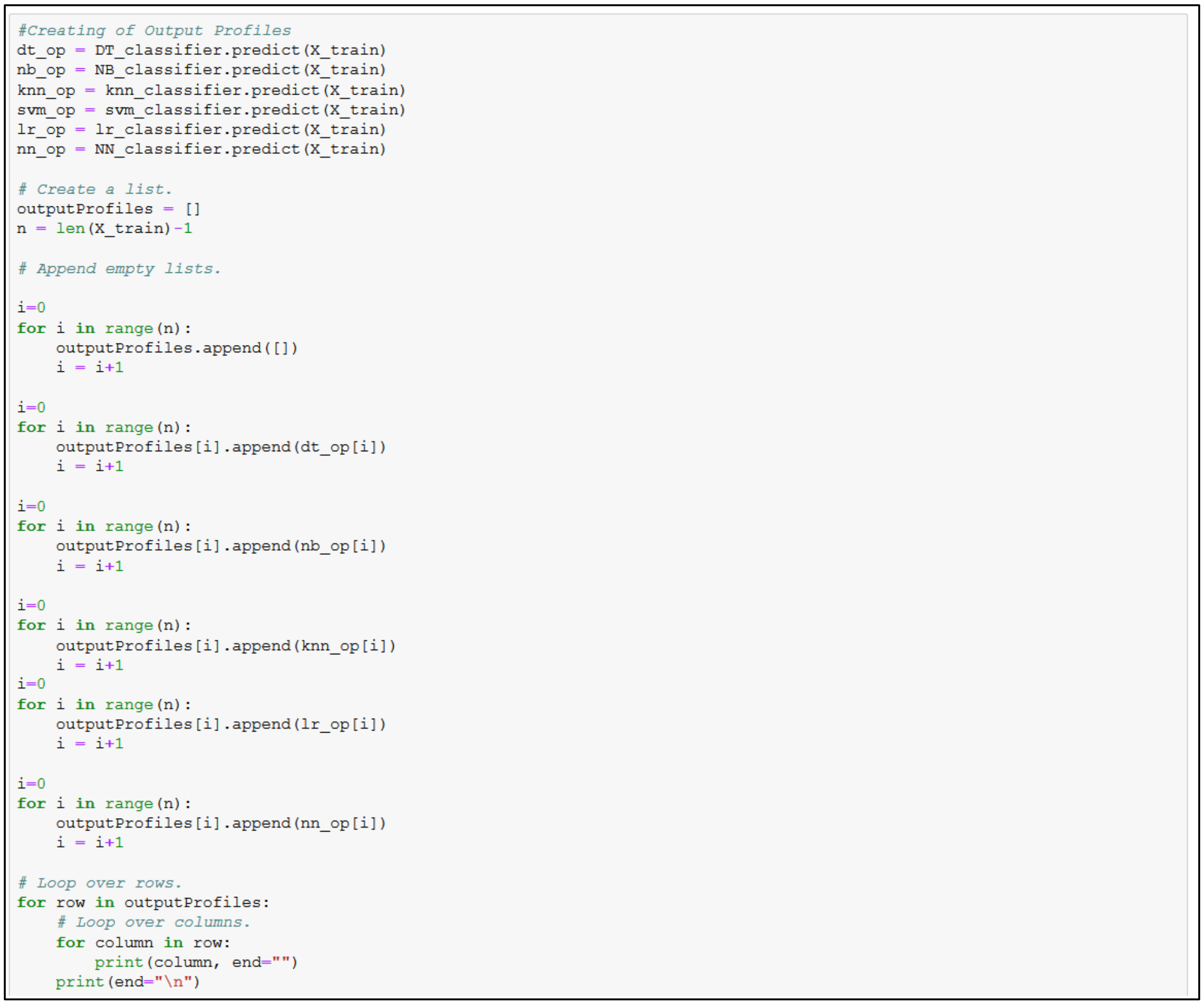
4. K-Means is used to make clusters of training instances and these clusters are used to find out the region of competence of each instance.



```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2, random_state = 10, max_iter=200).fit(X_reduce)
y_kmeans = kmeans.predict(X_reduce)

# New Dataframe called cluster
cluster = pd.DataFrame(y_kmeans)

# Adding cluster to the Dataset1
df['cluster'] = cluster

df.head()
```

Figure 4.1-9: Making Clusters by Using K-Means

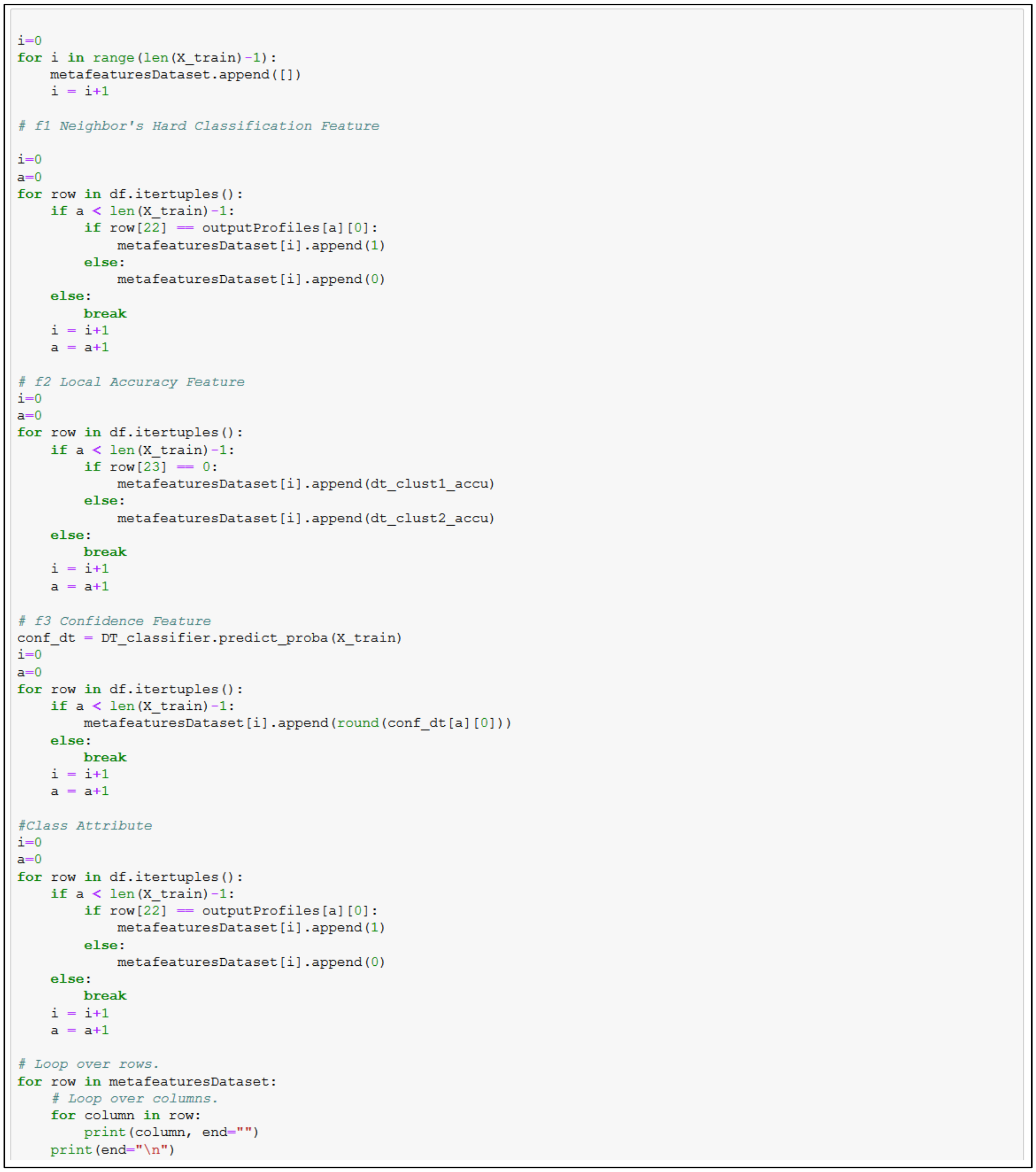
5. Output profile of each training instances is formed and stored in a vector that contains the decision of each classifier in pool for that instance.



```
#Creating of Output Profiles
dt_op = DT_classifier.predict(X_train)
nb_op = NB_classifier.predict(X_train)
knn_op = knn_classifier.predict(X_train)
svm_op = svm_classifier.predict(X_train)
lr_op = lr_classifier.predict(X_train)
nn_op = NN_classifier.predict(X_train)

# Create a list.
outputProfiles = []
n = len(X_train)-1

# Append empty lists.

i=0
for i in range(n):
    outputProfiles.append([])
    i = i+1

i=0
for i in range(n):
    outputProfiles[i].append(dt_op[i])
    i = i+1

i=0
for i in range(n):
    outputProfiles[i].append(nb_op[i])
    i = i+1

i=0
for i in range(n):
    outputProfiles[i].append(knn_op[i])
    i = i+1
i=0
for i in range(n):
    outputProfiles[i].append(lr_op[i])
    i = i+1

i=0
for i in range(n):
    outputProfiles[i].append(nn_op[i])
    i = i+1

# Loop over rows.
for row in outputProfiles:
    # Loop over columns.
    for column in row:
        print(column, end="")
    print(end="\n")
```

Figure 4.1-10: Method for Forming Output Profile of Instances

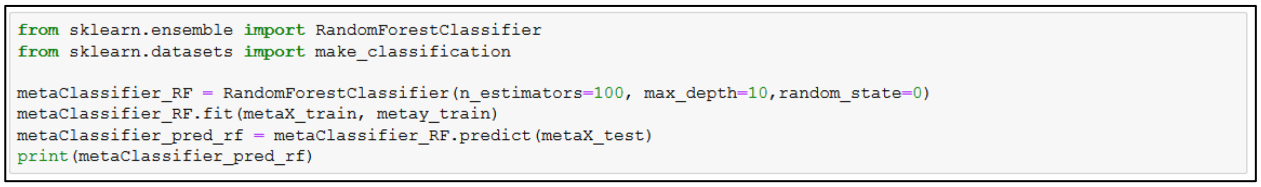
6. After that meta-features are calculated for each instance in training set and a meta-features dataset is formed.



```
i=0
for i in range(len(X_train)-1):
    metafeaturesDataset.append([])
    i = i+1

# f1 Neighbor's Hard Classification Feature

i=0
a=0
for row in df.itertuples():
    if a < len(X_train)-1:
        if row[22] == outputProfiles[a][0]:
            metafeaturesDataset[i].append(1)
        else:
            metafeaturesDataset[i].append(0)
    else:
        break
    i = i+1
    a = a+1

# f2 Local Accuracy Feature
i=0
a=0
for row in df.itertuples():
    if a < len(X_train)-1:
        if row[23] == 0:
            metafeaturesDataset[i].append(dt_clust1_accu)
        else:
            metafeaturesDataset[i].append(dt_clust2_accu)
    else:
        break
    i = i+1
    a = a+1

# f3 Confidence Feature
conf_dt = DT_classifier.predict_proba(X_train)
i=0
a=0
for row in df.itertuples():
    if a < len(X_train)-1:
        metafeaturesDataset[i].append(round(conf_dt[a][0]))
    else:
        break
    i = i+1
    a = a+1

#Class Attribute
i=0
a=0
for row in df.itertuples():
    if a < len(X_train)-1:
        if row[22] == outputProfiles[a][0]:
            metafeaturesDataset[i].append(1)
        else:
            metafeaturesDataset[i].append(0)
    else:
        break
    i = i+1
    a = a+1

# Loop over rows.
for row in metafeaturesDataset:
    # Loop over columns.
    for column in row:
        print(column, end="")
    print(end="\n")
```

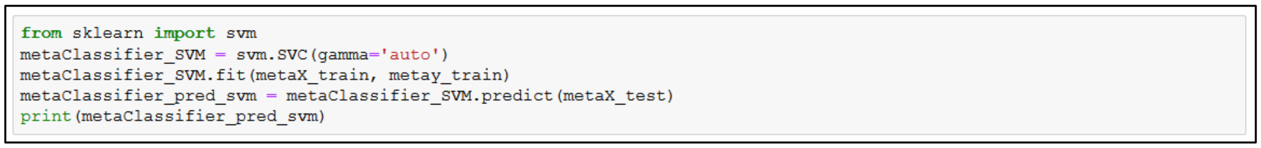
Figure 4.1-11: Method for Calculating Meta-Features

7. Meta-Features dataset is then used to train the meta-classifiers.



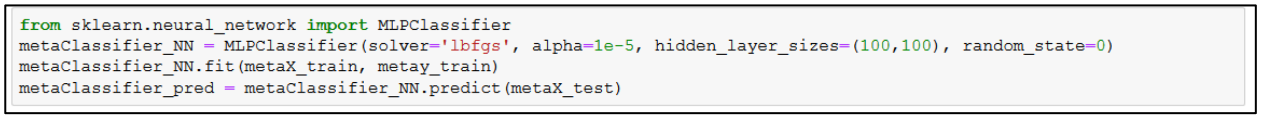
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification

metaClassifier_RF = RandomForestClassifier(n_estimators=100, max_depth=10,random_state=0)
metaClassifier_RF.fit(metaX_train, metay_train)
metaClassifier_pred_rf = metaClassifier_RF.predict(metaX_test)
print(metaClassifier_pred_rf)
```

Figure 4.1-12: Training of Meta-Classifier: Random Forest



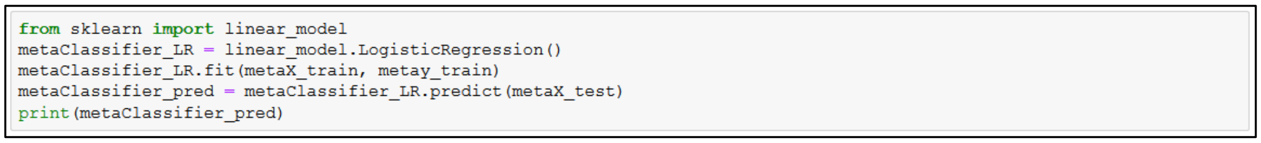
```
from sklearn import svm
metaClassifier_SVM = svm.SVC(gamma='auto')
metaClassifier_SVM.fit(metaX_train, metay_train)
metaClassifier_pred_svm = metaClassifier_SVM.predict(metaX_test)
print(metaClassifier_pred_svm)
```

Figure 4.1-13: Training of Meta-Classifier: Support Vector Machine



```
from sklearn.neural_network import MLPClassifier
metaClassifier_NN = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(100,100), random_state=0)
metaClassifier_NN.fit(metaX_train, metay_train)
metaClassifier_pred = metaClassifier_NN.predict(metaX_test)
```

Figure 4.1-14: Training of Meta-Classifier: Multilayer Perceptron

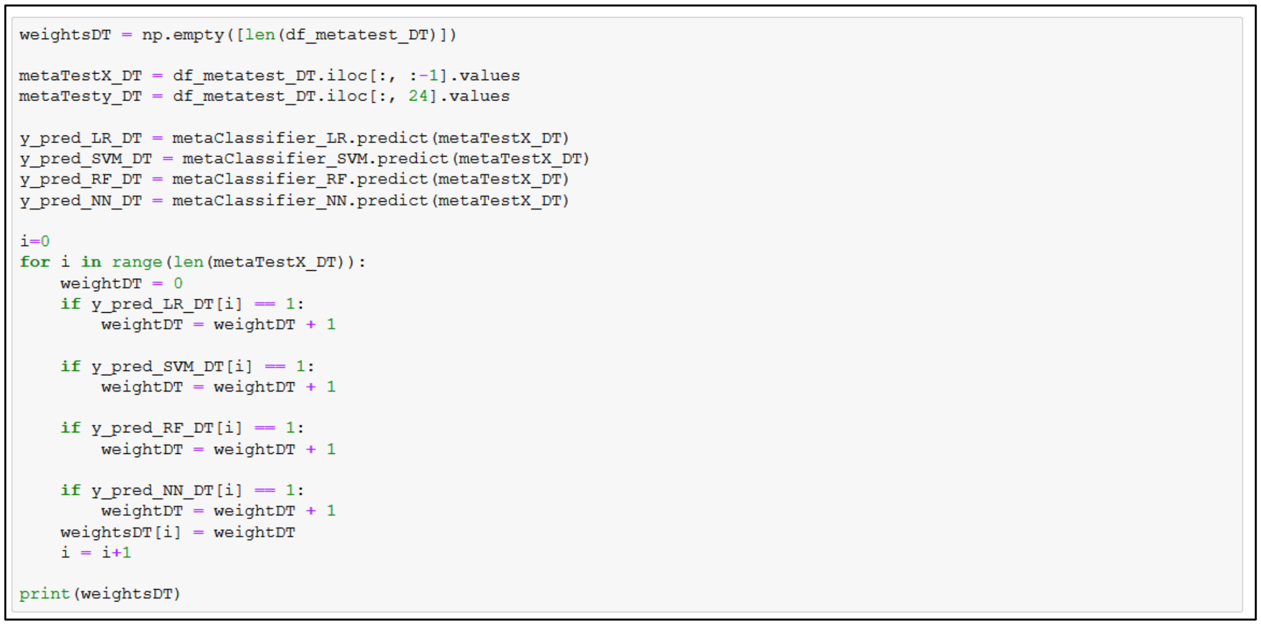


```
from sklearn import linear_model
metaClassifier_LR = linear_model.LogisticRegression()
metaClassifier_LR.fit(metaX_train, metay_train)
metaClassifier_pred = metaClassifier_LR.predict(metaX_test)
print(metaClassifier_pred)
```

Figure 4.1-15: Training of Meta-Classifier: Logistic Regression

8. Then for each instance of test set steps from step 3 are repeated.

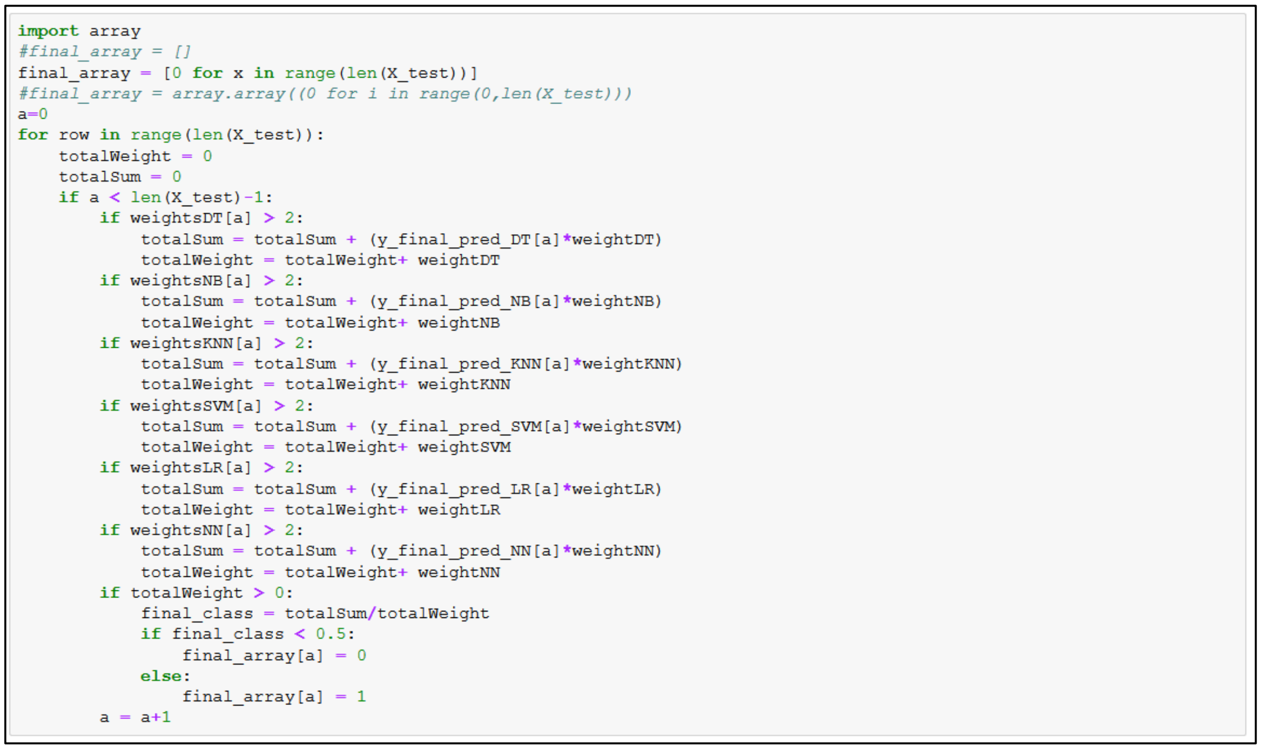
9. Meta-Features vector for each test instance is passed to the meta-classifiers that will compute the compete level for each classifier and weights are dynamically assigned to the competent classifiers.



```
weightsDT = np.empty([len(df_metatest_DT)])

metaTestX_DT = df_metatest_DT.iloc[:, :-1].values
metaTesty_DT = df_metatest_DT.iloc[:, 24].values

y_pred_LR_DT = metaClassifier_LR.predict(metaTestX_DT)
y_pred_SVM_DT = metaClassifier_SVM.predict(metaTestX_DT)
y_pred_RF_DT = metaClassifier_RF.predict(metaTestX_DT)
y_pred_NN_DT = metaClassifier_NN.predict(metaTestX_DT)

i=0
for i in range(len(metaTestX_DT)):
    weightDT = 0
    if y_pred_LR_DT[i] == 1:
        weightDT = weightDT + 1

    if y_pred_SVM_DT[i] == 1:
        weightDT = weightDT + 1

    if y_pred_RF_DT[i] == 1:
        weightDT = weightDT + 1

    if y_pred_NN_DT[i] == 1:
        weightDT = weightDT + 1
    weightsDT[i] = weightDT
    i = i+1

print(weightsDT)
```

Figure 4.1-16: Method for Dynamically Calculating Weights for Classifiers

10. After that using weighting voting approach, final class label for that instance is computed.



```
import array
#final_array = []
final_array = [0 for x in range(len(X_test))]
#final_array = array.array((0 for i in range(0,len(X_test)))
a=0
for row in range(len(X_test)):
    totalWeight = 0
    totalSum = 0
    if a < len(X_test)-1:
        if weightsDT[a] > 2:
            totalSum = totalSum + (y_final_pred_DT[a]*weightDT)
            totalWeight = totalWeight+ weightDT
        if weightsNB[a] > 2:
            totalSum = totalSum + (y_final_pred_NB[a]*weightNB)
            totalWeight = totalWeight+ weightNB
        if weightsKNN[a] > 2:
            totalSum = totalSum + (y_final_pred_KNN[a]*weightKNN)
            totalWeight = totalWeight+ weightKNN
        if weightsSVM[a] > 2:
            totalSum = totalSum + (y_final_pred_SVM[a]*weightSVM)
            totalWeight = totalWeight+ weightSVM
        if weightsLR[a] > 2:
            totalSum = totalSum + (y_final_pred_LR[a]*weightLR)
            totalWeight = totalWeight+ weightLR
        if weightsNN[a] > 2:
            totalSum = totalSum + (y_final_pred_NN[a]*weightNN)
            totalWeight = totalWeight+ weightNN
        if totalWeight > 0:
            final_class = totalSum/totalWeight
            if final_class < 0.5:
                final_array[a] = 0
            else:
                final_array[a] = 1
        a = a+1
```

Figure 4.1-17: Method for Predicting Class Labels

4.2 Datasets

The proposed method is tested on 10 datasets coming from the UCI machine learning repository, Knowledge Extraction based on Evolutionary Learning (KEEL) repository and Kaggle. Both ill-de*fi*ned problems, such as, Heart, Blood Transfusion and Breast Cancer as well as larger databases, such as, Phoneme, Thyroid, Sonar, Diabetes and cardiotocography are considered.

4.2.1 Heart Dataset

Heart dataset is the most commonly used dataset for experimentation. The dataset is collected from Kaggle [48]. It contains 76 attributes, but experiment is conducted on the subset of 14 of them. Experiments have revealed the presence of heart disease by predicting values 0 or 1 where 0 shows the absence of disease. Meta-Data of heart dataset is described in Table 4.2-1.

Table 4.2-1. Description of Heart Dataset

Data Set Characteristics:	Multivariate
Attribute Characteristics:	Categorical, Integer, Real
Associated Tasks:	Classification
Number of Instances:	303
Number of Attributes:	15
Missing Values?	Yes
Area:	Life
No. of Classes	2

4.2.2 Diabetes Dataset

Diabetes dataset is collected from Kaggle [49]. It contains 9 attributes out of which 8 are independent medical variables while 1 is dependent variable that predict the outcome. The predicted outcome is either 0 or 1. Total Instances are 768 out of which 268 are 1 and other are 0. Meta-Data of diabetes dataset is described in Table 4.2-2.

Table 4.2-2. Description of Diabetes Dataset

Data Set Characteristics:	Multivariate
Attribute Characteristics:	Categorical, Integer, Real
Associated Tasks:	Classification
Number of Instances:	768
Number of Attributes:	9
Missing Values?	No
Area:	Life
No. of Classes	2

4.2.3 Blood Transfusion Dataset

Blood Transfusion dataset is collected from UCI [50]. It contains 5 attributes out which 1 is dependent variable that predict the outcome.Dataset contains the data of 748 donors, each one included R (Recency - months since last donation), F (Frequency - total number of donation), M (Monetary - total blood donated in c.c.), T (Time - months since first donation). The predicted outcome is either 1 or 0 where 1 stands for donating blood and 0 stands for not donating blood. Meta-Data of blood transfusion dataset is described in Table 4.2-3.

Table 4.2-3. Description of Blood Transfusion Dataset

Data Set Characteristics:	Multivariate
Attribute Characteristics:	Real
Associated Tasks:	Classification

Number of Instances:	748
Number of Attributes:	5
Missing Values?	No
Area:	Business
No. of Classes	2

4.2.4 Sonar Dataset

Sonar dataset is collected from Kaggle [51]. It contains 61 attributes out which 1 is dependent variable that predict the outcome. Dataset contains the data of 208 instances. The predicted outcome is either 1 or 2. Meta-Data of sonar dataset is described in Table 4.2-4.

Table 4.2-4. Description of Sonar Dataset

Data Set Characteristics:	Multivariate
Attribute Characteristics:	Real
Associated Tasks:	Classification
Number of Instances:	208
Number of Attributes:	61
Missing Values?	No
Area:	Physical
No. of Classes	2

4.2.5 Cardiotocography (CTG) Dataset

Cardiotocography dataset is collected from UCI [52] that contains the measurement of heart rate and uterine contractions from cardiotocograms. It contains 23 attributes out which 1 is dependent variable that predict the outcome. Dataset contains the data of2126 instances. The predicted outcome is either 1, 2 or 3. Meta-Data of cardiotocography dataset is described in Table 4.2-5.

Table 4.2-5. Description of Cardiotocography Dataset

Data Set Characteristics:	Multivariate
Attribute Characteristics:	Real
Associated Tasks:	Classification
Number of Instances:	2126
Number of Attributes:	23
Missing Values?	No
Area:	Life
No. of Classes	3

4.2.6 Phoneme Dataset

Phoneme dataset is collected from KEEL [53]. The dataset helps to distinguish between nasal and oral sounds by assigning class 0 and 1 respectively. It contains 5 attributes out which 1 is dependent variable that predict the outcome. Dataset contains the data of 5404 instances. The predicted outcome is either 0 or 1. Meta-Data of phoneme dataset is described in Table 4.2-6.

Table 4.2-6. Description of Phoneme Dataset

Data Set Characteristics:	Multivariate
Attribute Characteristics:	Real
Associated Tasks:	Classification
Number of Instances:	5404

Number of Attributes:	5
Missing Values?	No
Area:	Life
No. of Classes	2

4.2.7 Breast Cancer Dataset

Breast cancer dataset is collected from UCI [54] The dataset helps to distinguish between malignant and benign by assigning class M and B respectively. It contains 32 attributes out which 1 is dependent variable that predict the outcome. Dataset contains the data of 569 instances. Meta-Data of breast cancer dataset is described in Table 4.2-7.

Table 4.2-7. Description of Breast Cancer Dataset

Data Set Characteristics:	Multivariate
Attribute Characteristics:	Real
Associated Tasks:	Classification
Number of Instances:	569
Number of Attributes:	23
Missing Values?	No
Area:	Life
No. of Classes	2

4.2.8 Thyroid Dataset

Thyroid dataset is collected from KEEL [55]. The dataset helps to distinguish between normal, hyperthyroidism and hypothyroidism by assigning class 1, 2 and 3 respectively. It contains 21 attributes out which 1 is dependent variable that predict the outcome. Dataset contains the data of7200 instances. The predicted outcome is either 1, 2 or 3. Meta-Data of thyroid dataset is described in Table 4.2-8.

Table 4.2-8. Description of Thyroid Dataset

Data Set Characteristics:	Multivariate
Attribute Characteristics:	Real
Associated Tasks:	Classification
Number of Instances:	7200
Number of Attributes:	21
Missing Values?	No
Area:	Life
No. of Classes	3

The important features of datasets are described in Table 4.2-9.

Table 4.2-9. Important Features of Datasets

Dataset	No. of Instances	No. of Attributes	No. of Classes	Source
Heart	303	15	2	Kaggle
Diabetes	768	9	2	Kaggle
Blood Transfusion	748	5	2	UCI
Sonar	208	61	2	Kaggle

Cardiotocography	2126	23	3	UCI
Phoneme	5404	5	2	KEEL
Breast Cancer	569	23	2	UCI
Thyroid	7200	21	3	KEEL

4.3 Training and Testing

For training and testing we used the method of 10-fold cross validation [56]. As shown in Figure 4.3-1, it consists of 10 experiments, each time taking different sets for training and testing from the input dataset. In this process:

1. The input dataset is divided into 10 equal subsets.
2. From these10 subsets, 9 are used for training and 1 is used for testing.
3. Process is repeated 10 times, each time taking different subset for testing.
4. Final performance is evaluated by taking an average of results.

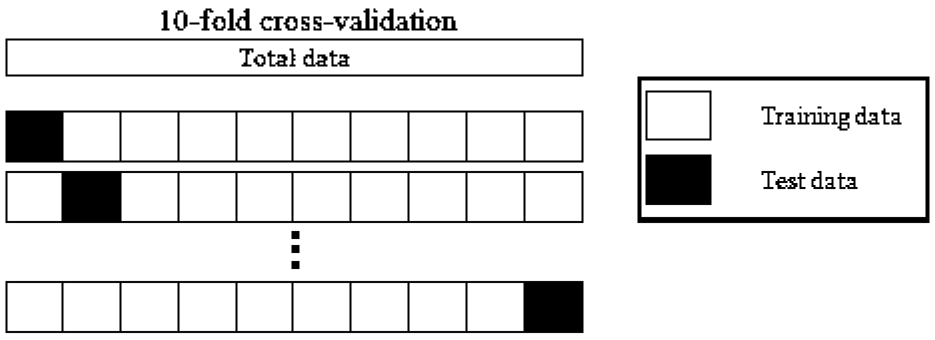


Figure 4.3-1: 10-Fold Cross Validation

4.4 Evaluation Measures

In order to evaluate the performance of proposed methodology, we have chosen some standard measures that includes accuracy, precision, recall and F-measure. Here are the mathematical formulas for these parameters.

4.5 Confusion Matrix

The results predicted by the classifiers are presented in a tabular form that separates the correct prediction of class from incorrect predictions. This is called confusion matrix [57]. It tells the correct and incorrect predictions. Other performance measures like accuracy, precision, recall and F-measure can be calculated by using this matrix. Confusion matrix is represented in Table 4.5-1. The four cells of this matrix show true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN).

TP = Number of predictions that are correctly classified by the classifier as positive.

TN = Number of predictions that are correctly classified by the classifier as negative.

FP = Number of predictions that are incorrectly classified by the classifier as positive.

FN = Number of predictions that are incorrectly classified by the classifier as negative.

Table 4.5-1. The Confusion Matrix

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

4.5.1 Accuracy

Accuracy is the percentage of instances that are correctly classified divided by the total number of instances [58]. It can be given as:

$$Accuracy = \frac{No.of\ Correctly\ classified\ Instances}{Total\ No.of\ Instances} \text{x}100$$

Confusion matrix can be used to find accuracy by using TP and TN that defines correctly classified instances and sum of all cells of confusion matrix that defines the total instances. It can be given as:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \times 100$$

4.5.2 Precision

Precision is the percentage of number of correct predictions by the total predictions [59]. It calculates the fraction of instances that are truly positive. In terms of probability, precision is the probability that an instance is correctly classified. In terms of confusion matrix, it can be measured as:

$$\text{Precision} = \frac{TP}{TP+FP}$$

4.5.3 Recall

Recall is the measure of the fraction of positive instances that were correctly classified [60]. In terms of confusion matrix, it can be measured as:

$$\text{Recall} = \frac{TP}{TP+FN}$$

4.5.4 F-Measure

F-Measure is the harmonic mean of precision and recall [61]. It provides a balance between precision and recall and uses both to calculate a performance measure. Its formula is given as:

$$\text{F- Measure} = \frac{2*Precision*Recall}{Precision+Recall}$$

For measuring the classifier performance, accuracy alone is not an appropriate measure. E.g consider a dataset having a total of 100 instances out of which 95 are negative and 5 are positive. If a classifier classifies all the instances as negative, the accuracy of the classifier will be 95%, despite that no positive instance is correctly classified. Hence, other performance measures overcome this limitation by calculating a fraction to measure TP.

4.6 Parameter Setting

The performance of our proposed framework is determined by two parameters i-e Number of clusters required by K-Means and Number of weights required by the classifier to be selected to select for ensemble. Experiments were performed on different parameter values and for simplicity those parameters values are selected in which the model performs best.

1. Number of Clusters required by K-Means is set to 2.
2. Number of weights required by the classifier to be selected to select for ensemble is set to

4.7 Experiment Protocol

Experiments were conducted by using 10 replications. Dataset was divided into 90% training and 10% testing during each replication. Weak classifiers were selected for the pool of classifiers. Considering weak classifiers as base classifiers in DES helps to achieve good results.

Chapter 5

Results and Comparisons

CHAPTER 5: RESULTS AND COMPARISONS

5.1 Classification Results

Classification was performed by using our proposed methodology on each of the dataset. Results were evaluated on the bases of accuracy, precision, recall and F-measure. Experiments were performed on the heart dataset using 10-Fold Cross validation. During each iteration dataset is divided into 90% training and 10% testing data.

5.1.1 Classification Results on Heart Dataset

Classification results of heart dataset is represented in Table 5.1-1.

Table 5.1-1. Classification Results on Heart Dataset

Heart Dataset	
Accuracy	86.18%
Precision	0.97
Recall	0.90
F-measure	0.94

5.1.2 Classification Results on Diabetes Dataset

Classification results of diabetes dataset is represented in Table 5.1-2.

Table 5.1-2. Classification Results on Diabetes Dataset

Diabetes Dataset	
Accuracy	87.6%
Precision	0.91
Recall	0.89
F-measure	0.88

5.1.3 Classification Results on Blood Transfusion Dataset

Classification results of blood transfusion dataset is represented in Table 5.1-3.

Table 5.1-3. Classification Results on Blood Transfusion Dataset

Blood Transfusion Dataset	
Accuracy	90.67%
Precision	0.93
Recall	0.91
F-measure	0.90

5.1.4 Classification Results on Sonar Dataset

Classification results of sonar dataset is represented in Table 5.1-4.

Table 5.1-4. Classification Results of Sonar Dataset

Sonar Dataset	
Accuracy	90.61%
Precision	0.98
Recall	0.91
F-measure	0.94

5.1.5 Classification Results on Cardiotocography (CTG) Dataset

Classification results of cardiotocography dataset is represented in Table 5.1-5.

Table 5.1-5. Classification Results of Cardiotocography Dataset

Cardiotocography (CTG) Dataset	
Accuracy	85.41%
Precision	0.91
Recall	0.86
F-measure	0.84

5.1.6 Classification Results on Phoneme Dataset

Classification results of phoneme dataset is represented in Table 5.1-6.

Table 5.1-6. Classification Results of Phoneme Dataset

Phoneme Dataset	
Accuracy	87.20%
Precision	0.89
Recall	0.87
F-measure	0.86

5.1.7 Classification Results on Breast Cancer Dataset

Classification results of breast cancer dataset is represented in Table 5.1-7.

Table 5.1-7. Classification Results of Breast Cancer Dataset

Breast Cancer Dataset	
Accuracy	98.21%
Precision	0.96
Recall	0.95
F-measure	0.96

5.1.8 Classification Results on Thyroid Dataset

Classification results of thyroid dataset is represented in Table 5.1-8.

Table 5.1-8. Classification Results of Thyroid Dataset

Thyroid Dataset	
Accuracy	99.17%
Precision	0.99
Recall	0.99
F-measure	0.99

Summarized classification results are shown in Table 5.1-9.

Table 5.1-9. Summarized Classification Results

Datasets	**Accuracy**	**Precision**	**Recall**	**F-Measure**
Heart	86.18%	0.97	0.90	0.94
Diabetes	87.6%	0.91	0.89	0.88
Blood Transfusion	90.67%	0.89	0.91	0.90
Sonar	90.61%	0.88	0.91	0.94
CTG	85.41%	0.91	0.86	0.84
Phoneme	87.20%	0.89	0.87	0.86
Breast Cancer	98.21%	0.96	0.95	0.96
Thyroid	99.17%	0.99	0.99	0.99

5.2 Comparison with State of Art Techniques

We have evaluated our framework on the selected datasets and compared its results with state-of-art techniques in the literature. The techniques used in this process are described below.

5.2.1 Comparison with Dynamic Classifier Selection Techniques

Following dynamic classifier selection techniques were used for the evaluation of our proposed framework.

1. META-DES
2. META-DES.H
3. KNORA-UNION
4. DES-FA
5. Local Classifier Accuracy (LCA)
6. Overall Local Accuracy (OLA)
7. Modified Local Accuracy (MLA)
8. Multiple Classifier Behavior (MCB)
9. K-Nearest Output Profiles (KNOP)

Comparison of our proposed framework with dynamic classifier selection techniques is presented in Table 5.2-1.

Table 5.2-1. Comparison with State-of-Art Dynamic Classifier Selection Techniques

Dataset	**Proposed Framework**	**META-DES** [6]	**META-DES.H** [19]	**KNORA-U** [62] [63]	**DES-FA** [64] [65]	**LCA** [66] [67]	**OLA** [68] [69]	**MLA** [70]	**MCB** [71]	**KNOP** [72] [73]
Heart	**86.18**	84.80	85.62	83.82	83.82	85.29	85.29	86.76	83.82	83.82
Diabetes	**87.6**	79	78.80	76.60	73.95	73.95	73.95	77.08	76.56	73.42
Blood Transfusion	**90.67**	79.14	79.85	77.12	73.40	75.00	75.00	76.06	73.40	77.54
Sonar	**90.61**	80.55	82.06	76.69	78.52	76.51	74.52	76.91	76.56	75.72
CTG	**85.41**	84.62	86.89	85.71	86.27	86.65	86.65	86.27	85.71	86.02
Phoneme	**87.20**	80.35	82.68	78.92	79.06	78.84	78.84	64.94	73.37	78.92
Breast Cancer	**98.21**	97	97.02	97.18	97.88	97.88	97.88	95.77	97.18	95.42
Thyroid	**99.17**	96.78	96.99	95.95	95.37	95.95	95.95	94.79	95.95	95.95

Our proposed framework has achieved good results in comparison with other dynamic classifier selection techniques in state-of-art. We have considered eight datasets i-e Heart, Diabetes, Blood Transfusion, Sonar, CTG, Phoneme, Breast Cancer and Thyroid for experimentation. On all these eight datasets our proposed framework outperforms in comparison with state-of-art techniques.

Table 5.2-1 shows the comparison of our achieved accuracy with eight other dynamic classifier selection techniques evaluated on the selected datasets. Highest accuracy of *99.17%* was achieved with thyroid dataset while other techniques like META-DES, META-DES.H, KNORA-UNION, DES-FA, Local Classifier Accuracy (LCA) , Overall Local Accuracy (OLA), Modified Local Accuracy (MLA) , Multiple Classifier Behavior (MCB) , K-Nearest Output Profiles (KNOP) has achieved accuracy 96.78 , 96.99, 95.95, 95.37, 95.95, 95.95, 94.79, 95.95, 95.95 respectively. These results show that our proposed framework outperforms these techniques of each this dataset and on other selected datasets.

5.2.2 Comparison with Static Classifier Selection Techniques

Following static classifier selection techniques were used for the evaluation of our proposed framework.

1. Single-Best Classifier
2. Bagging
3. AdaBoost
4. Static Selection

Comparison of our proposed framework with static ensemble selection techniques is presented in Table 5.2-2.

Table 5.2-2. Comparison with State-of-Art Static Classifier Selection Techniques

Dataset	**Proposed Framework**	**Single Best** [74] [75]	**Bagging** [76] [77]	**AdaBoost** [77] [79]	**Static Selection** [80] [81]
Heart	**86.18**	80.26	82.50	81.61	82.05
Diabetes	**87.6**	73.57	73.28	72.52	72.86
Blood Transfusion	**90.67**	75.07	75.24	75.18	75.74
Sonar	**90.61**	78.21	76.66	74.95	79.03
CTG	**85.41**	84.21	84.54	83.06	83.06
Phoneme	**87.20**	75.87	72.60	75.90	72.70
Breast Cancer	**98.21**	97.04	96.35	98.18	96.83
Thyroid	**99.17**	95.15	95.25	96.01	96.24

Our proposed framework has achieved good results in comparison with other static classifier selection techniques in state-of-art. We have considered eight datasets i-e Heart, Diabetes, Blood Transfusion, Sonar, CTG, Phoneme, Breast Cancer and Thyroid for experimentation. On all these eight datasets our proposed framework outperforms in comparison with state-of-art techniques.

Table 5.2-2 shows the comparison of our achieved accuracy with four other static classifier selection techniques evaluated on the selected datasets. Highest accuracy of *99.17%* was achieved with thyroid dataset while other techniques like Single Best, Bagging, AdaBoost and Static Selection has achieved accuracy 95.15, 95.25, 96.01, 96.24 respectively. These results show that our proposed framework outperforms these techniques of each this dataset and on other selected datasets.

5.3 Results Evaluation

Results are evaluated on our datasets including Heart, Diabetes, Blood Transfusion, Sonar, CTG, Phoneme, Breast Cancer and Thyroid datasets. The results obtained shows that our proposed model outperforms existing models for dynamic classifier selection. It is also observed that use of multiple meta-classifier models and dynamic weighted voting approach for results aggregation has a positive effect on the prediction. Dynamic weights have been assigned to the base classifiers selected for an ensemble formation by meta-classifiers. Using dynamic weighted voting scheme helps us to deal with outliers and noisy samples as weights are assigned and most competent classifier will have higher impact in output prediction. We have used dynamic weighted voting in our experimentation that helps in generating better results.

The results of our proposed model along with the model to be compare on same dataset as our model are shown in the Figure 5.3-1. Results shows that our proposed ensemble model is the most accurate.

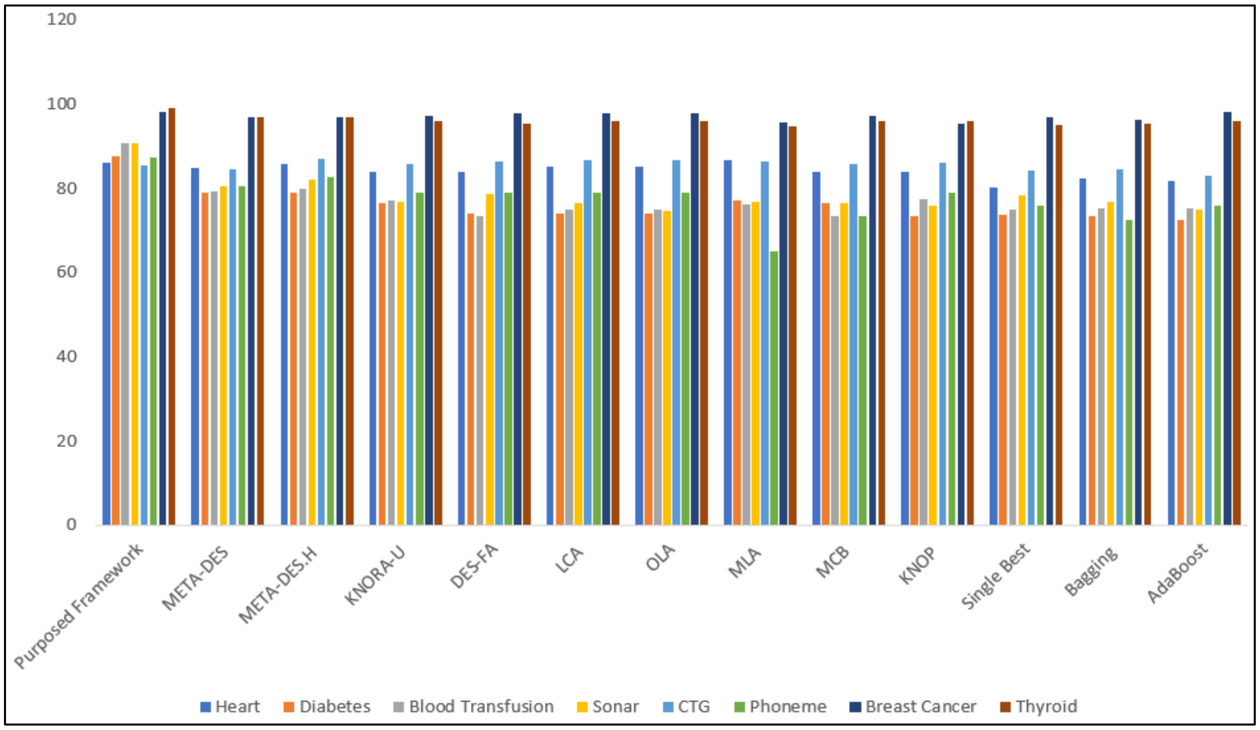


Figure 5.3-1: Comparison of Different Models Results

5.4 Discussion

Dynamic selection of classifier is a vast field of research and different models and frameworks are proposed in literature for improving accuracy of correctly predicting the class label. Machine learning has another technique called Ensemble Formation that has being explored for the task of dynamic classifier selection. These approaches perform better perform better than tradition machine learning approaches and models exists in literature as the output of weak models are combined to produce a strong model. Weighed voting approach has been used to aggregate the results of multiple classifiers. Existing models can be improved by improving architecture of the models and this motivation is the main reason to conduct this research and for proposing the ensemble framework.

Chapter 6

Conclusion and Future Work

CHAPTER 6: CONCLUSION AND FUTURE WORK

In this work, we have suggested some improvements in second step (training) and last step (generalization) of the DES framework. In training phase, four different models are used as meta-classifiers. While in generalization phase, we had used dynamic weighting scheme in which meta-classifiers will dynamically assign weights to selected competent classifiers based on their competence level and final decision will be aggregated using a weighting voting scheme. Hence, the classifiers having high competence level will have greater impact on the final prediction, and this will help us to deal with the case of noisy samples where all classifiers from pool will be considered as competent for an ensemble and incompetent classifiers will be having equal impact on final output as that of competent ones. Thus, modifications suggested in this paper altogether enhance performance and accuracy of the framework in contrast with other dynamic selection techniques in literature.

Future work on this framework involves:

- New set of meta-features can be defined that can help us better to find the competence of base classifier.
- Different training scenarios can be evaluated for the training of meta-classifier.
- Evaluating different techniques for generalization phase can further improve the accuracy of proposed solution.

References

[1] Woźniak, Michał, Manuel Graña, and Emilio Corchado. "A survey of multiple classifier systems as hybrid systems." *Information Fusion* 16 (2014): 3-17.

[2] S. B. Jr., R. Sabourin. and L. E. Oliveira. "Dynamic selection of classifiers as a comprehensive review;" Pattern Recognition, vol. 47, no. 11, pp. 3665 - 3680, 2014.

[3] A. H. R. Ko. R. Sabourin, and u. S. Britto, Jr., "From dynamic classifiers election to dynamic ensemble selection." Pall ern Recognition. vol. 41,pp. 1735-1748. May 2008.

[4] P. R. Caval in, R. Sabourin, and C. Y. Suen, "Dynamic selection approaches for multiple classifier systems." Neural Computing and Applications. vol. 22. no. 3-4, pp. 673-688, 2013.

[5] --, "Logid: An adaptive framework combining local and global incremental learning for dynamic selection of ensembles of HMMs, "Pattern Recognition, vol. 45, no. 9, pp. 3544-3556,2012.

[6] Cruz, Rafael MO, Robert Sabourin, George DC Cavalcanti, and Tsang Ing Ren. "META-DES: A dynamic ensemble selection framework using meta-learning." Pattern recognition 48, no. 5 (2015): 1925-1935.

[7] Woloszynski, Tomasz, Marek Kurzynski, Pawel Podsiadlo, and Gwidon W. Stachowiak. "A measure of competence based on random classification for dynamic ensemble selection." Information Fusion 13, no. 3 (2015): 207-213.

[8] Cruz, Rafael MO, Robert Sabourin, and George DC Cavalcanti. "Prototype selection for dynamic classifier and ensemble selection." Neural Computing and Applications 29, no. 2 (2018): 447-457.

[9] Mejri, Dhouha, Mohamed Limam, and Claus Weihs. "A new dynamic weighted majority control chart for data streams." Soft Computing 22, no. 2 (2018): 511-522.

[10] Brun, André L., Alceu S. Britto Jr, Luiz S. Oliveira, Fabricio Enembreck, and Robert Sabourin. "A framework for dynamic classifier selection oriented by the classification problem difficulty." Pattern Recognition 76 (2018): 175-190.

[11] Ye, Rui, and Qun Dai. "A Novel Greedy Randomized Dynamic Ensemble Selection Algorithm." Neural Processing Letters 47, no. 2 (2018): 565-599.

[12] Krawczyk, Bartosz, Leandro L. Minku, Joao Gama, Jerzy Stefanowski, and MichałWoźniak. "Ensemble learning for data stream analysis: A survey." Information Fusion 37 (2017): 132-156.

[13] Pérez-Gállego, Pablo, Alberto Castaño, José Ramón Quevedo, and Juan José del Coz. "Dynamic Ensemble Selection for Quantification Tasks." Information Fusion (2018).

[14] García, Salvador, Zhong-Liang Zhang, Abdulrahman Altalhi, Saleh Alshomrani, and Francisco Herrera. "Dynamic ensemble selection for multi-class imbalanced datasets." Information Sciences 445 (2018): 22-37.

[15] Cheriguene, Soraya, Nabiha Azizi, Nilanjan Dey, Amira S. Ashour, and AmelZiani. "A new hybrid classifier selection model based on mRMR method and diversity measures." International Journal of Machine Learning and Cybernetics (2018): 1-16.

[16] Almeida, Paulo RL, Luiz S. Oliveira, Alceu S. Britto Jr, and Robert Sabourin. "Adapting dynamic classifier selection for concept drift." Expert Systems with Applications 104 (2018): 67-85.

[17] Feng, Xiaodong, Zhi Xiao, Bo Zhong, Jing Qiu, and Yuanxiang Dong. "Dynamic ensemble classification for credit scoring using soft probability." Applied Soft Computing 65, no. C (2018): 139-151.

[18] Cruz, Rafael MO, Dayvid VR Oliveira, George DC Cavalcanti, and Robert Sabourin. "FIRE-DES++: Enhanced online pruning of base classifiers for dynamic ensemble selection." Pattern Recognition 85 (2019): 149-160.

[19] Cruz, Rafael MO, Robert Sabourin, and George DC Cavalcanti. "Prototype selection for dynamic classifier and ensemble selection." Neural Computing and Applications 29, no. 2 (2018): 447-457.

[20] Xiao, Hongshan, Zhi Xiao, and Yu Wang. "Ensemble classification based on supervised clustering for credit scoring." Applied Soft Computing 43 (2016): 73-86.

[21] Xia, Yufei, Chuanzhe Liu, Bowen Da, and FangmingXie. "A novel heterogeneous ensemble credit scoring model based on bstacking approach." Expert Systems with Applications 93 (2018): 182-199.

[22] Oliveira, Dayvid VR, George DC Cavalcanti, and Robert Sabourin. "Online pruning of base classifiers for Dynamic Ensemble Selection." Pattern Recognition 72 (2017): 44-58.

[23] Souza, Mariana A., George DC Cavalcanti, Rafael MO Cruz, and Robert Sabourin. "Online local pool generation for dynamic classifier selection." Pattern Recognition 85 (2019): 132-148.

[24] Bashir, Saba, Usman Qamar, and Farhan Hassan Khan. "BagMOOV: A novel ensemble for heart disease prediction bootstrap aggregation with multi-objective optimized voting." Australasian physical & engineering sciences in medicine 38, no. 2 (2015): 305-323.

[25] Bashir, Saba, Usman Qamar, Farhan Hassan Khan, and Lubna Naseem. "HMV: A medical decision support framework using multi-layer classifiers for disease prediction." Journal of Computational Science 13 (2016): 10-25.

[26] Liu, Xiaoqian, Qianmu Li, Tao Li, and Dong Chen. "Differentially private classification with decision tree ensemble." Applied Soft Computing 62 (2018): 807-816.

[27] van Rijn, Jan N., Geoffrey Holmes, Bernhard Pfahringer, and Joaquin Vanschoren. "The online performance estimation framework: heterogeneous ensemble learning for data streams." Machine Learning (2018): 1-28.

[28] Roy, Anandarup, Rafael MO Cruz, Robert Sabourin, and George DC Cavalcanti. "A study on combining dynamic selection and data preprocessing for imbalance learning." Neurocomputing 286 (2018): 179-192.

[29] Zhang, Zhong-Liang, Yu-Yu Chen, Jing Li, and Xing-Gang Luo. "A distance-based weighting framework for boosting the performance of dynamic ensemble selection." *Information Processing & Management* 56, no. 4 (2019): 1300-1316.

[30] Cruz, Rafael MO, Robert Sabourin, and George DC Cavalcanti. "META-DES. Oracle: Meta-learning and feature selection for dynamic ensemble selection." Information fusion 38 (2017): 84-103.

[31] Pratama, Mahardhika, Witold Pedrycz, and Edwin Lughofer. "Evolving Ensemble Fuzzy Classifier." IEEE Transactions on Fuzzy Systems (2018).

[32] Khamar, M., and M. Eftekhari. "Multi-Manifold based Rotation Forest for classification." Applied Soft Computing (2018).

[33] Dantas, Carine A., Rômulo de O. Nunes, Anne MP Canuto, and João C. Xavier-Júunior. "Evaluating the Dynamicity of Feature and Individual Classifiers Selection in Ensembles of Classifiers." In 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1-8. IEEE, 2018.

[34] Hussein, Sarfaraz, Pujan Kandel, Candice W. Bolan, Michael B. Wallace, and UlasBagci. "Lung and pancreatic tumor characterization in the deep learning era: novel supervised and unsupervised learning approaches." IEEE transactions on medical imaging (2019).

[35] Cao, Wenming, Sheng Qian, Si Wu, and Hau-San Wong. "Unsupervised Multi-task Learning with Hierarchical Data Structure." Pattern Recognition 86 (2019): 248-264.

[36] Lin, Peijie, Zhouning Peng, Yunfeng Lai, Shuying Cheng, Zhicong Chen, and Lijun Wu. "Short-term power prediction for photovoltaic power plants using a hybrid improved Kmeans-GRA-Elman model based on multivariate meteorological factors and historical power datasets." Energy conversion and management 177 (2018): 704-717.

[37] Gu, Hui, Yanfeng Cui, Hongxia Zhu, Rui Xue, and Fengqi Si. "A new approach for clustering in desulfurization system based on modified framework for gypsum slurry quality monitoring." Energy 148 (2018): 789-801.

[38] Wu, Minghu, Xiang Li, Cong Liu, Min Liu, Nan Zhao, Juan Wang, Xiangkui Wan, Zheheng Rao, and Li Zhu. "Robust global motion estimation for video security based on improved k-means clustering." Journal of Ambient Intelligence and Humanized Computing 10, no. 2 (2019): 439-448.

[39] Lopez, Christian, Scott Tucker, Tarik Salameh, and Conrad Tucker. "An unsupervised machine learning method for discovering patient clusters based on genetic signatures." Journal of biomedical informatics 85 (2018): 30-39.

[40] Wang, Sherrie, George Azzari, and David B. Lobell. "Crop type mapping without field-level labels: Random forest transfer and unsupervised clustering techniques." Remote sensing of environment 222 (2019): 303-317.

[41] Chung, Neo Christopher, Bilal Mirza, Howard Choi, Jie Wang, Ding Wang, Peipei Ping, and Wei Wang. "Unsupervised classification of multi-omics data during cardiac remodeling using deep learning." Methods (2019).

[42] Zhong, Haidong, Xianyi Chen, and Qinglong Tian. "An Improved Reversible Image Transformation Using K-Means Clustering and Block Patching." Information 10, no. 1 (2019): 17.

[43] Nematzadeh, Zahra, Roliana Ibrahim, and Ali Selamat. "Comparative studies on breast cancer classifications with k-fold cross validations using machine learning techniques." In 2015 10th Asian Control conference (ASCC), pp. 1-6. IEEE, 2015.

[44] Ani, R., Jithu Jose, Manu Wilson, and O. S. Deepa. "Modified Rotation Forest Ensemble Classifier for Medical Diagnosis in Decision Support Systems." In Progress in Advanced Computing and Intelligent Engineering, pp. 137-146. Springer, Singapore, 2018.

[45] Omondiagbe, David A., Shanmugam Veeramani, and Amandeep S. Sidhu. "Machine Learning Classification Techniques for Breast Cancer Diagnosis." In IOP Conference Series: Materials Science and Engineering, vol. 495, no. 1, p. 012033. IOP Publishing, 2019.

[46] Buchlak, Quinlan D., Nazanin Esmaili, Jean-Christophe Leveque, Farrokh Farrokhi, Christine Bennett, Massimo Piccardi, and Rajiv K. Sethi. "Machine learning applications to clinical decision support in neurosurgery: an artificial intelligence augmented systematic review." Neurosurgical review (2019): 1-19.

[47] M. Fernandez-Delgado,E . Cernadas,S . Barro,a nd D. Amorim," Do we need hundreds of classifiers to solve real world classification problems?"Journal of Machine Learning Research, vol. 15, pp. 3133-3181,2014. [On line]. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016).

[48] Heart Disease Dataset -https://www.kaggle.com/c/heart-disease dated: Sept 2019.

[49] Diabetes Dataset - https://www.kaggle.com/c/diabetes dated: Sept 2019.

[50] Blood Transfusion Dataset - https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center: Sept 2019

[51] Sonar Dataset - https://www.kaggle.com/adx891/sonar-data-set: Sept 2019

[52] Cardiotocography Dataset - https://archive.ics.uci.edu/ml/datasets/cardiotocography: Sept 2019

[53] Phoneme Dataset - https://sci2s.ugr.es/keel/dataset.php?cod=105: Sept 2019

[54] Breast Cancer Dataset - https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic): Sept 2019

[55] Thyroid Dataset - https://sci2s.ugr.es/keel/dataset.php?cod=67: Sept 2019

[56] A Ramezan, Christopher, Timothy A Warner, and Aaron E Maxwell. "Evaluation of sampling and cross-validation tuning strategies for regional-scale machine learning classification." Remote Sensing 11, no. 2 (2019): 185.

[57] Düntsch, Ivo, and Günther Gediga. "Confusion matrices and rough set data analysis." In Journal of Physics: Conference Series, vol. 1229, no. 1, p. 012055. IOP Publishing, 2019.

[58] Bensusan, Hilan, and Alexandros Kalousis. "Estimating the predictive accuracy of a classifier." In European Conference on Machine Learning, pp. 25-36. Springer, Berlin, Heidelberg, 2001.

[59] Koehrsen, William. "Beyond accuracy: Precision and recall." Towards Data Science (2018).

[60] Davis, Jesse, and Mark Goadrich. "The relationship between Precision-Recall and ROC curves." In Proceedings of the 23rd international conference on Machine learning, pp. 233-240. ACM, 2006.

[61] Soleymani, Roghayeh, Eric Granger, and Giorgio Fumera. "F-Measure Curves: A Tool to Visualize Classifier Performance Under Imbalance." Pattern Recognition (2019): 107146.

[62] Dantas, Carine, Romulo Nunes, Anne Canuto, and João Xavier-Júnior. "Instance Hardness as a Decision Criterion on Dynamic Ensemble Structure." In 2019 8th Brazilian Conference on Intelligent Systems (BRACIS), pp. 108-113. IEEE, 2019.

[63] A.H.R. Ko, R. Sabourin, A.S. Britto Jr., From dynamic classifier selection to dynamic ensemble selection, Pattern Recognit. 41 (2008) 1735–1748.

[64] R.M.O. Cruz, G.D.C. Cavalcanti, T.I. Ren, A method for dynamic ensemble selection based on a filter and an adaptive distance to improve the quality of the regions of competence, in: Proceedings of the International Joint Conference on Neural Networks, 2011, pp. 1126–1133.

[65] Tiago Pessoa Ferreira de Lima, Anderson Tenório Sergio, Teresa Bernarda Ludermir, "Improving Classifiers and Regions of Competence in Dynamic Ensemble Selection", Intelligent Systems (BRACIS) 2014 Brazilian Conference on, pp. 13-18, 2014.

[66] K. Woods, W.P. Kegelmeyer Jr., K. Bowyer, Combination of multiple classifiers using local accuracy estimates, IEEE Trans. Pattern Anal. Mach. Intell. 19 (1997) 405–410.

[67] Georgiou, Harris V., and Michael E. Mavroforakis. "A game-theoretic framework for classifier ensembles using weighted majority voting with local accuracy estimates." arXiv preprint arXiv:1302.0540 (2013).

[68] Vriesmann, Leila M., Alceu S. Britto, Luiz S. Oliveira, Alessandro L. Koerich, and Robert Sabourin. "Combining overall and local class accuracies in an oracle-based method for dynamic ensemble selection." In 2015 International Joint Conference on Neural Networks (IJCNN), pp. 1-7. IEEE, 2015.

[69] Brun, André L., Alceu S. Britto, Luiz S. Oliveira, Fabricio Enembreck, and Robert Sabourin. "Contribution of data complexity features on dynamic classifier selection." In 2016 International Joint Conference on Neural Networks (IJCNN), pp. 4396-4403. IEEE, 2016.

[70] P.C. Smits, Multiple classifier systems for supervised remote sensing image classification based on dynamic classifier selection, IEEE Trans. Geosci. Remote Sens. 40 (4) (2002) 801–813.

[71] G. Giacinto, F. Roli, Dynamic classifier selection based on multiple classifier behaviour, Pattern Recognit. 34 (2001) 1879–1881

[72] P.R. Cavalin, R. Sabourin, C.Y. Suen, Dynamic selection approaches for multiple classifier systems, Neural Comput. Appl. 22 (3–4) (2013) 673–688.

[73] Cruz, Rafael MO, Robert Sabourin, and George DC Cavalcanti. "Analyzing dynamic ensemble selection techniques using dissimilarity analysis." In IAPR Workshop on Artificial Neural Networks in Pattern Recognition, pp. 59-70. Springer, Cham, 2014.

[74] A.S. Britto, Jr., R. Sabourin, L.E.S. de Oliveira, Dynamic selection of classifiers—a comprehensive review, Pattern Recognit. 47 (11) (2014) 3665–3680.

[75] L.I. Kuncheva, A theoretical study on six classifier fusion strategies, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2) (20[53]02) 281–286.

[76] L. Breiman, Bagging predictors, Mach. Learn. 24 (1996) 123–140.

[77] Isikhan, Selen Yilmaz, Erdem Karabulut, Afshin Samadi, and Saadettin Kılıçkap. "Adaptation of Error Adjusted Bagging Method for Prediction." International Journal of Data Warehousing and Mining (IJDWM) 15, no. 3 (2019): 28-45.

[78] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, in: Proceedings of the Second European Conference on Computational Learning Theory, 1995, pp. 23–37.

[79] Jiang, Zhehan, Kevin Walker, and Dexin Shi. "Applying AdaBoost to Improve Diagnostic Accuracy." Methodology (2019).

[80] D. Ruta, B. Gabrys, Classifier selection for majority voting, Inf. Fusion 6 (1) (2005) 63–81.

[81] Mahdavi, Sedigheh, Shahryar Rahnamayan, and Abbas Mahdavi. "Majority voting for discrete population-based optimization algorithms." Soft Computing 23, no. 1 (2019): 1-18.