Dynamic Committees for Handling Concept Drift in Databases (DCCD)

By

Mohammed Alshammeri

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.Sc. degree in
Computer Science

School of Electrical Engineering and Computer Science

University of Ottawa

Abstract

Concept drift refers to a problem that is caused by a change in the data distribution in data mining. This leads to reduction in the accuracy of the current model that is used to examine the underlying data distribution of the concept to be discovered. A number of techniques have been introduced to address this issue, in a supervised learning (or classification) setting. In a classification setting, the target concept (or class) to be learned is known. One of these techniques is called "Ensemble learning", which refers to using multiple trained classifiers in order to get better predictions by using some voting scheme. In a traditional ensemble, the underlying base classifiers are all of the same type. Recent research extends the idea of ensemble learning to the idea of using committees, where a committee consists of diverse classifiers. This is the main difference between the regular ensemble classifiers and the committee learning algorithms. Committees are able to use diverse learning methods simultaneously and dynamically take advantage of the most accurate classifiers as the data change. In addition, some committees are able to replace their members when they perform poorly.

This thesis presents two new algorithms that address concept drifts. The first algorithm has been designed to systematically introduce gradual and sudden concept drift scenarios into datasets. In order to save time and avoid memory consumption, the Concept Drift Introducer (CDI) algorithm divides the number of drift scenarios into phases. The main advantage of using phases is that it allows us to produce a highly scalable concept drift detector that evaluates each phase, instead of evaluating each individual drift scenario.

We further designed a novel algorithm to handle concept drift. Our Dynamic Committee for Concept Drift (DCCD) algorithm uses a voted committee of hypotheses that vote on the best base classifier, based on its predictive accuracy. The novelty of DCCD lies in the fact that we employ diverse heterogeneous classifiers in one committee in an attempt to maximize diversity. DCCD detects concept drifts by using the accuracy and by weighing the committee members by adding one point to the most accurate member. The total loss in accuracy for each member is calculated at the end of each point of measurement, or phase. The performance of the committee members are evaluated to decide whether a member needs to be replaced or not. Moreover, DCCD detects the worst member in the committee and then eliminates this member by using a weighting mechanism.

Our experimental evaluation centers on evaluating the performance of DCCD on various datasets of different sizes, with different levels of gradual and sudden concept drift. We further compare our algorithm to another state-of-the-art algorithm, namely the MultiScheme approach. The experiments indicate the effectiveness of our DCCD method under a number of diverse circumstances. The DCCD algorithm generally generates high performance results, especially when the number of concept drifts is large in a dataset. For the size of the datasets used, our results showed that DCCD produced a steady improvement in performance when applied to small datasets. Further, in large and medium datasets, our DCCD method has a comparable, and often slightly higher, performance than the MultiScheme technique. The experimental results also show that the DCCD algorithm limits the loss in accuracy over time, regardless of the size of the dataset.

Acknowledgment

I am indebted to a very long list of people who spent a lot of time and effort to support me. Many thanks go to:

A big thank you goes to my supervisor Dr. Herna Viktor for her great inputs and guidance throughout the entire thesis. Her effort will never be forgotten and always be appreciated. I would also like to thank my co- supervisor Dr. Azzedine Boukerche for his understanding and support. Moreover, I would like to thank my parents and my siblings; without them I would have been lost.

Many thanks to the Saudi Cultural Bureau in Canada for their advice, support and encouragement have brought me where I am today. My warm thanks also go to my friends and colleagues who have encouraged me over the course of this endeavour.

Table of Contents

Table of Figures

CHAPTER ONE
INTRODUCTION

CHAPTER ONE

INTRODUCTION

Data mining is one of the steps of discovering knowledge which refers to a process that analyzes large databases to find valid, new, useful, and understandable patterns. A classifier is one of the data mining techniques which combines evolutionary computing, reinforcement learning, supervised /unsupervised learning, and heuristics to provide intelligent systems [**4**]. In such systems, the target attribute (or class) is known and our task is to build a model to describe the interplay between other attributes and the class. Our main focus in this study will be on building accurate classifiers when considering data that experiences changes, or drifts, in their distributions.

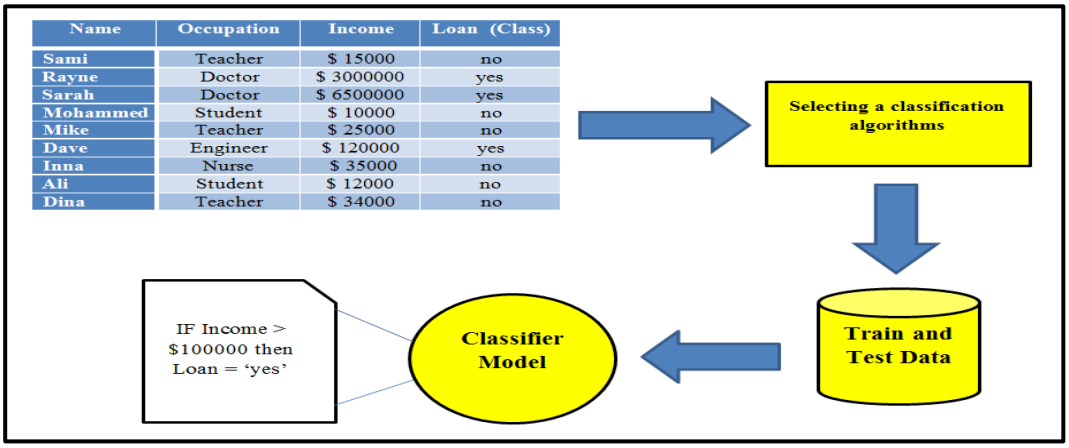

Figure 1: Data mining bank model distinguishes which customers are qualified for a loan

For example, suppose that a bank aims to build a data mining model that can distinguish which customers are qualified for a loan (i.e. target class), as shown in Figure 1. The data mining approach to this problem will start by gathering as many examples as possible of both qualify and not qualify. Next, the data miner will provide these examples, together with labels and target class, indicating if they qualify or not, to the selected classifier which will produce a classification or prediction rule. The selected classifier aims to generate a rule that makes the most accurate predictions possible on new test examples, as shown in Figure 2.

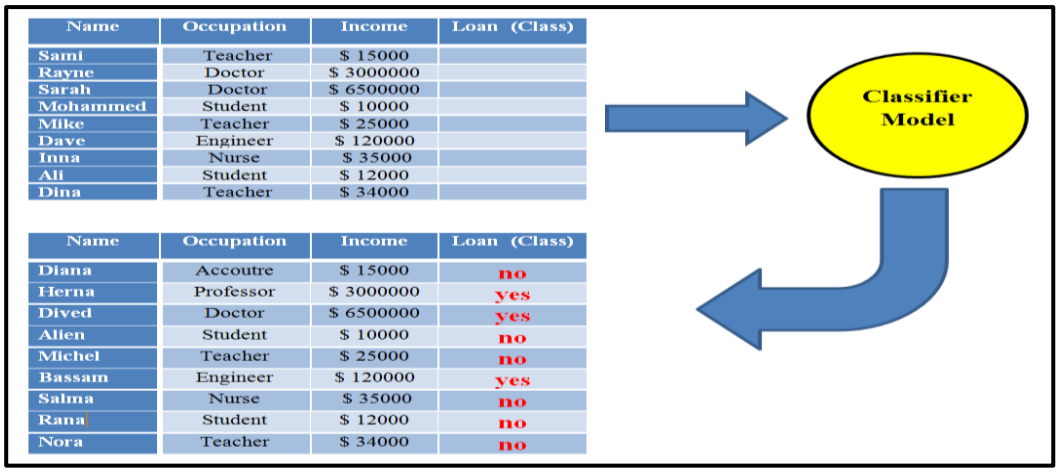

Figure 2: New unlabeled customers, such a rule attempt to predict if it is qualified or not

One of the issues that face the data mining society is the ***concept drift***, which refers to changing in the data distribution that leads to changes to the target classes [**7**] [**8**] [**9**] [**10**] and [**11**]. Concept drift is defined as a quantity that needs to be predicted where the concept is unstable and it changes over a certain period of time. Common types of concepts are weather patterns, customer preferences, temperature and behavioral changes. The underlying data distribution that is used in explaining concepts will also be subject to some changes as a result of the unstable nature of concepts. Such changes in the underlying data distribution cause the models built on old data to be inconsistent with the new concept's data, which will lead to the model being outdated. This creates a problem known as concept drift which complicates the task of learning the new model and the new data that makes up the concept [**12**].

For instance, suppose the same bank as in Figure 2 uses the original data mining model, to distinguish between those customers that are qualified for a loan and those that are not eligible. Moreover, suppose the value of national currency dropped ten times, which is sometimes a cause of concept drift; from the time that model was built. Based on these two

assumptions, the regular data mining approach will approve the loan for all applicants who have an income over $100000 regardless of the change in the data distribution. This change leads to a change in the target classes, which can be seen in the table as the red "yes" in Figure 3. Traditional data mining methods consider all incoming data as static data represented by a set of features and this, in turn, does not change the data regardless of modifications from the external environment [7].

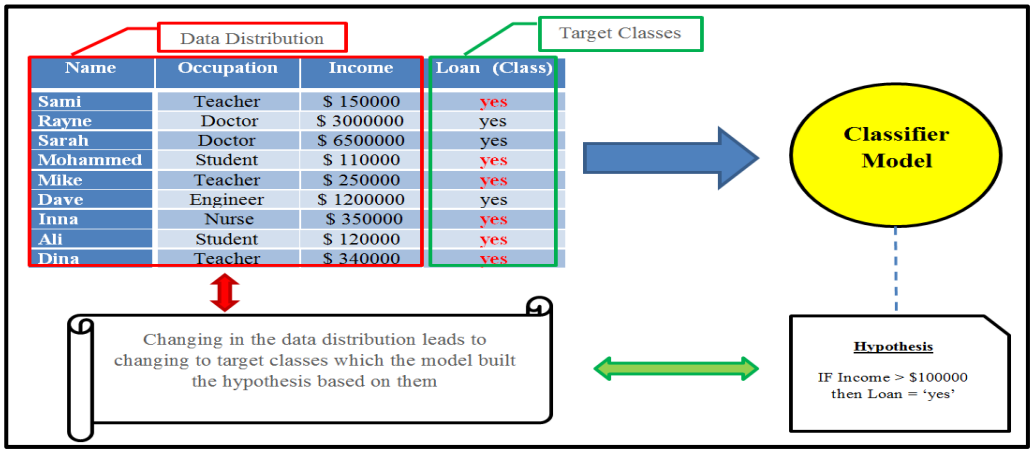

Figure 3: Concept drift scenario

To summarize the problem statement of this study, the data distributions often change over time due to the non-stationary environments in real world datasets. ***This change leads to a reduction in the current classifier's accuracy that is used in examining the underlying data distribution.*** A concept may drift due to several motivations such as the above-mentioned changing of the value currency (or say a sensor failure), which increases network traffic. Due to this issue, the need for such classifiers that handle concept drift become necessary. In such a case, the adopted classifier should be able to adjust quickly to changing conditions.

1.1 Motivation

Recall that the classification task refers to one of the data mining techniques that is used to train a training set, collect the records, and assign a value to each record which is called "target class". A classifier is the data mining algorithm that used to construct a model. If we use one algorithm, this is referred to as a base learner or learning methods. Examples include Decision Trees, Rule Learners, and Support Vector Machines. In the real world, classes, the target

concepts, and data distributions are often changing over time. Concept drift is a cause of non-stationary environments that might change the data distribution.

For instance, the patterns of customers' taste in buying products are constantly changing over time. This problem complicates the task of a learning model to learn from the input data and requires special approaches that are different from commonly used techniques [**13**]. Throughout the past two decades, various learning approaches have been developed and implemented to address the problem of concept drifts. Most of the previous studies assumed that using one base classifier would be sufficient to handle concept drift properly, which is not always correct [**14**] [**15**] [**9**] and [**10**]. Moreover, the current classifier may be accurate at a certain time, but when concept drifts occur, the selected classifier may lose its efficiency. This led to the idea of ensemble learning, which selects one learning method to create several classifiers from the selected learning method and then selects the most accurate classifier among them. The ensemble systems will be discussed in details in Chapter 2.

However, a problem with regular ensemble systems is the assumption of selecting one learning method which subsequently creates more than one classifier. This assumption ignores the idea of selecting a poor base classifier and creates more than one classifier, which might affect the system outputs even if we use an excellent ensemble system. Furthermore, the selected learning method might be effective at a certain time but when concept drifts occur, the learning method may have lost its efficiency. Therefore, the need of merging diverse base classifiers can assure that even when concept drifts occur, we still have the best base classifier.

For example, suppose a company needs to discover patterns from their data. Also, suppose they select a Decision Tree classifier as a base classifier and then employ a Bagging ensemble. A Decision Tree, which **will be discuss in section 3.1.1,** refers to flow-chart-like tree structure while Bagging, **will be discuss in section 2.3.1**, refers to an ensemble classifier that trains multiple classifiers and then selects the most accurate model among them by the voting mechanism. The issue with this scenario is the assumption that applying Decision Tree classifier is good for every dataset at all times. As we discussed earlier, the data distributions are often changing over time, which means that the decision tree may be excellent at the moment, but it might not be the best choice when concept drifts start to appear [**6**].

This thesis focuses on creating a dynamic committee that replaces weak base classifiers (learning methods) by new base classifiers in order to handle concept drifts. We approach the problem of concept drift by building a dynamic committee of diverse base classifiers. This

dynamic committee allows adding and removing base classifiers to a blacklist, based on their performance which also provides diversity of base classifiers. The main idea of this approach is to increase the competition between the committee members and base classifiers, which will increase the overall performance of the system. For example, suppose a committee consists of three base classifiers as shown in Figure 4.

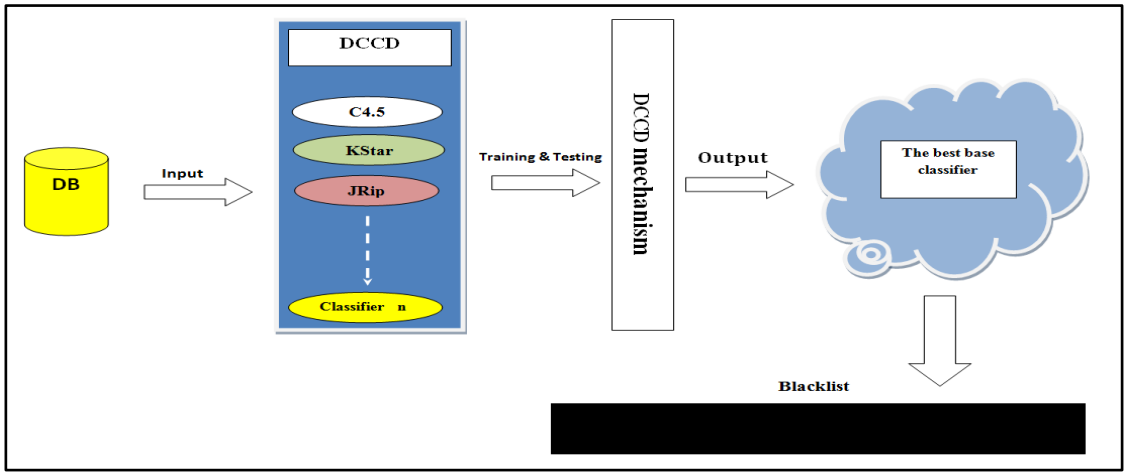

Figure 4: Dynamic Committee for Concept Drift

When a batch of data comes in, we train all the committee members and classifiers and then record their performance. Suppose the best classifier in the beginning is classifier 2. After some time, concept drifts start to appear. If classifier 2 fails to match the committee standards, the committee moves classifier 2 to the blacklist and adds a new base classifier to the committee, as shown in Figure 5.

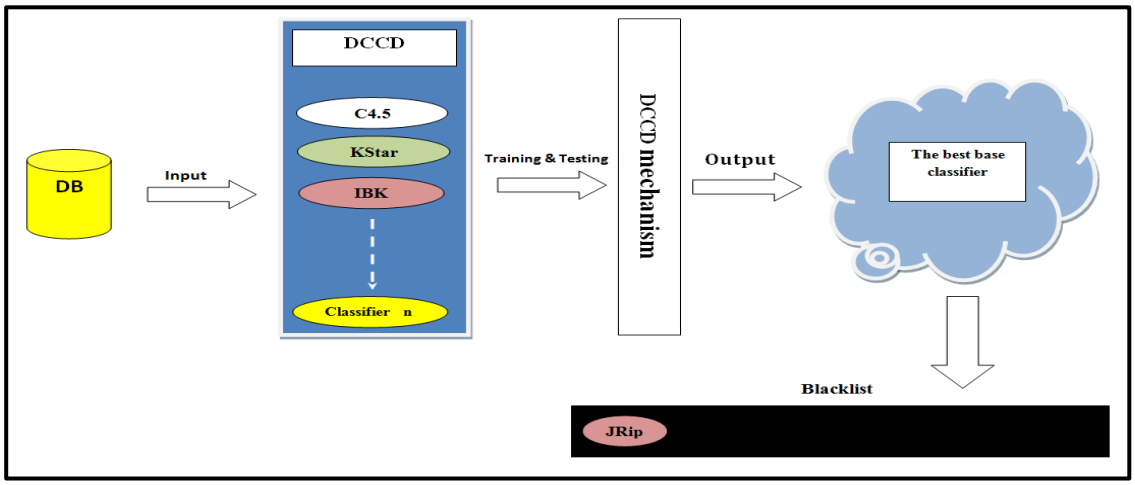

Figure 5: The new dynamic committee for concept drift

1.2 Layout

This thesis is organized as follows. In Chapter 2, we provide an overview of previous works on the concept drift issue. We also provide a detailed discussion of the different types of concept drift. Chapter 3 presents an overview and critique of classifiers that will be used in our experiment. This is followed by Chapter 4 which provides our methodology. Chapter 5 presents the experimental design which includes the pre-processing as well as the methods used to apply the drift scenarios. Chapter 6 provides the experiment results when evaluating our algorithm's performance. In Chapter 7, we further evaluate and examine the results of our experiments by comparing it a state-of-the-art algorithm. Finally, in Chapter 8, we conclude by providing a summary that highlights our contributions and provide possible future directions in research.

CHAPTER TWO
LITERATURE REVIEW

CHAPTER TWO

LITERATURE REVIEW

In supervised learning, there is some form of outputs knowledge that we expect from a given source of inputs [**16**]. The other type of machine learning that can occur is called unsupervised learning. Here, unlike in supervised learning, we only have a set of data samples that we wish to train, but we do not have a function that will map the inputs in the available set to specific outputs in a way that we can determine. Both of the machine-learning types explained above face an issue called concept drift [**17**].

Yeung and Chan [**3**] define the classification task as follows: Given input data R, collections of records, and a set of classes $C = \{C_1, C_2, C_3, \ldots, C_n\}$. The classification task is to define a mapping f: R $\rightarrow C$ where each tuple in R has to assign to one class in C. This procedure is called the Decision Function. Based on this general formula, the best classifiers are the ones that assign the maximum number of tuples in R to the right classes in *C* [**3**].

2.1 Concept Drift Overview

There are two major types of concept drift; namely Real concept drift and Virtual concept drift. Real concept drift, as introduced in Chapter 1, is the problems that are caused by a change in the data distribution, which leads to reduction in the current model accuracy that is used in examining the underlying data distribution of the concept. Most researchers refer to the real concept drift as "*concept drift or concept shift*". Concept drift is also described as phenomena

that include examples that might have legitimate labels at one time and illegitimate labels at another time. The following example may help to further illustrate concept drift: when data distribution in R changes, this change affects the Decision Function. Such changes in the data distribution cause the classifiers to assign tuples to the incorrect classes, which will affect the classifiers' accuracy. Koronacki [**18**] uses the example of a cloud as a target concept; concept drift occurs when the cloud changes its position, shape, and size in the sky over a certain period of time. With regards to the Bayesian decision theory, the transformations to the cloud equate to the changes that take place on the form of the prior target cloud [**18**].

Concept drift commonly occurs in the real world, especially when it comes to people's changing preferences for products and services. For instance, customers' taste in shopping might change over time due to many reasons such as their economic status, new available technologies or seasonality. The reasons behind this change are not as essential as dealing with this change itself. Suppose we want to predict computer sales in a company, and say that the company has developed a model to predict the future sales. The model uses different inputs such as the amount of money spent on marketing, offers, promotions, and so on. Over time, customers change their preferences and the company will notice a reduction in the current model's accuracy, and that is considered as one of the causes of concept drift.

Concepts are subject to change over time, which means that they are unstable in nature. Such changes in the underlying data distribution models make the task of learning, especially machine learning, more complicated. Learning also becomes difficult if there are changes in the hidden context of the target concept, which the model tries to predict, and this leads to concept drifts. The problem of handling concept drift usually arises when it comes to distinguishing between true concept drift and noise. Some machine learning algorithms might overreact to noise such as STAGGER [**19**], misinterpreting the noise to be concept drifts. Other algorithms on the other hand might react to noise by adjusting to the changes very slowly, such as FLORRA 1 and FLORRA 2 [**7**].

2.2 Types of Concept Drift

As we introduced in the previous section, concept drift has two major types; real concept drifts and virtual concept drifts. Virtual concept drift is defined as the need to change the current model due to a change in the data distribution. The hidden changes that exist in a certain context might cause a change in the target concept, which might in turn cause a change in the underlying

data distribution of the concept. If the target concept remains the same, the underlying data distribution might change to reflect changes to the concept, which might create a need to revise the current model that is used in explaining the concept. This creates a virtual concept drift that necessitates a change in the current model [**20**].

To further illustrate virtual concept drift, consider the following example of Widmer [**29**]. When a teacher explains a certain topic to students, the teacher might consider presenting only the relative knowledge to the topic. After a while, the relative knowledge that was explained by the teacher might become less important to the topic. For instance, when we want to construct a data warehouse, we only consider the relative and essential attributes from the relational database. Then, we apply the data mining methods to this data warehouse. Eventually, some of these attributes become less important, and that is what we call virtual concept drift [**29**].

Virtual concept drift occurs on a high level (i.e. data warehouse structure level) while real concept drift occurs on a low level (i.e. data distribution). An example of virtual concept drift and a real concept drift is that virtual concepts might occur in cases of spam categorization while the real concept drifts might not be caused by spam categorizations. Virtual concept drifts ensure that the shifts in the concept have been properly represented in the current model that is used in explaining the underlying distribution data. Virtual concept drifts, which are also known as sampling shifts, help in determining the types of unwanted messages that remain the same over a long period of time [**20**].

There are four subcategories of real concept drifts namely sudden, gradual, incremental, and seasonal. The two most common types of concept drifts that might occur, in the real world, include sudden (instantaneous) and gradual concept drifts. An example of a sudden concept drift is the changing preferences of customers when they demand products or services that will meet their constantly changing needs. Sudden concept drifts are the easiest type to recognize by most methods due to the radical change in the data distributions or in accuracy reduction. Another type of concept drift that exists in the real world is gradual concept drift, where a certain aspect changes over a gradual period of time [**20**].

Figure 6 illustrates the differences between sudden concept drift and gradual concept drifts. Sudden drifts occur in a short period of time and at an extremely rapid rate in the data distribution as shown on Figure 6. In contrast, gradual concept drifts evolve the data

distributions slowly and steadily over a long period of time, which makes gradual drifts hard to recognize. Some researchers, such as Stanley [**9**], divided the gradual concept drifts into two subcategories based on the rate of change: moderate and slow [**9**].

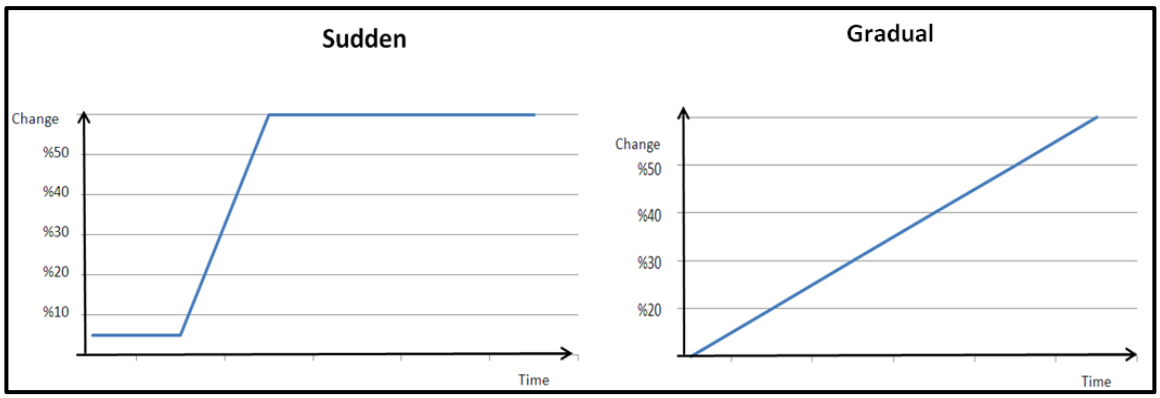

Figure 6: Illustration of the differences between sudden and gradual concept drifts concept drift

Incremental or reoccurring concept drift is another type of concept drift, which refers to a reoccurring change in data distributions at an unknown period in time. Another type of concept drift is seasonal concept drift, which refers to change in data distributions at regular periods in time [**12**], [**22**], [**23**]. We can distinguish between incremental (reoccurring) and seasonal concept drift based on the prediction certainty as presented in Figure 7. In other words, seasonal drift occurs in a specific time every year while the occurrence of incremental drift cannot be predicted [**11**].

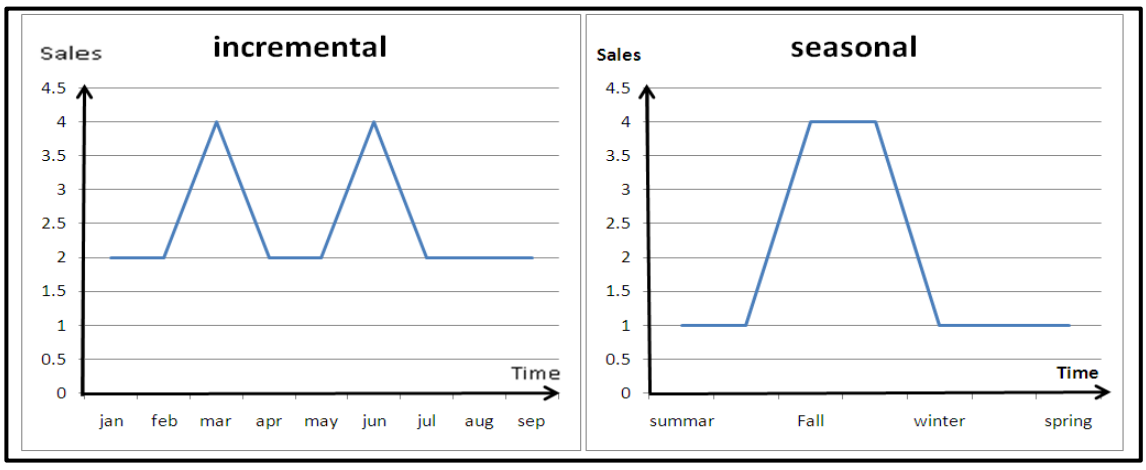

Figure 7: Illustration of the differences between incremental (reoccurring) and seasonal concept drift

For instance, a company has developed a model to predict customers' behaviors. The model shows that at Christmas time, the sales of a certain product increases in this period every

year, and that is what we call the seasonal concept drift. Whereas, the reoccurring concept drift appears at different times depending on the previous event. For example, the government of Saudi Arabia wants to increase their oil production this year. The decision of increasing the production will be based on the rate of the international demands for oil. In other words, Saudi Arabia is interested in increasing its production if the demand for oil becomes high.

2.3 Systems for Handling Concept Drift

Over the years, researchers have developed a number of approaches to address the concept drift issue. Therefore, categorizing the approaches becomes a necessary action. There are two dimensions to categorize the approaches, based on **How** a system should handle concept drift or **When** a system should concept drift [**11**].

Figure 8 gives a general idea about how concept drift systems work, which consists of four main aspects that need to be considered. The first aspect is the future assumption, which the model is trying to predict. Therefore, the learner designer must predict the future data source. The second aspect is detecting the changing type, which consists of the previously introduced four types. The third aspect is the learning adaptively, which represents the classifiers' quality. Learning adaptively must be based on the first and second aspects. The last aspect is the model selection and evaluation [**11**].

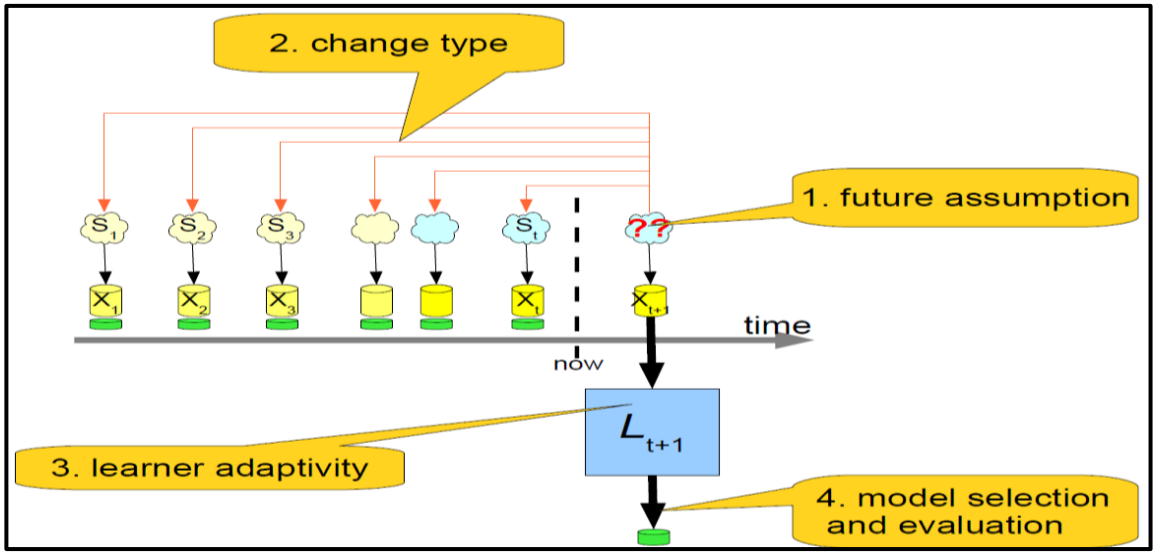

Figure 8: Design Concept drift learning system [11]

In Figure 3, S_1, S_2 , S_3 …, S_t, represent the data source, which can be a historical data (i.e. data warehouse) or streaming data. Once S_1 arrives to the learner L_{t+1}, we train L_{t+1} to predict X_{t+1}. If S_2 arrives, the (classifier) learner L_{t+1}will detect if there is any type of concept drifts in S_2. If there are concept drifts present, the learner will detect the change and change the current model. The **When** technique assumes that the effective way to handle concept drifts is by focusing on a suitable change detection that takes place in the coming data. The **When** technique is divided into two subcategories; for streaming data and batch data [**11**].

The most common method that is used in handling concept changes is information filtering where data streams are classified according to whether they are relevant or irrelevant to the target concept. The main purpose of information filtering is to reduce the information load presented to a user that might be of interest to them. Information filters are supposed to remove irrelevant information from the data streams to ensure that only the relevant information has been presented to the user. The reason behind that is that concepts are unstable and constantly changing. Information filters used in unstable environments have to consider classification accuracy to ensure that the concept changes have been properly documented [**24**].

The **How** technique, the most common dimension of categorizing, focuses on how the classifier handles concept drifts. In the sections below, we describe the approaches that handle concept drift based on the **How** categorizing. There are four system categories that handle concept drift according to the **How** technique. The first technique is ELS which refers to Ensemble Learning Systems for handling concept drift. Ensemble learning uses multiple trained classifiers to obtain better predictions by combined models using voting or weighted voting [**20**]. BLS is the second technique for handling concept drift which stands for Base Learning Systems. BLS uses one trained classifier in order to handle concept drifts. The third technique is Instance Selection Systems (ISS) for handling concept drift. ISS selects specific instances relevant to the current concept to train the current classifier. The last technique is IWS which refers to Instance Weighting Systems for handling concept drift. IWS detects the location where a concept begins to drifts [**11**].

2.3.1 Ensemble learning systems (ELS)

Ensemble learning refers to using multiple trained classifiers, i.e. base-learning-systems, in order to get better predictions by combined models using voting or weighted voting [**20**]. Ensemble learning can be used for improving prediction accuracy. This is what made ensemble

learning the most common technique that has been used to deal with the concept drift issue. In ensemble learning, the final decision must be taken either by combined classifications outputs of several models or by selecting the best one [**11**].

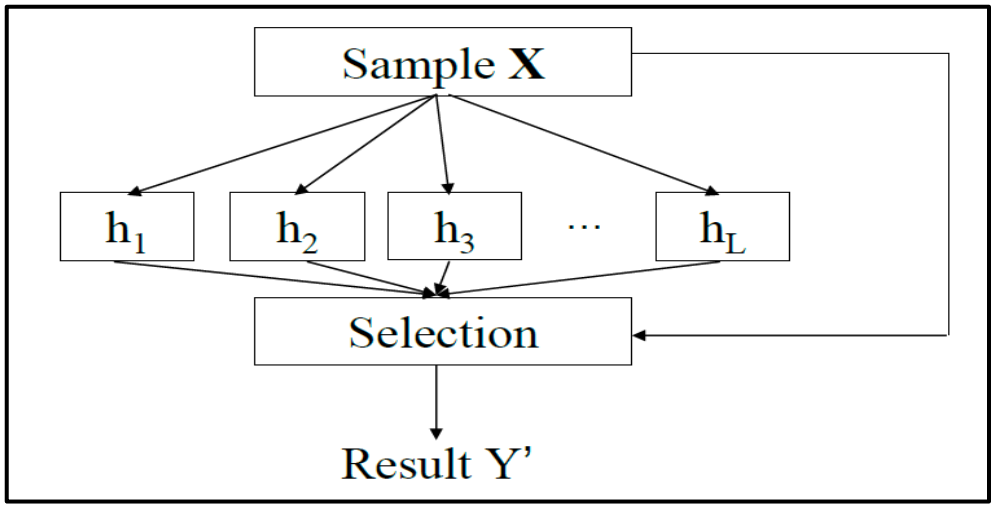

Figure 9: Ensemble learning system architecture [**3**]

In Figure 9, we present the most commonly used architecture of ensemble classifiers. In this example, X represents the type of ensemble classifier that we use, such as Bagging; while h_i stands for the individual classifier that we use, such as decision trees. After that, each ensemble classifier has a mechanism to select the best classifier output or combine more than one classifier output [**3**].

One of the most popular types of ensemble learning is Bagging. Bagging creates multiple predictors and uses them to get an aggregated predictor as shown in Figure 10. For an estimation problem, Bagging does aggregation averages over the multiple predictors while for classification problem, Bagging does majority voting to predict a class. Bagging makes bootstrap replicates of the learning set and uses them as new learning sets to form the multiple predictors. In other words, based on a uniform probability distribution, Bagging samples the training sets with replacement. Every sample has the same size as the original data, some instances may appear more than once in the same training set and then Bagging builds a classifier on each sample [**14**].

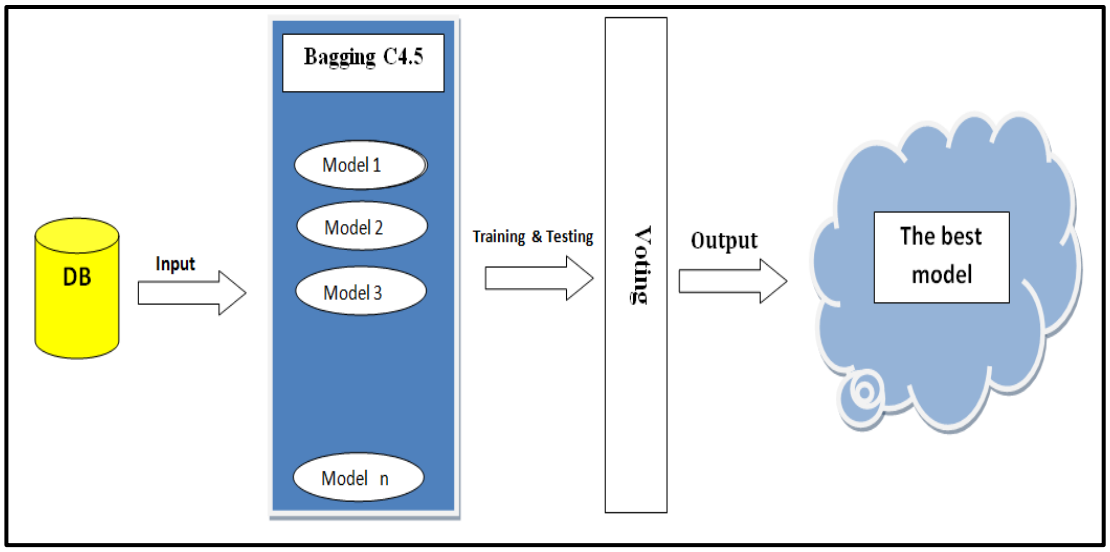

Figure 10: Bagging

Boosting, which is popular ensemble learning, is a general method for increasing the accuracy of any learning method. Freund and Schapire [25] introduced the AdaBoost algorithm in 1995. Boosting is a sequential production of classifiers where each classifier focuses on the previous one's errors. Each classifier focuses on the examples that are incorrectly predicted in previous classifiers by weighting them more [25].

Geoffrey [49] extends AdaBoost and merges it with Wagging to produce MultiBoostAB algorithm. Geoffrey aims to merge the high bias which is the greatest advantage in AdaBoost with the greatest advantage in Wagging. The superior variance reduction is the greatest advantage in Wagging. MultiBoostAB forms a committee to combine the outputs of the committee members in a classification task. This outputs combination creates a single classification from the committee as a whole. MultiBoostAB decides its final output by using the majority voting technique. Combining the committee members' outputs enables MultiBoostAB to develop decision committees in addition to classifying and weighting the committee members based on their performance [49].

Another Meta learner algorithm is Grading which is based on a straightforward technique. Grading aims to detect and correct incorrect predictions at the base level by grading them as correct or incorrect [65]. Grading detects which base classifiers obtain correct prediction on a given example. Grading adds a new class labels into the original dataset to encode whether the prediction of the base classifier for each example is correct or incorrect. Furthermore, when

the prediction of a base classifier for an example is incorrect, the selected Meta classifier learns whose responsible to predict in case a base classifier errs [65].

There are many ensembles learning classifiers for handling concept drift. Schlimmer [**19**] presents the first ensemble classifier that addresses the concept drift issue called STAGGER. STAGGER conducts the weighted voting, which maintains a set of concept descriptions that will be used to construct the best according to their relevance with the new data. STAGGER handles concept drift in two main steps. The first step is adjusting the characterization weights while the second step is gradually creating new ones. The second step is "described in terms of a search through the space of possibilities and is shown to require linear space with respect to the number of attribute-value pairs in the description language" [**19**].

Tsymbal [**10**] suggests a treatment to address the issue of the local concept drifts; this treatment is called the dynamic integration of classifiers (DIC). The local concept drifts issue arises when a lower-weighted classifier is assigned to predictions from base classifiers, not according to its performance, but due to their global accuracy on the current block, which is high. The type and severity of changes may depend on the location in the instance space [**12**]. Street and Kim [**27**] introduce a streaming ensemble algorithm (SEA) that handles the problem of concept drifts in streaming or large scale data. SEA is an ensemble of decision trees, which uses a mechanism that divides the training data into sequence chunks. Separate classifiers are built based on these chunks. The main goal of this method is to provide a high level of scalability by reducing the time that the ensemble learning such as Bagging takes. Bagging requires the complete availability to the training set all at once.

Scholz and Klinkenberg [**15**] present Knowledge-Based Sampling (KBS) that trades off predictive performance versus scalability by an online-classifier by reading examples in the batches and deciding, for each batch, if there is a need to add a new expert to the ensemble or not [**15**]. Further research improves the idea ensemble learning to the dynamic ensemble learning systems, which means the chosen classifiers may change over time. The base learning classifiers for concept drift become essential to the data mining sociality after it has been proven that the regular ensemble learning, such as Bagging [**14**], becomes inappropriate in the data mining context because they require the complete availability for the training set all at once [**27**].

For that reason, many base learning classifiers have been established to address the issue of the concept drift. Researchers have used many techniques to address the issues. For example, Harries [**8**] does not assume, as many researchers do, that the training set is homogeneous. Therefore, he presents an incremental learning approach called SPLICE to detect the hidden context in the homogeneous datasets. Harries defines contextual clustering as follows: "a stable concept is an expression that holds true for some period of time" [**8**]. In the ensemble systems, we do not replace a classifier; we just select the most accurate classifier while the base learning classifier adds and removes classifiers according to their performances. For instance, Bagging is one of the ensemble systems which uses more than one classifier to get a better prediction and selects the best classifier by the voting mechanism. In the case of base learning systems, we select the most accurate classifier and we replace the weakest classifier by a new one [**8**].

Kolter [**28**] extends the idea of ensemble learning and builds ensembles of classifiers of different "age" so that each classifier recognizes the latest instances. However, most of the new ensemble approaches use some criteria to dynamically eliminate, reuse, or create new ensemble members, which are normally based on the base models' consistency with the current data. Kolter introduces the dynamic weighted majority (DWM); this method uses an online learner to train the ensemble by weighting the learners according to their performance. Afterwards, the online learner removes and adds new learners based on their performance [**28**].

Stanley [**9**] presents another method, which uses decision trees to handle concept drift; this method is called Concept Drift Committee (CDC). The committee consists of two or more decision tress. CDC uses a weighted committee that votes on the current classifier. If a classifier voting record drops below a minimum threshold, the classifier must be replaced by another classifier [**9**].

2.3.2 Base learning systems (BLS)

Base learning system (BLS) uses one trained classifier in order to handle the concept drift issue. The base learning systems that handle concept drift include systems using rule-based learning such as [**29**] or decision trees such as [**9**] and [**30**]. Other Base-Learning-Systems make use of Bayes theorem such as [**31**] or Support Vector Machine such as [**32**]. Some of the BLSs employ the same idea of the dynamic ensemble learning system, which means the chosen classifiers might change over time.

Harries' [**8**] method that is used in weighted voting is conceptual clustering, where stable hidden contexts are identified by clustering instances of the new concept that are similar to the hidden context. When compared to data combining approaches, ensemble techniques have been more effective in determining concept drift problems than the data combining approaches and they are, therefore, more suitable in data streams and batches because they do not need to retain any previous data sets as with the data combining methods.

Other researchers, such as Widmer in his early paper [**7**], introduce the FLORA system. The main principle for this system is to learn the concept gradually, by processing labeled training examples one at a time. In other words, the model cannot be built at once, but instead incrementally. Also, Widmer [**29**] uses rule-based learning, which is described as learning from previous experiences when a similar change reoccurs. This rule-based learning was added to his original system, the FLORA, and he calls the new versions FLORA2 and FLORA3. The main goal of these systems is to provide flexibility in adapting to the changes in the target concept and detects concept drift. Other researchers used the Support Vector Machine (SVM) technique such as Klinkenberg [**32**]. The principle tasks in his works are to provide a flexibility of changing the window size, selection of examples, and the example weighting; but the overfitting issue stopped researchers in this area. A fixed window size makes strong assumptions about how quickly the concept changes.

2.3.3 Instance selection systems (ISS)

In this section, we overview the Instance Selection Systems (ISS), that handle concept drift. The main idea of ISS is to choose instances relevant to the current concept [**20**]. ISS works by generalizing from a window that moves over recently arrived instances and uses the learnt concepts for prediction only in the immediate future. ISS determine any instance related to the current concept in order to select it. The main idea in this technique is to learn the concept over-time, which can be done by processing labeled training examples one at time. ISS has been considered as the best technique to handle seasonal and reoccurring concept drifts [**7**].

There are many algorithms that use the instance selection technique, but the most popular algorithm is FRANN presented by Widmer [**34**]. After that, the idea of FRANN was extended and FLORA [**7**] was presented. The problem with these approaches arises when a large size time window is unable to adapt quickly to the concept drift and, on the other hand, a small

size time window is unable to track a target concept that is stable or recurrent [**35**]. Therefore, Salganicoff made a modification and extinction to the window idea in FLORA and created Time-Windowed Forgetting [**36**]. Salganicoff's approach can improve learning by using a predefined set of time intervals that delete older exemplars. His approach decreases the classifier's accuracy when the input-space sampling distribution of the learning set is time-varying. Other researchers such as [**32**] and [**38**] used window adaptive size.

2.3.4 Instance weighting systems (IWS)

The ability to correctly detect the location and derive the contextual information where a concept begins to drift is essential in the study of domains with changing context. The correct location of changes can help to produce information that justifies and explains the change. Furthermore, the presence of noisy examples or irrelevant features in a dataset degrades the performance of machine learning algorithms. Therefore, weighting good instances may improve the algorithm's performance [**39**]. Instance weighting uses the ability of some learning algorithms, such as Support Vector Machines (SVMs), to process weighted instances [**32**]. Instances can be weighted according to their "age", and their competence with regard to the current concept [**32**]. Instance weighting methods could be a single classifier. Ivan [**40**] presents a single classifier for gradual forgetting. This approach introduces a time-based forgetting function. The main goal of this function is to make the recent observations more significant than the previous ones, which lead to the importance of examples to decrease over time. The main idea behind using the forgetting function is to assign to each training example a weight, based on its appearance over time [**40**].

Another single classifier is presented by Oommen [**41**] and is called the Stochastic Learning Weak Estimator (SLWE). SLWE was developed based on the principles of stochastic learning. Based on the value of the current sample, the estimate is updated at each time instant. However, this updating is not achieved using an additive updating rule. Others such as Chu and Zaniolo [**42**] propose a new boosting ensemble method that works based on a dynamic sample-weight assignment scheme. The Boosting scheme extends to handle concept drift via change detection. The technique assures faster learning and competitive accuracy using simpler base models because it achieves the accuracy of traditional boosting without requiring multiple passes through the data. In order to avoid serious deterioration of the ensemble performance, the change detection approach aims at significant data changes [**41**].

Another important method is presented by Mak and Krause [**43**], where they introduce a top-down learning method with the incorporation of a learning accuracy mechanism to efficiently detect and manage context changes within a large dataset. With the utilization of simple search operators to perform a convergent search with a graphical viewer to derive context information, the identified hidden contexts are shown with the location of the disjoint points, the contextual attributes that contribute to the concept drift, the graphical output of the true relationships between these attributes and the Boolean characterization which is the context. There are some methods such as [**44**] and [**45**] that make another group of evolving adaptation techniques, which can be reached by combining or selecting rules. Instance weighting methods usually focus on the instances, which were misclassified employed from boosting [**46**].

2.4 Discussion

The importance of concept drift increased over the years due to the complexity of this issue and the omnipresence of concept drift in datasets. In this section, we present a discussion about the types of concept drift in addition to the four main types of systems that handle concept drift. Moreover, this discussion includes the general advantages and disadvantages of each system type that handles concept drift, as well as the differences between them. Sudden, gradual, incremental, and reoccurring are the four main types of concept drift.

The difference between sudden concept drift and gradual concept drift is based on the drifting rate, whereas incremental drift can be identified by the timing of the drifting appearance. Also, seasonal drift can be identified based on the rate of change at a specific time. The main difference between incremental concept drift and seasonal one is the timing. In other words, seasonal drift occurs at a specific time while incremental can happen at any time. In the table below, we present different features that each type has in addition to the general advantages and disadvantages as well as some references to those types. Furthermore, some types of systems have been developed mainly to handle specific types of concept drifts.

Table 1 presents four main types of systems that handle concept drift in the data-mining field. The difference between those systems can be distinguished based on the mechanisms that these systems use. The systems are utilized to handle concept drift that may be identified as: base learning systems, instance selection systems, instance weighting systems, and ensemble

learning systems. The main advantages of the base learning systems are the scalability and the easiness of understanding the classifier structure.

	BLS	ELS	ISS	IWS
Type of change	All	All	Seasonal/ reoccurring	All/ gradual
Classifier	One	Multiple	One / Multiple	One / Multiple
Stream/Batches	Mostly Batches	Stream/Batches	Stream/Batches	Mostly Batches
Advantages	Simple / fast / scalable	More accurate BLA	More accurate then ELA in the seasonal and reoccurring	The accuracy increases over time by identifying the concept drift place
Disadvantages	Less accurate than ELA	Time consumption	Memory consumption	size of the Window
References	[7], [29] [9], [28] [8], [27] [19]	[27] [47] [19] [12] [48] [15] [11] [9] [28]	[46] [45], [44], [42] [36] [7]. [35]	[40], [41], [50] [45], [44], [42]

Table 1: Summary of the literature review

In contrast, the ensembles learning systems focuses on getting the highest accuracy possible regardless of other aspects such as the time consumption. Instance selection systems give an advanced mechanism specifically to deal with the seasonal and reoccurring concept drift, while instance weighting systems aim to learning from previous experiences, which can increase the system accuracy incrementally. Our proposed algorithm will be introduced in chapter 4. It can be considered as an ensemble learning system (ELS) because it uses multiple classifiers and then, selects the most accurate classifiers among them. The reason behind choosing ELS to handle concept drift is the advantage of using more than one base classifier beside the ability of replacing the unreliable classifier from the committee.

2.5 Summary

This chapter presented an overview of concept drifts in addition to its types. In addition, the systems that handle concept drift have been overviewed with regards to the various types of machine learning processes, and theoretical works to handle the concept drift issue that exists. Concept drift is commonly an issue in artificial intelligence activities as well as in the development of various types of technology that are used in the real world, such as in the

diagnoses of diseases. The discussion has also focused on the concept drifts by defining the term and also identifying the various types of methods that can be used in dealing with concept drifts.

In conclusion, the problem of concept drift refers to a change in the data distribution over time, which might lead to a reduction in the classifier accuracy. Concept drift is a complex learning task that requires different solutions from common learning techniques. In Chapter 4, we propose a dynamic committee approach to handle concept drift. Our suggested approach integrates different classifiers and removes them based on their performance.

CHAPTER THREE
OVERVIEW OF CLASSIFIERS

CHAPTER THREE

OVERVIEW OF CLASSIFIERS

Large databases are loaded with information that can be useful in some decision making. Hence, the need of extracting hidden knowledge from databases becomes a necessity. Recall that classifiers are learning algorithms developed in order to get useful knowledge from historical and streaming data. That is, classifier refers to a learner algorithm that analyzes data and constructs a model to predict categorical labels. Classifiers allow the presentation of information extracted from large database become understandable even for ordinary people. For example, we can build a classification model that categorizes jobs applications of potential employees based on their degree and experience. The classification model determines which job applicants are "qualified" or "unqualified" for the job [**6**].

This chapter provides an overview of the classifiers that we use as base learners in our work. We introduce each one of the techniques we use, and we present the advantaged and disadvantages of these algorithms, when aiming to address the issue of concept drift.

3.1 Decision Trees

Learning of decision tree rules entails the creation of a decision tree and then converts the decision trees into sets of rule. This simplified or determine the most powerful rule then used repeatedly first by eliminating data covered by the rule. For example, doctors may aim to analyze breast cancer data in order to predict the "best" treatment for the patient. The outcomes

of this prediction usually come in categorical labels such as risky or safe. Every classifier has definite number of successive stages from the initial input stage to the termination stage of the output. The decision tree is a "flow chart like tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node holds a class label" [**6**]. Figure 11 shows an example of a decision tree. This algorithm was designed and implemented by Ross Quinlan [**51**].

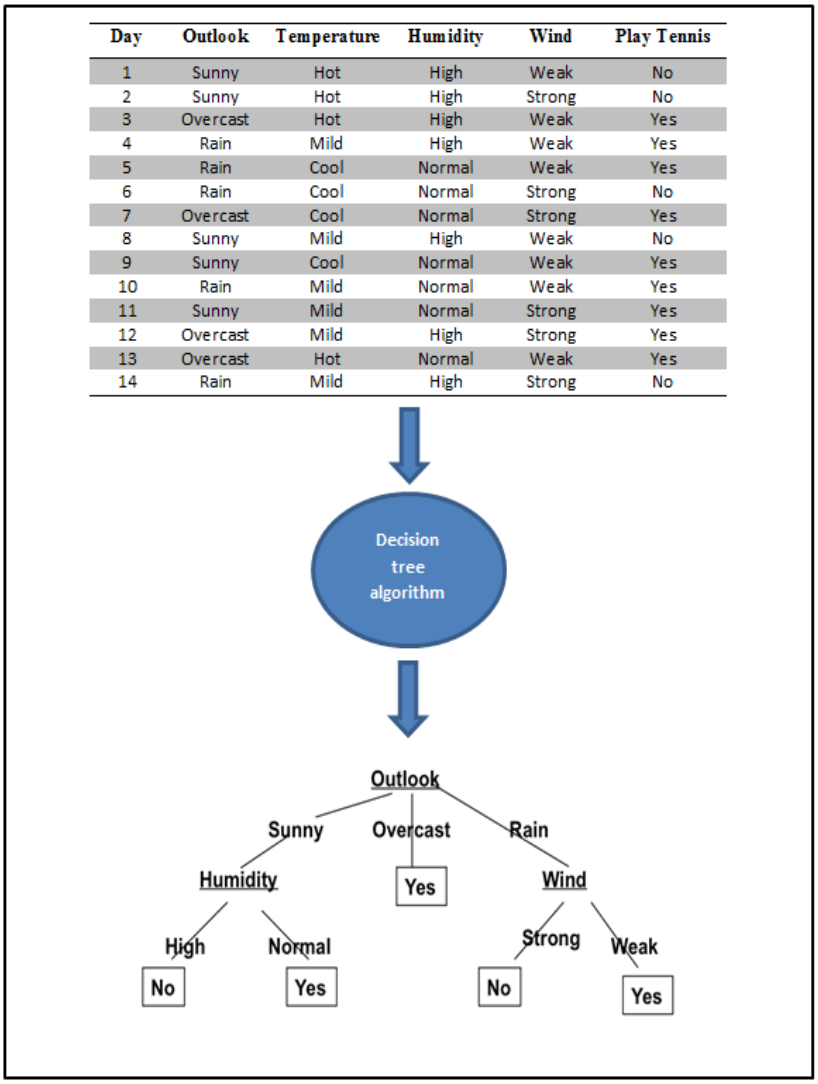

Figure 11: Example of a decision tree [3]

3.1.1 C4.5 decision tree classifier

In this section, we give an overview of the C4.5 algorithm, which is one of the algorithms that will be used in our experiment. The C4.5 classifier is considered to be one of the most famous

and widely used algorithms of generating decision trees. C4.5 is a single learner base algorithm that generates a decision tree, with unlimited number of paths within the node.

It is necessary to follow the next demands for working with C4.5 algorithm:

1. Each record from set of data should be associated with one of the offered classes. It means that one of the attributes should be considered as a class mark. It may be concluded that all the samples should belong to the same class, otherwise the mistakes are inevitable.
2. Each class should be discrete. Each sample should definitely belong to one of the classes.
3. In the considered scope of data, the number of classes should be much fewer than the number of samples. One should understand that C4.5 algorithm works slowly with very large datasets.

Using the concept of Gain Ratio measure which is driven from the Information Gain in earlier algorithm called "ID3", C4.5 builds the decision trees on the basis of the set of data [**53**]. The main advantage of Gain Ratio measure is to avoid overfitting the data. Next, we explain the basic algorithm for inducing a decision tree from training tuples in order to understand C4.5 algorithm [**6**].

The ID3 algorithm is adapted from divide and conquer algorithm. In Figure 11, Kamber presents the basic algorithm of a decision tree. This algorithm uses a straightforward strategy which depends on three variables as it shows.

- D refers to data partition
- Attribute_list refers to a description list for the tuples
  Attribute_Selection_ method refers to the used procedure to choose attribute that best identify the given tuples based on classes by applying the attribute selecting measures such as Information Gain in ID3 or Gain Ratio in C4.5

Gain Ratio measure execute some kind of normalization that change the tendency of the Information Gain measure which is used in this algorithm to attributes with more distinguishable values [**51**]. This Gain Ratio measure is defined as follows:

$$\boldsymbol{Gain\ Ratio\ (A)} = \frac{\boldsymbol{Gain\ (A)}}{\textbf{Split Info (A)}} \tag{3.1}$$

Basic algorithm for inducing a decision tree from training tuples

Algorithm: generate decision tree. *Generate a decision tree from the training tuples of data partition D .*

Input*:*

- *Data partition, D , which is a set of training tuples and their associated class label;*
- *Attribute_list, the set of candidate attributes;*
- *Attribute_selection_method, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of splitting_attribute and, possibly, either a split point or splitting subset.*

Output*: a decision tree.*

Method*:*

(1) Create a node N ;

(2) ***If*** *tuples in D are all of the same class, C* ***then****;*

(3) Return N as a leaf node labelled with class C ;

(4) ***If*** *Attribute_list is empty* ***then****,*

(5) Return N as a leaf node labelled with the majority class in D ; // majority voting //

(6) Apply ***Attribute_selection_method*** *(D , attribute_list) to* ***find*** *the "best" spitting_criterion;*

(7) Label node N with splitting_criterion;

(8) ***If*** *splitting_attribute is discrete-value* ***and*** *multiple splits allowed* ***then*** *// not restricted to binary classes//*

(9) Attribute_list ← Attribute_list – splitting_attribute; // remove splitting_attribute //

(10) ***For each*** *outcome j of spitting_criterion // partition the tuples and grow sub tree for each partition //*

(11) Let D_j be the set of data tuples in D satisfying outcome j ; // a partition //

(12) ***If*** *D_j is empty* ***then***

(13) Attach a leaf labelled with the majority class in D to node N ;

(14) ***Else*** *attach the node returned by* ***Generate_decision_tree*** *(D_j , attribute_list) to node N;*

endfor

(15) *Return N*

Figure 12: Basic algorithm for inducing a decision tree from training tuples [6]

"Where Split Info represents the information due to dividing the dataset S into v subsets S_1, S_2, ..., S_v and each subset S_j contains those instances that have the value A_j of attribute A" [**54**].

$$\boldsymbol{Split\ Info\ (A)} = -\sum_{j=1}^{v} \frac{S_i}{S} \log_2 \left(\frac{S_i}{S}\right) \tag{3.2}$$

"The Split Information tends to increase as the number of an attribute outcome increases which therefore reduce the Gain Ratio to a reasonable number" [**54**]. There are two ways how this algorithm can generate decision trees: batch mode and iterative mode. Batch mode, often called default mode, generates a single decision tree. This tree covers all the data available for the decision. The other kind of this algorithm, iterative mode, is based on the random basis where the data is selected randomly. Then, a decision tree is generated with adding some specific objects which have been misclassified. The actions are repeated and the decision tree is continued until it is classified in a correct way or it is found out that there is no progress. Keeping in mind that there can be many different decision trees due to the multiple trials; therefore, the presence of the unpruned file stem is necessary. The file stem is created with the purpose of collecting the decision trees in the process. If the similar data is used for generating decision trees, the latest variant of the tree is used. The machine saves the best generated decision tree in the file stem tree [**54**].

Advantages of C4.5

The C4.5 classifier primary aims at finding small decision trees. Based on this, we can say that the decision trees produced are simple to understand. C4.5 can detect how deep to grow a decision tree in order to avoid overfitting the data, which makes C4.5 robust to noisy data [**55**]. Unavailable attribute values are accounted for in C4.5 by assessing the information gain using the records where the particular attribute is defined. A fourth advantage is the ability to handle both continuous and discrete attributes. By defining the attributes with a continuous range, we create partitions based on a predetermined pattern in the training set and calculate the information gain on each partition. Then the partition that maximizes the information gain is picked. Lastly, tree pruning after creation ensures tree simplicity by replacing some branches with leaf nodes [**51**].

Disadvantages of C4.5

One of the major disadvantages of this algorithm is the fact that a sub-tree can be replicated several times. Another disadvantage is that it does not work very well on a small training set which may lead to different decision trees especially when the attributes values are close to each other. Furthermore, preforming C4.5 on relational databases or redundant data warehouse may badly affect the construction of a decision tree because it contains irrelevant attributes [**51**].

3.2 KStar Classifier

KStar (K*) is an instance-based classifier which uses a distance-based measure to classify variables by examining their performance on a variety of problems. Instance-based classifiers such as KStar classify instances by comparing them to other pre-classified examples. The process assumes that similar instances usually have similar classification even though this poses a challenge in defining such instances and classifications. The distance measure employs the approach of computing distance between instances by using information theory. It therefore employs the intuition that such distances define the complexity of converting one instance into the other. This can be done in two processes, one of which involves defining finite set of transformations to map one instance on another. This can be done by adding termination symbol at the end of each string. The shortest distance between two instances defines the distance of measure. This leads to a distance that does not solve issues of smoothness since it is very sensitive to small changes [**56**].

KStar reduces this problem of high sensitivity and hence reduced smoothness by summing over all these transformations that exist between any two instances. However, this is also not very clear as which transformations are summed. In this regard, the effectual number of instances for any function can be calculated as follows [**56**].

$$\boldsymbol{n_0} \leq \frac{\left(\sum_b \boldsymbol{P^*(b/a)}\right)^2}{\sum_b \boldsymbol{P^*(b/a)^2}} \leq \boldsymbol{N} \tag{3.3}$$

Where *N* stands for the effective number of training instances and n_0 stands for the number of training instances at the smallest distance. Moreover, *a* represents the first blending parameter while *b* is the second blending parameter. The K* algorithm works to choose one value from x_0(s). To achieve this, it selects this number between N and n_0. At the end, it inverts the expression shown above [**56**].

This method is referred to as the Kolmogorov complexity and its summation satisfies the Kraft inequality. This allowed the second blending parameter *b* interpreted as the probability of generating a program through random selection of such transformations. It can also refer to the probability of arriving at an instance by random walk from the first instance. The units of complexity are therefore obtained by calculating its logarithms. This method has been found to bring out the most realistic and robust measure of the link to DNA sequence. Similarities are prominent in distance measures of real valued attributes. For example, probability instances

drops heavily with increase in distance when x_0 is small and therefore functions as a measure of nearest neighbor. However, in case x_0 is very large, then virtually all instances shall have similar transformations, which are equally weighted. In both cases the number of instances tends to vary from extreme 1, in which the distribution is the nearest neighbor, to that of extreme N, where the instances are equally weighted [**56**].

Advantages of KStar

Firstly, in terms of accuracy, this algorithm is comparable to C4.5 on voluminous UCI benchmark datasets. KStar performs better and with a higher speed than C4.5 on big numbers of text classifications. Thirdly, this algorithm is low in time complexity which means it is very fast. Its speed can be compared to Naive Bayes [**57**]. Another advantage is that it uses entropy as a measure of distance, hence, providing a consistent approach to management of symbolic attributes, unknown values and continuous value attributes. The results presented compare satisfactorily with many machine-learning algorithms. This algorithm is an instance-based classifier. It classifies an instance by comparison; an instance is compared to pre-classified examples stored in a database. New instances to be added to the instance database and choice of instances from the database to be used in classification are determined by a concept description updater. This helps reduce memory space requirements and also improves tolerance to the noise in data [**58**].

Disadvantages of KStar

One of the major disadvantages of this algorithm is the fact that it has to generate distance measures for all the recorded attributes [**57**]. Another disadvantage is that it is not cost effective. Both building and learning processes are quite expensive in any task except in the text classifications. In terms of memory space, as much as the algorithm uses an instance updater to select and determine the instances to be used, the storage of sample instances is also involved in terms of memory space. Although the algorithm can be speeded up by combining it with other algorithms, it is sometimes a disadvantage since the combination process involves extra costs [**58**].

3.3 Bayesian Network Classifier

A Bayesian Network (BN) is a network structure of a directed acyclic graph of a set of variables (U) and it represents a probability distribution of the variables (U) in a given set of data. A Bayesian Network has been considered as an advanced form of general Bayesian probability. This probability can be explained as an extension of logic that allows figuring out with propositions whose truth or falsity is uncertain [**6**]. Bayesian probability is defined as follows:

$$\boldsymbol{P(H|X)} = \frac{\boldsymbol{P(X|H)P(H)}}{\boldsymbol{P(X)}} \tag{3.4}$$

Here, X is considered as a tuples that consists of n attributes. H stands for a hypothesis that classify the tuples X belongs to a class C. For example, suppose our dataset confined to a patient described by the attributes age and smoker, respectively, X is a 60-years-old patient with (smoker = yes). Suppose that H is the hypothesis that our patient will get a lung cancer. Then $P(X|H)$ reflects the probability that patient X will get a lung cancer given that we know the patient's age and (smoker=yes) [**6**].

The Bayesian Network structure contains some assumptions. For instance, it assumes that all variables in the data set are discrete definite variables and if not, they are 'discredited' first before rule application. Another assumption is that, the data set has no missing variables and if there are, the missing values are filled [**60**]. The initial step in application of the network entails the confirmation of whether the data set meets all the assumptions and if not met, an automatic filtering becomes necessary. There are two processing stages in Bayesian Network learning: first to learn a network structure, and then to learn the probability tables [**61**].

Learning a network structure can be considered an optimization problem where, a quality measure of a network structure given the training data needs to be maximized. We call this procedure the local score metrics. The quality measure can be based on a Bayesian approach, minimum description length, information and other criteria such as log likelihood. These metrics have the practical property that the score of the whole network can be decomposed as the product of the score of the individual nodes. This allows for local scoring and thus local search methods [**61**].

A conditional independence test is a task that stems from the goal of uncovering causal structure. The assumption is that there is a network structure that exactly represents the

independencies in the distribution that generated the data. Then it follows that if a (conditional) independency can be identified in the data between two variables, there is no arrow between those two variables. Once locations of edges are identified, the direction of the edges is assigned such that conditional independencies in the data are properly represented. In using Bayesian network as a classifier, one simply calculates the ($\text{argmax}_y P(y|x)$) using the distribution P (U) of the Bayesian structure as follows [**61**]:

$$\text{argmax}_y \boldsymbol{P} = \boldsymbol{P}(\boldsymbol{y}|\boldsymbol{x}) \tag{3.5}$$

These variables are in a probability distribution as described as $P(y|x) = P(U) / P(x)$. Successful application of this rule involves first learning the network structure then the probability tables. This calls for the use of different approaches such as local score metrics, conditional independence tests, global score metrics, and fixed structure strategies [**60**]. Each case here uses a different search algorithm; for instance, annealing, hill climbing, simulated and taboo searches in aid of searching for a good network structure. This opens the way for estimation of a conditional probability table for each variable [**61**].

Bayesian Network Advantages

The first advantage of this classifier is its computational efficiency. The representation of large and complex computational problems is decomposed into smaller and simpler self-sufficient models to enhance efficiency. The EM algorithm refers to Expectation – Maximization (EM) which is "an iterative method that attempts to find the maximum likelihood estimator of a parameter θ of a parametric probability distribution" [**62**]. Secondly, the natural combination of the EM algorithm and the probabilistic representation helps address the problems with missing data. The ability of this classifier to indicate all the possible classes of a new sample to which might belongs is another advantage in case of a misclassification [**63**].

Another advantage is the presence of an updater that enables the classifier to learn from new data samples. Bayesian networks have the advantage of being able to capture the complexity of decision making [**64**]. Lastly, the system's rules of classification can be semantically determined and justified to both the novice and the expert [**85**].

Bayesian Network Disadvantages

The mismatch between the data likelihood and the actual label prediction accuracy tends to make the learning method suboptimal. Secondly, Bayesian networks require an expert to give domain information for the creation of the network [**64**]. Despite the substantial amount of research carried out, the creation of these networks is still limited to data sets consisting of only a few variables which are very informative. Another disadvantage of these networks is the fact that their interpretation and efficiency is quite limited when rule sets are drawn from the network. In comparison to rules derived from decision trees whose interpretation is simple and direct, Bayesian networks are more complex [**86**].

3.4 Fast Effective Rule Induction (JRip classifier)

JRip is a classifier that uses learning rule sets which are easy to understand by regular users. The main motive of this classifier is to overcome the weakness of handling large noisy datasets that other rules sets systems cannot deal with. Therefore, Incremental reduced error pruning (IREP) or (its enhanced version JRip in WEKA) consolidates reduced error pruning and separate-and-conquer rules in learning algorithm aims to decrease the unacceptable large errors in large noisy datasets. It is important to understand the phases of the improvement that have been done on REP before we explain JRip. There are four enhancements that have been added to REP. First, REP which is the basic version improved to IREP. The second enhancement is improving IREP to IREP*. The third enhancement is enhancing IREP to RIPPER. Finally, RIPPER improves to RIPPERk or (JRip in WEKA). Therefore, In order to understand JRip, we have to explain IREP first and then explain JRip because JRip is an improved version of IREP [**67**].

Figure 13 shows the original algorithm of IRIP which works as follows: In creating a rule of IREP, randomising the variables into two subsets is necessary, where either variable is in growing sets or pruning sets. This allows the formulation of the Grow rule, which starts with empty conjunction of conditions and keeps on adding any condition if nominal attributes equals legal values added. Addition of the conditions maximizes the FOIL information until the rule covers no negative variables in the grow data set [**67**]. The FOIL refers to First Order Inductive Learner which is a rule-based learning algorithm [**66**].

The IRIP algorithm

```
procedure IREP (Pos, Neg)
    begin
    Ruleset : = θ
      while Pos ≠ θ do
        /* grow and prune a new rule */
        split (Pos, Neg) into (GrowPos, GrowNeg)
        and (PrunePos, PruneNeg)
        Rule : =  GrowRule (GrowPos, GrowNeg)
        Rule : =PruneRule (Rule, PrunePos, PruneNeg)
            If  the error rate of Rule on (PrunePos,PruneNeg) exceeds 50% then
                return Rule set
            else
                add Rule to Rule set
                remove examples covered by Rule from (Pos,Neg)
            endif
      endwhile
    return Ruleset
    end
```

Figure 13: The IRIP algorithm [67]

The generation of the rule allows for the pruning of the rule which involves deleting of some conditions to maximize the function. The deletion continues until further deletion does not affect the value of V [**67**].

$$\mathbf{v}\,(\mathbf{Rule}, \mathbf{PrunePos}, \mathbf{PruneNeg}) = \boldsymbol{p} + \frac{\boldsymbol{p}+(\boldsymbol{N}-\boldsymbol{n})}{\boldsymbol{P}+\boldsymbol{N}} \qquad (3.6)$$

Where v is (Rule, PrunePos, pruneNeg) and $\boldsymbol{P}$ (respectively N) represents the total example in PrunePos (PruneNeg). The total number of examples $\boldsymbol{P}(\boldsymbol{n})$ in PrunePos covered by rule sets that created by IREP. In other words, IREP creates rule sets until new rule results in too large error rate; then IREP randomly divides the data into a growing set and a pruning set. IREP grows rule from the growing set and prunes rule immediately. IREP deletes the final sequence of conditions that maximizes function v until no deletion increases the value of v. IREP adds pruned rule to the rule set in addition to deleting every example covered by rule$(\boldsymbol{p}/\boldsymbol{n})$. Cohen [**67**] adds improvements to IREP which results in the enhanced version of IREP*. The main goal of this improvement is handling large noisy data sets to be competitive with C4.5. There are

three main improvements that have been added to IREP. First, in pruning phase, alternative metric has been added because the old metric leads to occasional failure to converge [**67**]. The new metric becomes as follows:

$$\mathbf{v} * (\mathbf{Rule}, \mathbf{PrunePos}, \mathbf{PruneNeg}) = \frac{P-n}{P+n} \tag{3.7}$$

Second, for rule adding, new stopping heuristics have been added to IREP*because half of heuristics often stops too soon with moderate sized examples in addition to the sensitively to the small disjunction problem. The new version is called REPPER. The modification that Cohen added is after adding each rule; REPPER calculates the total description length of the rule set and examples. REPPER stops adding rules if the total description length is d bits higher than the smallest length so far. In REPPER implementation the value of d is 64 bits. The third improvement in IREP* is in the rule optimization, because IREP uses the repeated grow-and-simplify methods which cause different results from conventional reduced error pruning. The modification is in the post-prunes rules in IREP. JRip constructs two alternatives rules: R'_i (new rule) and R''_i (based on R'_i). The final rule is selected based on minimum description length [**67**].

Advantages of JRip classifier

The first advantage of this classifier is the use of propositional rule learner which ensures that errors are minimal. Also, the phase-by-phase implementation of the algorithm ensures that the overall results are as high as possible. During the rule set growing phase, the condition with the highest information gain is picked to ensure that the rule set is correct [**68**]. Another advantage is that the algorithm is made shorter and simpler by pruning unused parts of a rule. This makes the classifier easy to understand and interpret. The ease of generation of this classifier is a fourth advantage. This classifier is highly expressive; it is comparable to a decision tree. Even in terms of performance, it can be ranked the same level as a decision tree. JRip classifier has the ability to classify new instances rapidly. Lastly, it is easy for this classifier to handle missing values as well as numeric attributes [**2**].

Disadvantages of JRip classifier

The JRip algorithm requires a large investment in terms of time to learn the algorithm and test the features that can be customized. The fact that the JRip algorithm uses some induced rules which sometimes have to be replaced with expert-derived rules for some applications is another

disadvantage. In addition, the accuracy of JRip's results sometimes varies because the results produced differ depending on the option of the rule voting method used. Another disadvantage is that, highly accurate results can only be achieved by running the algorithm many times. Assignment of salience which is an order of that has been prescribed for firing rules may lend the expert system's inference engine powerless. This may also have negative effects on the performance of such a system which is rule-based [**69**].

3.5 PART (Partial decision tree) classifier

The PART algorithm integrates both C4.5 and the JRip algorithms in order to obtain better prediction. PART refers to partial decision tree classifier which is "an ordinary decision tree that contains branches to undefined sub trees" [**69**]. The main advantage in PART is that it does not require global optimization in order to produces accurate rule sets. PART builds a rule and deletes the instances it covers by using the separate and conquer strategy. PART continues repeating this strategy in the remaining instances until none is left. Applying the separate and conquer strategy gives PART flexibility and speed because PART uses this strategy in conjunction with decision trees [**69**].

In order to produce one single rule from the set current instances, PART builds a pruned decision tree to detect the leaf with the largest coverage. PART generates the single rule the leaf with the largest coverage and then it discards this tree. This strategy enables PART to avoid a tendency to over-prune which is a common problem of the basic separate-and-conquer rule learner. PART avoids fully exploring a partial decision tree by creating one. To create a partial tree, PART aims to find a stable tree that cannot be simplified further. In doing so, PART integrates construction and pruning operations. Figure 14 presents the methods to create a partial tree [**69**].

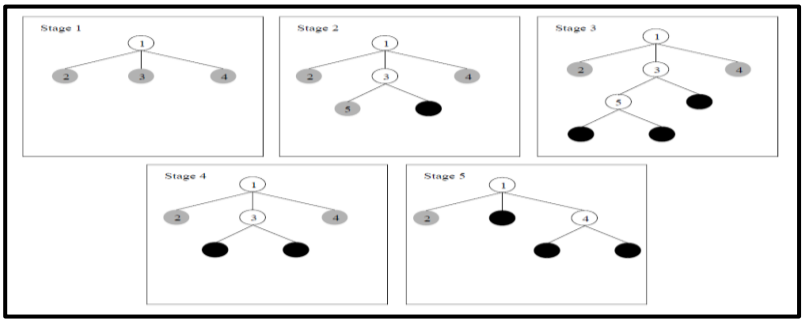

Figure 14: Example of how the algorithm builds a partial tree [69**]**

As Figure 15 shows, PART divides a set of examples repeatedly into a partial tree. PART uses the same mechanism that C4.5 uses in term of selecting a test and splits the examples into subsets accordingly. When PART finishes splitting the examples into subsets, PART expands the subsets continuously in order, starting from the smallest, based on their average entropy. PART repeats the expansion until a subset is expanded into a leaf. Once a subset is expanded into a leaf, PART continues by backtracking until internal nodes appear. PART starts pruning using the same mechanism that C4.5 uses which is called "sub-tree replacement" which ensure that either a node is better or it is replaced by a single leaf. If PART performs a replacement, PART backtracks to explore siblings of the newly replaced node. However, if a node is faced all whose children are not leaves during the backtracking, the rest sub-trees are left unexplored and the corresponding sub-trees are left undefined [**69**].

Method that expands a given set of examples into a partial tree

Procedure Expand subset

- Choose spilt of a given set of example into subsets
- **While** there are subsets that have not been expanded **and** all the subsets expanded so far are leaves
  - choose next subset to be expanded and expand it
  - **if** all the subsets expanded are leaves and estimated error for sub-tree ≥ estimated error for node
  - **undo** expansion into subsets and make node a leaf

Figure 15: PART Algorithm [69**]**

Advantages of PART Classifier

Most of rule learners operate in two stages. However, PART shows that good rule sets can be learned one rule at a time. PART works firstly by inducing an initial rule set and then PART refines the rule set using a complex optimization stage that discards individual rules to make them work better together. Another advantage of this classifier is simplicity which makes this classifier easy to understand and flexible in term of determining the places of the concept drift by using the divide and conquer methodology in conjunction with decision trees. Additionally, this classifier does not require a global optimization to produce accurate rule sets. Moreover, PART avoids the over pruning problem of the basic separate and conquer rule learner [**69**].

Disadvantages of PART Classifier

The main disadvantage of PART is accurate results can only be achieved by running the algorithm many times. PART constructs a partial tree that deals with missing data. This partial tree assigns to each of the branches a proportional weight. The weight proportional assigns to the number of training instances going down that branch normalized by the total number of training instances with known values at the node [**69**].

3.6 K-Nearest Neighbour Algorithm

Algorithms such as KStar (K*) and K-Nearest Neighbour (KNN) refer to instance-based learners that apply entropy in their distance measure. The classification task of a new object is based on attributes and training samples. The task is very involving as some data become noisy and can as well have irrelevant attributes which makes it difficult to learn from. In the K-nearest neighbour appearance, the result of new instance query is classified built on majority of K-nearest neighbour category [**70**].

Teknomo [**71**] presents an example to illustrate how KNN works. Suppose we have a dataset of a special paper tissue that we want to classify whether it is good or not, based on two attributes, namely acid durability and strength. Table 2 presents four training samples.

X_1 = **Acid Durability (seconds)**	X_2 = **Strength (kg/square meter)**	**Classification**
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

Table 2: Training samples of a special paper tissue [66**]**

Suppose a factory passes a laboratory test with $X_1 = 3$ and $X_2 = 7$ in order to produces a new paper tissue. The K-nearest neighbour (KNN) algorithm can predict this type of problem without another expensive survey. The K nearest neighbour algorithm works "based on minimum distance from the query instance to the training samples to determine the K-nearest neighbours. After we gather K nearest neighbours, we take simple majority of these K-nearest neighbours to be the prediction of the query instance" [**71**].

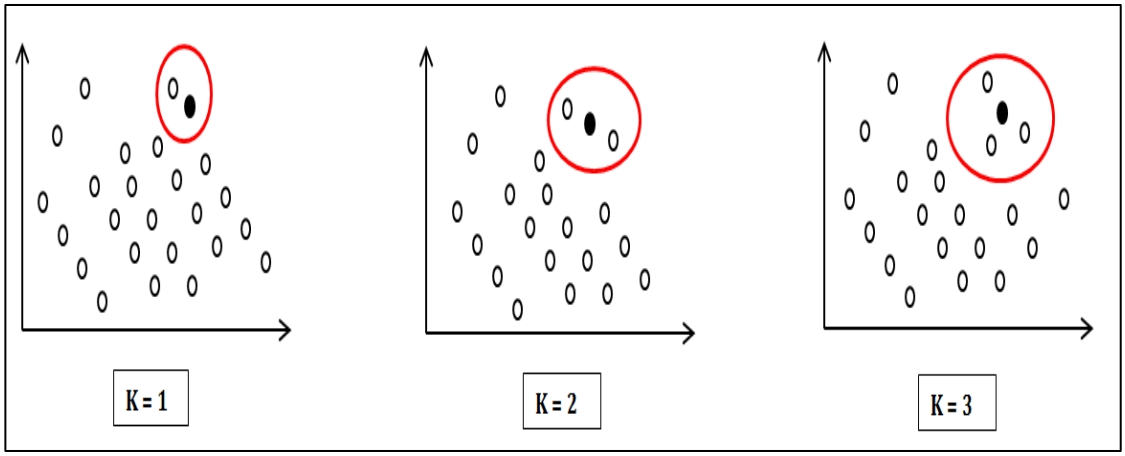

Figure 16: K-Nearest neighbour example [88]

The K-nearest neighbour algorithm is usually used for classification besides prediction and estimation. It gives a proper example of instance-based learning, which stores data. It does this to obtain classification for unclassified records which are new. To do this, it compares such records with those similar in the training set. In dealing with this classifier, several issues must be considered. These include the number of neighbours that one should consider, for instance, determination of *k*, since *k* represents the nearest neighbours. It also involves other issues such as how to measure the distance from the nearest neighbours as well as combining information from all the observations. The algorithm also involves determination of whether points should be weighted equally or not [**72**].

K-Nearest neighbour algorithm

Input: D, the set of k training objects, and test object $z = (\mathbf{x}', y')$

Process:

Compute $d(\mathbf{x}', \mathbf{x})$, the distance between z and every object, $(\mathbf{x}, y) \in D$.

Select $D_z \subseteq D$, the set of k closest training objects to z.

Output: $y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$

Figure 17: K-Nearest neighbour algorithm

*K*NN classifies new objects by using a majority vote of its neighbours that the new object has been assigned to the class that is the most similar amongst its k-nearest neighbours [**70**]. The K-nearest neighbour algorithm may also be used for prediction and estimation. This

may also include its use for continuous valued target variables. This might be achieved through using a locally weighted averaging method, among others. The weighted method can be achieved through multiplying the class of each of the K-nearest points by a weight proportional to the inverse of the distance between that point and the point for which the class is to be predicted [**72**].

In most cases, it would be assumed that neighbours closest to the new record should be considered more than those far away and thus weighted heavily. However, analysts tend to apply weighted voting which has the propensity to reduce ties. Several algorithms may be employed in classification of objects. In K-nearest neighbour classification, one considers the number of nearest similar variables to classify, predict or estimate its performance. This can be utilized in situations such as administering drugs to patients. By using known classifications, one can classify an unknown object by using the known ones to classify and estimate or predict its behaviour [**72**].

Careful considerations should be taken when choosing *K* in classifying variables. This is mainly because choosing a small *K* may result in problems such as noise. On the other hand, *K* that is not very small may smoothen out idiosyncratic behaviours which may be learned from the training set. Moreover, taking a larger *K* also has the probability of overlooking locally interesting behaviour [**72**].

Advantages of K-nearest neighbour classifier

K-nearest neighbour algorithm is robust to noisy training data and it is simple to implement. It is also very easy to use in parallel implementations. Another advantage of the K-Nearest neighbour algorithm is the simplicity of learning. In addition, the training speed is very high and the results are nearly optimal in the case of a large sample limit. This means that the algorithm is more effective if the set of training data is large [**73**]. The ability of this algorithm to approximate complex concepts of the target locally and also differently for every new instance is another advantage of this algorithm [**74**].

Disadvantages of K-nearest neighbour classifier

One disadvantage of this algorithm is that it has large memory space requirements. This is because of the needs to store all the data and hence the need for large memory space. Secondly,

during instance classification, all training occurrences have to be visited and this makes the algorithm slow during this procedure. A third disadvantage is that the algorithm is easily misled by irrelevant data attributes; this means that the accuracy decreases with an increase in the number of irrelevant data attributes. Also, the fact that the accuracy decreases with increased noise in the set of training data is another disadvantage. Another shortcoming of this algorithm is its computational complexity. This algorithm is highly biased by the value of *K* [**73**].

3.7 SMO Classifier: Sequential Minimal Optimization

The sequential minimal optimization is an algorithm that has been devised and proposed by Platt [**75**] to provide training for support vector machines (SVM). The Quadratic programming problem refers to the optimization problem of minimizing or maximizing a quadratic function of many parameters subject to linear constraints on these variables [**76**].

Normally, support vector machines will require large investment of quadratic a program during the process of training. The amount of programming that goes into a vector machine during the process of training determines the effectiveness of the vector machine in question. Optimization of problems in quadratic program is one of the key elements in training support vector machines. How well functional the support vector machine is usually dependent on the level of effectiveness employed in the solving process in quadratic programming. Usually, large quadratic programming problems are more difficult to get through compared to small quadratic programming problems. One of the best ways to deal with large quadratic programming problems and other similar problems is to break these problems into simpler quadratic programming problems [**77**].

The standard SVM training algorithm has been used as the standard in designing the different concepts involved in the SMO. The SMO is easy to implement, conceptually simple, faster compared to other SVM algorithms, and has better scaling properties for the different difficult SVM problems that are involved in the standards SVM training algorithms. This system was designed to ensure that the standard SVM system is made better using better methods. The analytic quadratic programming step is an important element in relation to presenting the difference that exists between the SVM's and the training algorithms. It is important to have some background information on the SVM algorithms to understand how the SMO system works [**77**] .

The SMO system is able to break down large and extensive quadratic programming problems into the smallest possible quadratic problems. Here, the system works endlessly to ensure that the problems are broken down into the smallest problems there can ever be. Unlike the large quadratic problems, the large quadratic problems are usually solved analytically. Solving these simpler problems analytically is more time efficient as compared to trying to deal with the large and extensive quadratic programming. Compared to other systems, this system is efficient in relation to the amount of memory it dedicates to the operations involved in solving the different problems. In the training, the amount of memory that is required for the SMO to function effectively is set size. This ensures that the system is able to deal with large and extensive problems efficiently while employing set size memory [**78**].

The system is able to deal with large quadratic programming optimization problems effectively without employing matrix computation at any one of the processes or stages. The various test problems that are presented in training are able to get solved through linear and quadratic computation. Operating somewhere in between the quadratic and linear set sizes in training is an important optimization feature. Sparse data sheets and linear SVM's have now become easy to deal with. Unlike in the standard chunking where SVM algorithm function somewhere in between linear and cubic in the usual training set size, this system's computation time is focused and based on the SVM evaluation. This is a major advantage of the system compared to other SVM algorithm. In the development of SMO, speed was an important factor to take into consideration. A closer look at how the system works on world sparse data sets is evidence of the speed in this system. Here, the SMO is able to deliver a speed that is over a thousand times faster compared to the traditional chunking algorism [**78**].

The SVM algorithms work by employing various numerical optimization steps. This is usually done for the purpose of overall optimization of the quadratic programming problem. The analysis of the SMO also presents a number of important things in relation to the real world and artificial problems. Whenever a user wants to access a quadratic programming package without necessarily tuning up the quadratic programming package, they can have access through employing SMO's. The SMO's have also proved to perform better on SVM's. This is true in cases where there are many Lagrange multipliers which are bound. This is an important aspect to take into consideration in relation to quadratic programming. The evaluation in SMO's is also an important factor to take into consideration. SMO's will perform well on linear SVM's due to the manner in which the computation takes place. Most of the computation in SMO's takes place through SVM dominated evaluation [**79**].

Advantages of SMO classifier

SMO's have the ability to perform well in most of the large problems due to the fact that they are able to improve scaling as opposed to other methods. The SMO system is able to break down large and extensive quadratic programming problems into the smallest possible quadratic problems. Here, the system works to ensure that the problems are broken down into the smallest problems. In the training, the set size of memory is required for the SMO to function effectively. This ensures that the system is able to deal with large and extensive problems efficiently while employing set size memory [**78**].

The SMO's have the ability to overcome the challenges presented by the set size in training. Even for non-linear SVM's which have sparse inputs, SMO's are able to perform well. This is mainly attributed to the kernel computations time. While working with SMO's it is possible to reduce the time that is involved in the kernel computation and this directly speeds up the SMO. This is one of the key issues in relation to the efficiency of SMO's. SMO's are able to exploit both the sparseness of the input data as well as linearity of the entire SVM. Unlike in chunking where most of the time is spent working in the quadratic programming, SMO's has a better way of working around this process to improve the time involved and the quality of the end product. SMO's stand out when it comes to the SVM training algorithms. There are a number of other options but SMO's stand out as the best candidate for the standard SVM training algorithm. Here, SMO's stand out due to the manner in which they can be used in scaling and training set sizes. There is ease of use as well as better scaling with training set size when it comes to the use of SMO's [**79**].

Disadvantages of SMO classifier

Support vector machines are applicable in a number of systems. These machines are fundamental in relation to solving some of the problems that are encountered in the day to day problems. Some of the day to day problems that involve the use of support vector machines include face detection, character recognition, text categorization, and pedestrian detection. All these systems are important and define some of the important day to day processes. Support vector machines have an expanded application in relation processes involving large quadratic programming. There are a number of challenges presented in relation to the use of these algorithms. SVM's are usually slow. This reduces their efficiency especially in relation to large

problems. Another major challenge facing these systems is in related to their implementation, because they are subtle and very complex [**80**].

3.8 Discussion

In this section, we will discuss the classifiers qualities in term of the advantages and disadvantages. In the process of developing and improving on machine learning, "machine learning" engineers have adopted several approaches in obtaining sources of information on machine learning. Among the important information sources that are applicable in machine learning include statistics. The quality of a classifier can be based on five criteria: accuracy, computational speed, robustness, scalability, and interpretability [81].

In Table 3, we listed some of the general advantages and disadvantages for each classifier. The main advantages of C4.5 are scalability, simplicity and the effective methods of handling missing data in addition to excellent performance in term of the accuracy. Replicating Sub-tree and the small training sets are the main drawback of C4.5. JRip handles large noisy datasets efficiently by using propositional rule learner. Moreover, representing the rules in first order logic makes JRip easy to understand. JRip faces a challenge of handling large training set which results in poor scalability.

One of the benefits of PART is avoiding the over pruning problem of the basic separate and conquer rule learner which results in high scalability in its performance. On the other hand, dealing with missing data effectively in PART is a quite challenge. Moreover, accurate results can only be achieved by running the algorithm many times. C4.5, JRip, and PART are the most scalable classifiers among the reviewed classifiers. Furthermore, these classifiers have been considered among the most accurate classifiers in the data mining society.

KNN, KStar, SMO and BayesNet are considered among the most accurate classifiers. However, in term of the scalability, they are considered as time consumers. In term of the algorithms easiness, C4.5, JRip, PART, and BayesNet are considered as easy to understand and to implement.

Classifiers	Advantages	Disadvantages
C4.5 Classifier	▪ Small, scalable, and simple ▪ Effective handling the missing values ▪ High accuracy	▪ Sub-tree can be replicated many times ▪ It does not work very well on a small training set.
JRip Classifier	▪ Use of propositional rule learner to efficiently handle large noisy datasets ▪ Usually better than decision Tree learners ▪ Representable in first order logic which make it easy to implement in Prolog ▪ Prior knowledge can be added and Easy to understand	▪ Highly accurate results can only be achieved by running the algorithm many times ▪ Scale poorly with large training set size ▪ Problems with noisy data likely in real-world data
PART Classifier	▪ Avoids the over pruning problem of the basic separate and conquer rule learner ▪ Using the divide and conquer methodology ▪ Scalable ▪ Easiness	▪ Not effective dealing with missing data ▪ Accurate results can only be achieved by running the algorithm many times
Bayes Network Classifier	▪ Its computational efficiency ▪ Simplifies the incorporation of domain knowledge ▪ The ability of this classifier to indicate all the possible classes a new sample ▪ Updater that enables the classifier to learn from new data samples ▪ It is easy to tell which other class the sample may belong	▪ The mismatch between the data likelihood and the actual label prediction accuracy ▪ The creation of these networks is still limited to data sets consisting of only a few variables which are very informative ▪ Require an expert to give domain information for the creation of the network
KStar Classifier	▪ In terms of accuracy, KStar is comparable to C4.5 ▪ Management of symbolic attributes, unknown values and continuous value attributes	▪ Large memory space requirements ▪ Generate distance measures for all the recorded attributes
K-nearest neighbour Classifier	▪ Highly adaptive behaviour ▪ Simple to implement ▪ Avails a generalisation accuracy that is favourable on many domains because it has Nonparametric architecture ▪ The ability of this algorithm to approximate complex concepts ▪ Effective if the set of training data is large ▪ Requires no training time	▪ Its computational complexity ▪ The accuracy decreases with increased noise in the set of training data ▪ Easily fooled by irrelevant data attributes ▪ Slow ▪ Large memory space requirements
SMO Classifier	▪ Avoiding the matrix computation makes SMO scalable compared to other SVM algorithms ▪ SMO is able to deal with large and extensive problems efficiently because in the training, the amount of memory that is required for the SMO to function effective is set size ▪ Easy to implement compared to other SVM algorithms	▪ Hard to implement SVM training algorithms because they are subtle and very complex

Table 3: Summary of the advantages and disadvantages of the reviewed classifiers

The main advantage of BayesNet is the ability of indicating all the possible classes a new sample by an updater that enables the classifier to learn from new data samples. KStar provides high accuracy which is comparable to C4.5 in addition to a good management of symbolic attributes, unknown values and continuous value attributes. In contrast, the main drawback of KStar is generating distance measures for all the recorded attributes which results in large memory space requirements. The main advantage of KNN is the large training sets it is able to handle. On the other hand, KNN consumes large memory space. The main advantage of SMO is that avoiding the matrix computation makes SMO scalable compared to other SVM algorithms. Another advantage of SMO is the ability to deal with large and extensive problems efficiently because in the training, the amount of memory that is required for the SMO to function effective is set size. The main challenge of SMO and SVM algorithms in general is the difficulty in implementation.

3.9 Summary

This chapter overviewed some classifiers that will be used further on in this study. The discussed classifiers have been presented with regards to the various types of machine learning processes, theoretical works that address these classifiers. We learn the basic techniques of these classifiers and how these classifiers work. We overviewed C4.5, JRip, Bayes Network, PART, and KStar in addition to K-nearest neighbour.

Moreover, we overviewed the advantages and disadvantages of each classifier that will be used in this thesis. This analysis was performed in order to aid us to understand the behaviour of these techniques when addressing the concept drift problem. In the next chapter, we will describe the thesis methodology, which creates an efficient algorithm called DCCD to handle gradual and sudden concept drift. Furthermore, the next chapter also presents the method that used to ensure that our approach provide high quality results. The next chapter presents a new algorithm called Concept Drift Introducer algorithm (CDI) to introduce gradual and sudden concept drift scenarios into datasets.

CHAPTER FOUR

DYNAMIC COMMITTEES FOR HANDLING CONCEPT DRIFTS IN DATABASES (DCCD)

CHAPTER FOUR

DYNAMIC COMMITTEES FOR HANDLING CONCEPT DRIFT IN DATABASES (DCCD)

This chapter highlights our thesis methodology, where we aim to create an efficient algorithm to handle gradual and sudden concept drifts. To this end, we design an algorithm called Dynamic Committee for Concept Drift (DCCD). A thorough description of our algorithm is provided in this chapter. Moreover, we introduce another new algorithm called Concept Drift Introducer (CDI) to insert gradual and sudden concept drift scenarios into datasets. This chapter also highlights the method that we used to ensure that our approach provide high quality results. Figure 18 shows the steps we followed in this study.

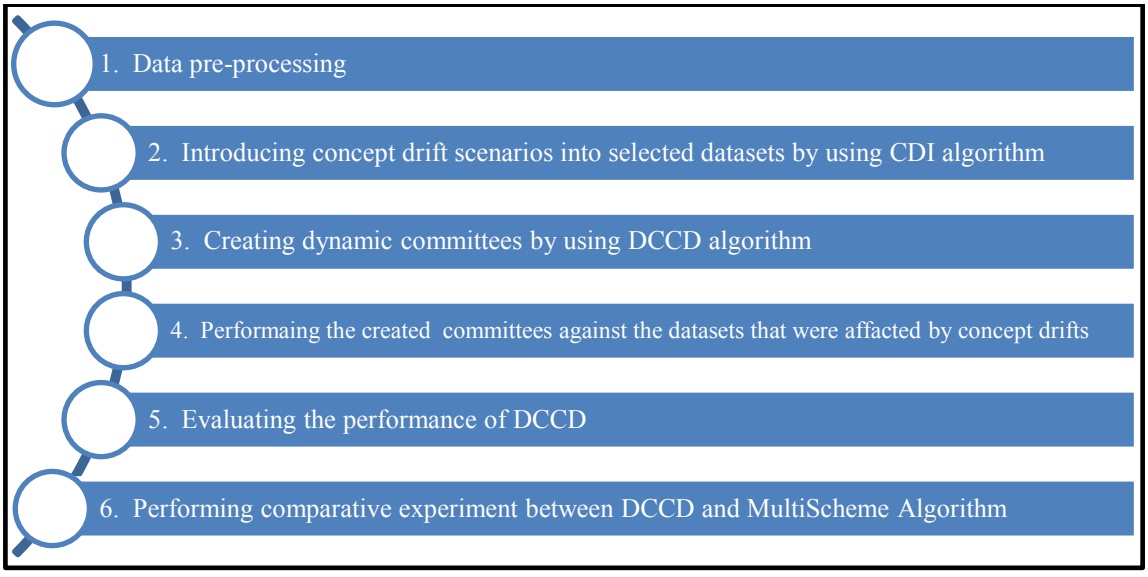

Figure 18: Steps of our methodology

4.1 Data Pre-processing

In this thesis, several datasets will be used in the experiment; hence, data pre-processing is necessary. Pre-processing our datasets is important for several reasons such as to ensure that all values in our datasets are accurate and complete. Some datasets have missing values. Based on the attribute type, missing values might be replaced with the mean or the mode of the attribute values. Furthermore, one of our datasets contains noisy data which may contain errors or, inconsistent values or outlier values. Noisy, missing, and inconsistent data occur for many reasons such as incorrect data collection. Another reason might be to avoid human mistakes at the data entry [**6**].

4.2 Introducing Concept Drift into Datasets

Having a good understanding about any dataset in a concept drift experiment is really an essential aspect. The reason for this is that, when we want to apply any change to the dataset; this change must be consistent with the dataset because this change might affect the final results. Therefore, we present a new algorithm called Concept Drift Introducer (CDI) to introduce gradual and sudden concept drift scenarios into any dataset. ***The main goal of CDI is to prepare experiments to test concept drift systems***. CDI generates gradual and sudden concept drift scenarios randomly and then introducing them into the dataset. Based on real-world databases, CDI also create partially artificial data sets that affected by concept drift scenarios.

The first step in CDI is selecting the dataset which we would like to introduce concept drift scenarios into it. The user must specify the target attribute (where concept drift scenarios will appear on it) to change its data distribution. Moreover, the user must specify the original value which needs to be changed by a new value in order to perform the drift scenarios. The second step of the CDI algorithm is specifying the number of gradual and sudden drift scenarios in the current phase. In order to ensure that the sequence of the drift scenarios is accurate, CDI divides the number of drift scenarios into a number of phases. The main advantage of using phases is providing high scalability to our main algorithm (DCCD) by evaluating each phase instead of evaluating each drift scenario. At the end, CDI change the data distribution of the target attribute by 10% for the gradual scenario and by 50% for the sudden. Figure 19 presents the Concept Drift Introducer Algorithm.

Concept Drift Introducer Algorithm (CDI)

Input:

- D, Dataset;
- NP, the number of phases;
- NDP, The number of drifts in each phase;
- TAN, Target Attribute number; (the introduced concept drift will take place on this attribute)
- TAV, Target Attribute Values; (it will be changed in order to introduce concept drift)
- NV, New value; (the changed value)
- NGCD, The number of Gradual Concept Drift scenarios for the current phase;
- NSCD, The number of Sudden Concept Drift scenarios for the current phase;
- Counter1 , integer default value = 1;
- Counter2 , integer default value = 1;

Output: Introducing (Gradual and Sudden) concept drift scenarios into a dataset

Method:

(1) *Add dataset(D);*
(2) *Setup the number of phases (NP);*
(3) *Enter Target Attribute number (TAN);*
(4) *Enter Target Attribute Values (TAV);*
(5) *Enter New value (NV);*
(6) *Enter The number of drifts in each phase (NDP);*
(7) ***While*** *counter1 =<the number of phases (NP)* ***do***

{
(8) *Enter* The number of Gradual Concept Drift scenarios for the current phase *(NGCD);*
(9) *Enter* The number of Sudden Concept Drift scenarios for the current phase *(NSCD);*
(10) ***While*** *counter2*=<The number of drifts in each phase *(NDP)* ***do***

{
(11) *Introduce the Gradual Concept Drift scenarios for the current phase;*
(12) ***Alter*** *Target Attribute Values (TAV)* ***To*** *the New value (NV);*
(13) *Introduce Sudden Concept Drift scenarios for the current phase ;*
(14) ***Alter*** *Target Attribute Values (TAV)* ***To*** *the New value (NV);*
(15) *Counter2 ++*
(16) ***End while***
}

(17) *Counter1 ++*
(18) ***End while***

}

Figure 19: Concept Drift Introducer Algorithm (CDI)

4.3 Dynamic Committees for Handling Concept Drifts (DCCD)

We present an algorithm to handle gradual and sudden concept drift by creating a committee that consists of diverse base classifiers. Recall that our algorithm is called Dynamic Committee for handling Concept Drifts in databases (DCCD) which can limit the accuracy reduction that concept drifts causes. Nowadays, the need for merging diverse base learners to increase the competition between the classifiers is significantly essential to increase the overall accuracy of the system. Moreover, such merging can limit the inappropriate selection of base classifiers that might be done by human. DCCD mainly proposes handling gradual and sudden concept drifts in databases. It extends the Committee of Decision Trees algorithm [**9**] which was discussed in Chapter 2. In our work, we used the idea of replacing the weakest member in the committee with a new member.

For illustration, suppose we select Bagging as the ensemble method and we selected C4.5 as a base classifier to train a dataset. Recall from chapter 2, Bagging creates multiple C4.5 classifiers and uses a voting mechanism to select the most accurate one as shows in Figure 20. The main issue in Bagging is when a user selects a poor base classifier to train a dataset. Even if Bagging created multiple C4.5 classifiers and selected the best classifier, the selection of the base classifier still is not the right decision. Also, the base classifier might loss its efficiency as concept drift occurs.

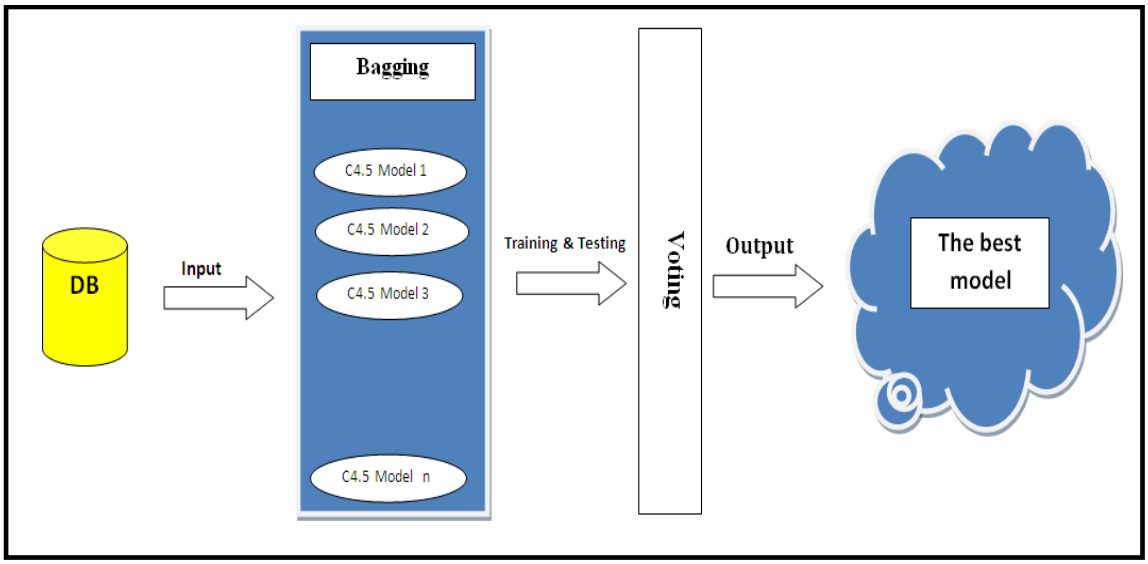

Figure 20: Example of Bagging C4.5

On the other hand, DCCD uses diverse base learners to increase the competition between the base classifiers to provide the highest possible accuracy. Moreover, using diverse base learners can limit the inappropriate selection of base classifiers that might be done by human as shows in Figure 21. Furthermore, DCCD has the ability to replace the base classifier that has the worst performance with a new member as presented in Figure 22 .

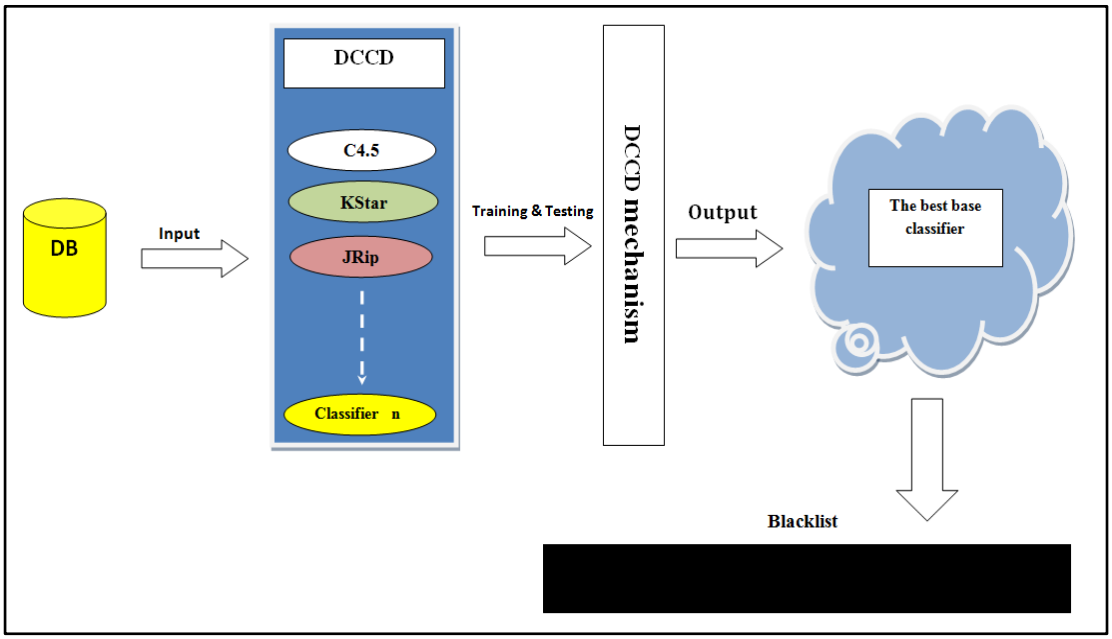

Figure 21: Example of DCCD

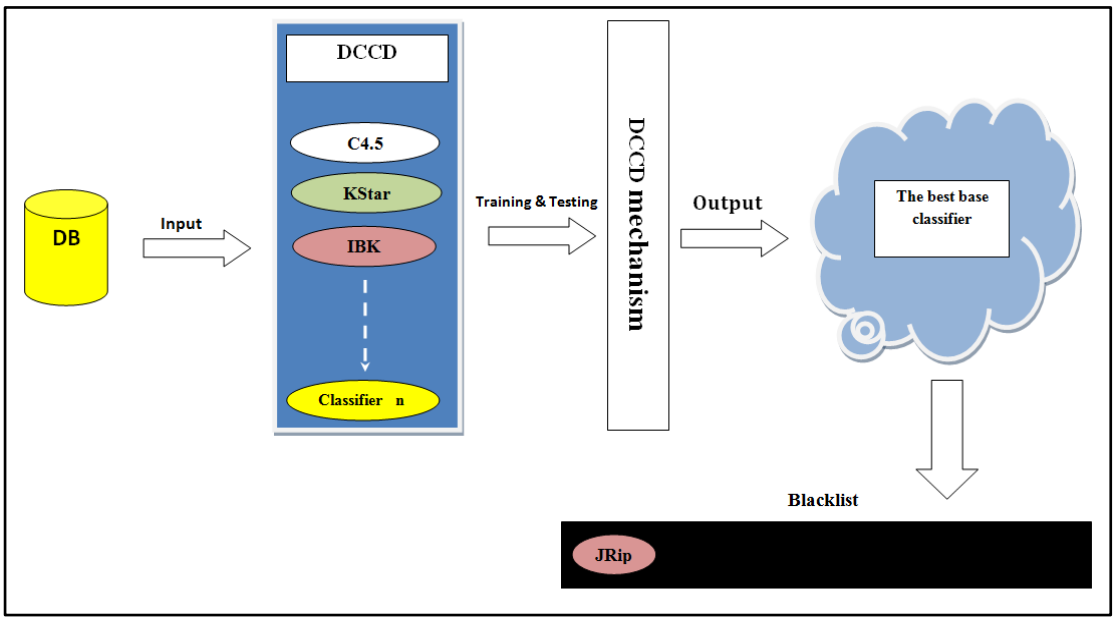

Figure 22: Example of DCCD after replacing a member

DCCD constructs a committee and then adds diverse base classifiers in the committee as members, as Figure 23 shows. DCCD uses the accuracy as the main measure to detect concept drifts. DCCD train the initial dataset and records the members' performance. When a new data arrives, DCCD trains the new incoming data and records the accuracy of each member in addition to updating the members' weight. Any negative change in the members' accuracy is considered as a concept drift.

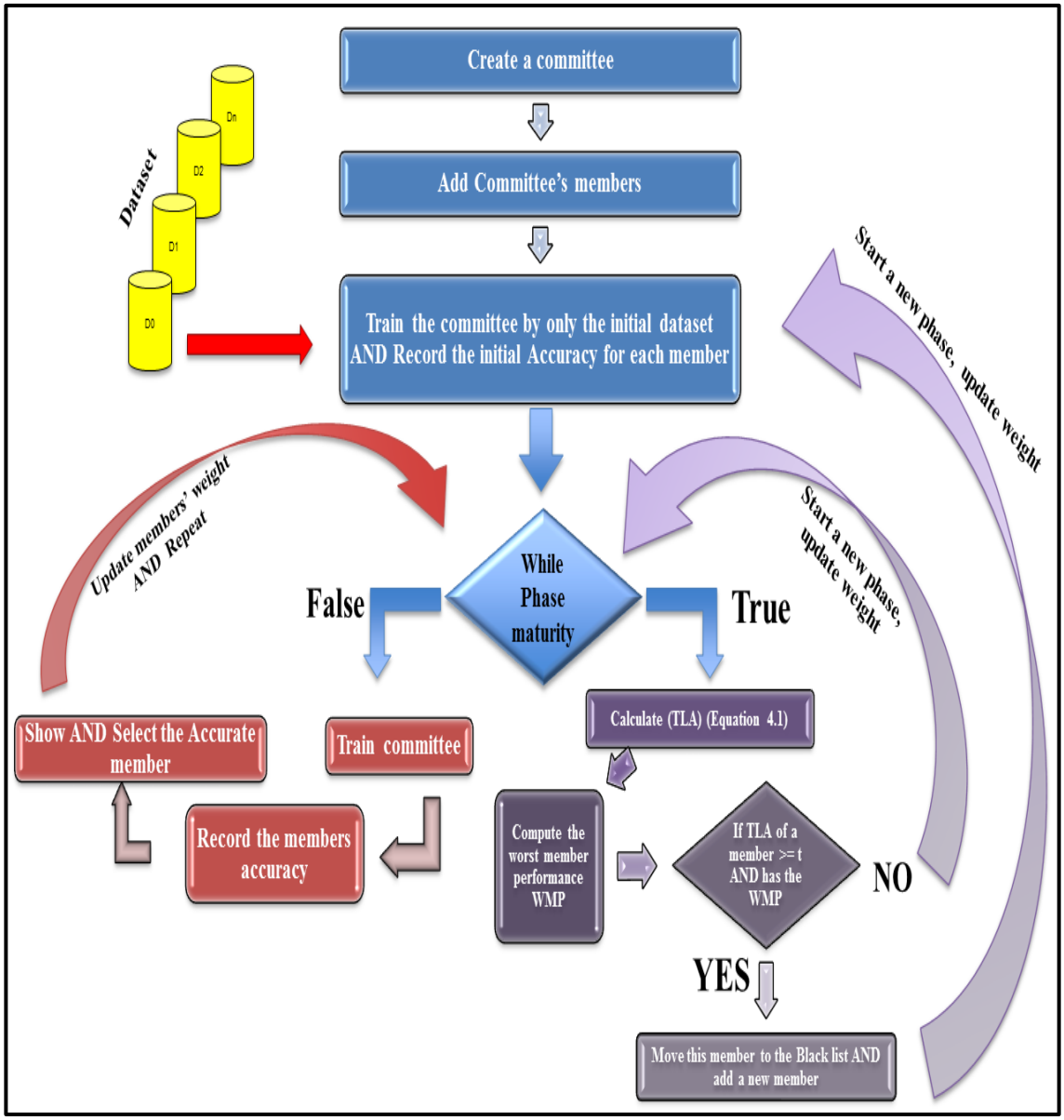

Figure 23: The DCCD Architecture

DCCD continues training the new incoming data until the phase maturity is achieved. A phase becomes mature when the number of drifts in the current phase is achieved. In order to save time and memory consumption, DCCD divides the number of drift scenarios into a number of phases. The main advantage of using phases is providing high scalability by evaluating each phase instead of evaluating each drift scenario. Furthermore, the number of drifts in each phase should not be too huge which might be late to handle concept drifts, or too small which can be too sensitive. This is hardest task which is setting the number of concept drifts in a phase.

When the number of drifts in a phase is low, the concept drift detection method will be stronger. This may increase the accuracy of DCCD, because DCCD will detect the worst member performance quickly. However, lowering the number of drifts in each phase may decrease the scalability of DCCD, because DCCD will have more evaluation points to perform. Therefore, the user must balance between accuracy and scalability to reach an acceptable compromise between them.

When a member's (i.e., a classifier's) performance drops below a minimal threshold *t* and it has the worst member performance at the end of a phase, this member is forced to retire. A new member then fills the open place on the committee. Moreover, DCCD detects the worst member in the committee to eliminate this member by using a weighting mechanism. DCCD weights the committee members based on their performance and detected the worst member performance based on their weighting (***will be discussed in sections 4.3.1 and 4.3.2***). DCCD has one main formula which is the Total Loss in Accuracy (TLA) to calculate the total lost accuracy at the end of each phase. The total loss in accuracy is calculated by equation (4.1):

$$\boldsymbol{TLA_i = \frac{\sum A_i}{D_i} - X_i} \tag{4.1}$$

Here, TLA_i refers to the total loss in accuracy for a committee $member_i$ and A_i refers to drifts accuracy in the current phase for a member. Moreover, D_i refers to the number of drifts in the current phase for the member while X_i stands for the initial accuracy for the member. DCCD calculate the total loss in accuracy by summing up the members' accuracy in the current phase and divide them by the number of drifts in the current phase. This gives us an indication of the rate of loss of the committee. The outcome of the previous division is subtracted from the initial accuracy.

For example, suppose we decided to have three drifts in each phase to make a good balance between accuracy and scalability. Moreover, the total lost accuracy must be less than 5% at the end of each phase. If the total lost accuracy for a member becomes more than 5%, DCCD identifies this reduction as a sign of concept drift which must be handled. Suppose our committee C_i that consists of three members as in Table 4.

Drift Scenarios		DCCD		
		BayesNet	**C4.5**	**JRip**
Drift 0 (Original)		85.71%	92.36%	86.69%
Phase 1	Drift 1	84.78%	89.93%	84.09%
	Drift 2	84.84%	87.15%	82.63%
	Drift 3	83.68%	90.39%	85.59%
Total Loss in accuracy TLA - Evaluation Point		-1.28%	-3.20%	-2.59%

Table 4: example shows how DCCD calculate the total loss in accuracy TLA

Now, we train the first coming data, the original dataset, and record the initial accuracy for each committee member. When new data arrives, DCCD starts to record the members' accuracy. At the end of the current phase, DCCD sums the accuracy of Drift 1, 2, and 3. Then, DCCD divides the previous outcome by the number of drifts which is in this case three. After that, DCCD subtracts outcome from the initial accuracy. Suppose we want to apply this to calculate the total loss in accuracy for JRip in Table 4.

$$TLA_{JRip} = \frac{84.69 + 82.63 + 85.59}{3} - 86.69$$

$$TLA_{JRip} = -2.59\%$$

This means that the total loss in accuracy for JRip TLA_{JRip} is (-2.59%) at the end of phase 1. This loss in accuracy is less than 5%, which is the acceptable range for losing accuracy. Therefore, DCCD will not force JRip to retire at this phase and JRip will continue to the next phase. Our algorithm works as shown in Figure 24.

Dynamic Committee for Concept Drift Algorithm (DCCD)

Input:

- D, Training set;
- NDP, the number of drifts in each phase
- C, Committee;
- C_i A committee member (learning scheme such as IBK, C4.5, JRip, etc.)
- IA, Initial Accuracy for each member;
- MA, Members' accuracy
- PM, Phase maturity Default value = false;
- AT, Accuracy threshold (The acceptable range for losing accuracy);
- MW_i Members Weighting Default value = '0';
- LWR The lowest weight recorded Default value = '0';
- BL, Fixed size blacklist;

Output: The most accurate member C_i

Method:

(1) *Create Committee C;*
(2) *Add the committee members C_i to Committee C (minimum = 2);*
(3) *Setup Accuracy threshold (AT);*
(4) *Train committee C by the Training set D;*
(5) *Record Initial Accuracy (IA) for each member;*
(6) *Update the weight of the most accurate member by adding 1;*
(7) *Show the result of the most accurate member;*
(8) *Start a phase;*

(9) ***While** Phase maturity PM== "false" **do***
{
(10) *Train committee C by the new incoming training set D_{new} ;*
(11) *Record Members' accuracy (MA);*
(12) *Update the weight of the most accurate member by adding 1;*
(13) *Show the result of the most accurate member;*
(14) *Update the lowest weight recorded (LWR);*
(15) ***End while***
}

(16) *Calculate (Total Loss in Accuracy (TLA) for each member) equation (4.1):*

$$TLA_i = \frac{\sum A_i}{D_i} - X_i;$$

(17) ***IF** (Total loss in Accuracy for a committee member C_i >= Accuracy threshold (AT) **AND** the weight of member C_i = the lowest weight recorded LWR)*
Then
(18) *Move the committee member C_i to the blacklist BL ;*
(19) *Add a new committee member $C_{i\,new}$ To committee C;*
(20) ***IF** the new committee member $C_{i\,new}$ in the blacklist BL **then,***
(21) *Add a new committee member $C_{i\,new}$;*
End if
(22) *Update the lowest weight recorded (LWR);*
End if
(23) ***Go to step (8);***

Figure 24: Dynamic Committee for Concept Drift (DCCD) Algorithm

DCCD begins by creating a committee *C* which consists of a minimum of two members. DCCD starts adding the committee's members C_i into C. After constructing the committee, DCCD trains the committee C by the first incoming data D and records the Initial Accuracy for each member AI in addition to show the result of the most accurate member. Moreover, DCCD updates the weight of the most accurate member by using the member's weighting mechanism (***as will be discussed in section 4.3.1***). When new data D_{new} arrives; DCCD trains C and records the members' accuracy because DCCD uses the accuracy to discover concept drifts. If DCCD discovers a change in the members' accuracy, DCCD considers this change as a concept drift. At each drift, DCCD updates the member's weighting, for each member by using the member's weighting mechanism because DCCD uses members' weighting mechanism to identify the worst member performance. The worst member performance is the member that has weight equal to lowest weight recorded (LWR) (***as will be discussed in section 4.3.2)***.

DCCD continues to record MA, the members' accuracy, until the value of PM, the phase maturity, becomes true. Once the value of PM becomes true, DCCD stops for an evaluation point. In the evaluation point, DCCD calculate the total loss in accuracy TLA by using equation (4.1). If a member lost more than the Accuracy threshold (AT) at end of a phase and has the worst member performance, DCCD moves this member to the blacklist BL. Finally, DCCD adds a new member ($C_{i\ new}$) to the committee *C*. Then, DCCD updates (LWR) and starts a new phase.

4.3.1 Overview of the weighting mechanism

The weighting mechanism calculates how many times a member has been selected to be the most accurate member in the entire committee's life cycle. In each drift, the most accurate member is awarded with a score of 1. The other members receive a score of 0. The weighting mechanism is used to detect the member with the worst performance.

For example, suppose we decided to have three drifts in each phase to make a good balance between accuracy and scalability. Moreover, the total lost accuracy must be less than 5% at the end of each phase. If the total lost accuracy for a member becomes more than 5%, DCCD identifies this reduction as a sign of concept drift which must be handled. Suppose our committee C_i that consists of three members as in Table 5.

Drift Scenarios		DCCD		
		BayesNet	C4.5	JRip
Drift 0 (Original)		85.71%	92.36%	86.69%
Phase 1	Drift 1	84.78%	89.93%	84.09%
	Drift 2	84.84%	87.15%	82.63%
	Drift 3	83.68%	88.39%	85.59%
Total Loss in accuracy - Evaluation Point		-1.28%	-5.20%	-2.59%

Table 5: Example for the weighting mechanism

According to Table 5, the most accurate classifier for drift 0 is C4.5, that means the weighting of C4.5 becomes 1. Also, C4.5 is the most accurate member in the committee in drift 1. Therefore, the weighting of C4.5 increases to 2. In drift 2 and drift 3, C4.5 continues to be the most accurate member in the committee, which makes the weighting of C4.5 increase to 4. For BayesNet and JRip, the weighting are 0 while the weighting for C4.5 is four because it is the most accurate member in all the drifts as presented in Table 6.

	BayesNet	C4.5	JRip
Member's Weight	0	4	0
ATL	-1.28%	-5.20%	-2.59%

Table 6: The members' weight at the end of phase 1 for the example for the weighting mechanism

4.3.2 Overview of the worst member performance detection mechanism

There are several conditions in our DCCD algorithm to determine when a committee member has to retire. One of these conditions is being the worst member, in terms of performance, in the committee. The worst member performance is determined by comparing the members' weight at the end of each phase to the lowest recorded weight. The member that has a weight equal to the lowest recorded weight in the committee is considered as the worst member performance ***WP***. Identifying the worst members in the committee can be done by the following example.

For example, suppose we setup the accuracy threshold, t, to be -5% and we decide that each phase consists of three drifts. Also, suppose our committee C consists of three members. These members are selected as follows:

- Member 1: BayesNet (BN)
- Member 2: K-Nearest Neighbour (IBK)
- Member 3: Decision Tree (C4.5)

Now, we train the classifiers with incoming dataset, the original dataset, and record the accuracy for each committee member. Suppose the results of running this committee against a dataset as presented in Table 7(a).

Drift Scenarios		DCCD		
		BN	IBK	C4.5
Drift 0		97.05%	90.33%	**98.06%**
Phase 1	Drift 1	**98.91%**	97.45%	97.45%
	Drift 2	86.00%	85.60%	**90.52%**
	Drift 3	94.69%	94.69%	**95.36%**
TLA		2.74%	-5.48%	-2.61%

a) **Results of running a committee against a dataset**

	BN	IBK	C4.5
Member's Weight	1	0	3
lowest recorded weight	0		

b) **The members' weight and the lowest recorded weight in the committee**

Table 7: Example to detect the worst member performance

Based on Table 7, the member that has its results in bold is the most accurate member in the committee. In each drift, the most accurate member is awarded with a score of 1 that will be added to its weight. The results from Table (a) show the weighting of C4.5 is 3 while the weighting of BayesNet is 0 and IBK is 1. Therefore, the lowest recorded weight in the committee is 0 which is IBK's weight. Therefore, DCCD considers IBK as the worst member performance in the committee. However, IBK lost more than 5% at the end of phase 1 and the IBK is the worst member performance in the committee. Therefore, DCCD forces IBK to retire because it lost more than t in its accuracy and its weighting is considered as the worst member performance.

4.4 Discussion

In this section, we present a discussion of our two algorithms. In this chapter, we introduced a new algorithm for introducing concept drift into databases called Concept Drift Introducer (CDI) to prepare experiments to test concept drift systems. Based on real-world databases, CDI create partially artificial data sets that affected by concept drift scenarios. CDI generates gradual and sudden concept drift scenarios randomly and then, introduces them into the selected dataset. In order to ensure that the sequence of the drift scenarios is accurate, CDI divides the number of drift scenarios into a number of phases. The main advantage of using

phases is providing high scalability to our main DCCD algorithm by evaluating each phase instead of evaluating each drift scenario.

Furthermore, we presented the DCCD algorithm that creates a committee that consists of two or more diverse base classifiers to handle gradual and sudden concept drift in databases. Recall that concept drifts cause a reduction in the current classifier accuracy that is used in examining the underlying data distribution. Therefore, DCCD uses the accuracy as the main measure to detect concept drifts. When the current classifier starts to lose its accuracy when concept drifts appear, DCCD considers this reduction in the accuracy as a sign of concept drift. Therefore, DCCD handles concept drift immediately when the loss in accuracies reaches unacceptable ranges.

Regular ensemble systems select one base classifier which subsequently creates **several** classifiers. This assumption ignores the idea of selecting a poor base classifier might affect the results. This is because the selected base classifier might be effective at a certain point of time. However, when concept drifts occur, the base classifier may have lost its efficiency. DCCD overcomes this issue by providing **diversity** through using base classifiers in one committee to ensure that DCCD provides highly accurate results in non-stationary environments. Through using the **diversity** of classifiers, we manage to get better results. In contrast, **severity** could provide misleading results, if the quality of the created classifiers is low. This is especially true in the ensembles that use majority voting.

The dynamically that DCCD has may limit the accuracy reduction that caused by concept drift. Recall that DCCD uses a weighting mechanism to detect the worst member performance. The main advantage of this mechanism is replacing the worst member performance with a new member that might provide better results. When a committee member has the worst performance in the committee and its accuracy drops below a minimal threshold, DCCD replace this member by a new member. The dynamically that DCCD has, increase the competition between the committee members, which will increase the overall performance of the system. This can be achieved by the dynamic ability in DCCD which allows adding and removing base classifiers, based on their performance. Furthermore, this dynamically can reduce the human errors when selecting inappropriate base classifiers to train a dataset. Further, DCCD provides flexibility for the user to use new base classifiers in the future. Moreover, the selected base classifiers might be effective at a certain time but when concept drifts occur, the selected base classifiers may have lost their efficiency.

In order to save time and memory consumption, DCCD divides the number of drift scenarios into a number of phases. The main advantage of using phases is providing high scalability by evaluating each phase instead of evaluating each drift scenario. DCCD provides flexibility to the users to reach an acceptable balance between accuracy and scalability based on their needs. In other words, the number of drifts in each phase should not be too huge which might be late to handle concept drifts, or too small which can be too sensitive. When the number of drifts in a phase is huge, the scalability of DCCD will be high but the accuracy of DCCD might decrease because it might be late to handle concept drifts. On the other hand, when the number of drifts in a phase is low, the concept drift detection method will be stronger which can increase the accuracy of DCCD because DCCD will detect the worst member performance quickly. However, lowering the number of drifts in a phase may decrease the scalability of DCCD because DCCD will have more evaluation points to perform.

The number of base classifiers in a committee should be determined based on finding a good trade-off between accuracy and scalability; this depends on users' judgment and the characteristics of the datasets used. For instance, credit cards approval system in most banks requires the highest possible accuracy that can be achieved regardless of the system's speed. On the other hand, stocks prediction system in a bank requires the highest possible scalability that can achieve an acceptable level of accuracy for the bank. Therefore, users must balance between accuracy and scalability to reach an acceptable compromise between them, based on the nature of the required tasks.

Furthermore, DCCD can use as many base classifiers as possible; however, we must consider a good trade-off between the size of the committee and (accuracy and accommodating an acceptable scalability). Islam et al. [92] shows that the number of members in an ensemble classifier strongly affects the ensemble classifier performance. Generally, when the number of members is large, the computational costs would increase while small number of members would lead to poor generalisation as can be seen in the worst case of only one member [92].

4.5 Summary

In this chapter, we presented a discussion about our methodology and our two algorithms. We introduced the Concept Drift Introducer algorithm (CDI) that introduces gradual and sudden concept drift scenarios into dataset. CDI selects the dataset that we would like to introduce concept drift scenarios into it and then, CDI divides the number of gradual and sudden drift scenarios into a number of phases.

Furthermore, we presented the DCCD algorithm that creates a dynamic committee that consists of two or more diverse base classifiers to handle gradual and sudden concept drift. DCCD uses the accuracy as the main measure to detect concept drift. When the current model starts to lose its accuracy when new data arrive, DCCD considers this reduction in accuracy as a sign of concept drift. DCCD uses a weighting mechanism to detect the worst member, in terms of performance. When a committee member has the lowest weight in the committee and its accuracy drops below a minimal threshold from its initial accuracy, DCCD forces the member to retire.The dynamically that DCCD has can limit the accuracy reduction that caused by concept drift. Furthermore, this dynamically may reduce the human errors in case of selecting inappropriate base classifiers for a dataset and provides flexibility for the user to use new base classifiers in the future.

To summarized, this chapter introduced the two algorithms that we have developed. The two algorithms are Concept Drift Introducer algorithm CDI and Dynamic Committee for Concept Drift algorithm (DCCD). In chapter 5, we present our experimental design and setup in addition to introducing the prepared datasets that will be used in this study.

CHAPTER FIVE

EXPERIMENTAL DESIGN

CHAPTER FIVE

EXPERIMENTAL DESIGN

This chapter highlights our experimental design and setup. To evaluate the performance of DCCD, we apply a ten-fold cross validation method. In this method, data are first divided into ten equal folds randomly. One fold is used as the test set and the rest of the nine sets are used as training sets. The error rate on each fold is averaged to yield an overall error rate [73]. Throughout our experiments, we have considered the accuracy as the main measure. The accuracy refers to the number of correctly classified examples divided by the total number of examples Accuracy is represented by the following formula:

$$Accuracy = \frac{\mathbf{TP+TN}}{\mathbf{TP+TN+FP+FN}} \quad (5.1)$$

Having a good understanding of the dataset in any concept drift experiment is an essential aspect. The reason for this is that when we want to apply any change to a dataset, this change must be consistent with the dataset as this change might affect the final results. In the interest of measuring the performance of our DCCD algorithm, concept drift scenarios have been introduced on several datasets by using CDI to determine several committees' reactions to these scenarios. DCCD divides the concept drift scenarios into several phases. Each phase consists of both gradual and sudden scenario. Table 8 provides the description of concept drift sceneries.

Procedure name	The Influence Percentage	Means
Original	**0%**	**The original dataset without any change**
Drift (Gradual)	**10%**	**Performing gradual drift scenario by changing 10% from an attribute distribution**
Drift (Sudden)	**50%**	**Performing gradual drift scenario by changing 50% from an attribute distribution**

Table 8: The description of concept drift sceneries

Finally, we perform a comparative comparison between DCCD and another algorithm called MultiScheme (MS) in order to evaluate the effectives of DCCD in handling concept drift compared to MS in Chapter 6.

5.1 The Selection of Datasets

Recall that DCCD algorithm is designed to work on batch datasets; hence, four datasets were selected to introduce concept drift scenarios into them by CDI. Our selection for those datasets is based mainly on the size to examine how DCCD performs against large, medium, small, and very small datasets. The selected datasets are taken from the UCI data repository, which were selected carefully to represent different scenarios [84]. Moreover, the dataset characteristics could make one classification algorithm more appropriate than another. However, the dataset characteristics do not affect the performance of DCCD directly but it may affect the performance of the base classifiers that DCCD uses ***as will be discussed in section 8.2***. Table 9 summarizes the main aspects of our datasets.

Dataset	Default Task	Attribute Types	Number of Attributes	Number of Instances	Number of Classes	Dataset Characteristics
Car Evaluation	Classification	Categorical	6	1728	4	Multivariate
Nursery	Classification	Categorical	8	12960	5	Multivariate
Contact lenses	Classification	Categorical	4	24	3	Multivariate
Iris Plants	Classification	Real	4	50	3	Multivariate

Table 9: Summary of our Datasets

5.2 Introducing Concept Drifts to the First Dataset: Car Evaluation Dataset

The first dataset is the Car Evaluation which was derived from a simple hierarchical decision model. The Number of Instances in the Car Evaluation database is 1728 and the number of input attributes is six attributes, namely buying, maintenance, doors, persons, the size of luggage boot, and safety. The attributes are described in Table 10.

Attribute name	Attribute means	Attribute name	Attribute means
buying	buying price	safety	estimated safety of the car
maint	price of the maintenance	persons	capacity in terms of persons to carry
doors	number of doors	lug_boot	the size of luggage boot

Table 10: Input Attribute Description

The attribute values for the buying and maintenance are v-high, high, med, or low. While the attribute values for the doors and persons are 1, 2, 3, 4, 5, or more, the size of luggage boot attribute values are big, med, or small and for the safety attributes high, med, or low. The target class value consists of four classes namely Unacceptable, Acceptable, Good, and Very good.

Drift Scenario	Drift point	The influence percentage
Drift 0 (original)	0	0%
Drift (Gradual)	172	10%
Drift (Sudden)	864	50%

Table 11: Drift Scenarios for Car Evaluation dataset

In Table 11, the drift scenario column corresponds to the drift type while the drift point column refers to the number of instances that have been changed. The influence percentage column refers to the percentage of change that influenced the entire dataset. However, all the drift scenarios were applied to the buying and the maintenance attributes. Drift 0 is the original dataset without any change. The gradual drift means that we applied changes to the original dataset by altering some of the maintenance attribute values form v-high to high. The total number of instances that we changed in gradual scenarios is 172 instances each time. This change influenced 10% of the entire dataset. In case of applying another gradual drift; we make additional changes to the dataset by another 10%.

The other type of drift scenario that has been applied is the sudden drift. We have applied many sudden scenarios to this dataset. Sudden scenario that has been applied to the dataset means altering 864 instances values from the maintenance attribute in the original

dataset each time. This type of drift is considered as a sudden and radical change, which influences 50% of the entire dataset. We divided this dataset into five phases; each phase consists of three drift scenarios. Moreover, each phase must consist of gradual and sudden concept drift.

5.3 Introducing Concept Drifts to the Second Dataset: Nursery Dataset

The Nursery dataset has been used many times in the past two decades, especially when there was excessive enrollment into nursery schools in Ljubljana, Slovenia. Many applications have been rejected and these rejections need an explanation to the authorities. The Nursery Database was derived from a hierarchical decision model originally developed to rank applications for nursery schools. The final decision depended on three sub problems: occupation of parents and child's nursery, family structure and financial standing, and social, and health picture of the family, as shown in Table 12. The number of instances in the Nursery database is 12960 instances and it has eight input attributes. The target class values consist of five classes, namely recommended, not recommended, very recommended, priority, and special prior.

Attribute Name	Input value
Parents	usual, pretentious, great_pret
has_nurs	proper, less proper, improper, critical, very crit.
form	complete, completed, incomplete, foster
children	1, 2, 3, more
housing	convenient, less conv, critical
finance	convenient, inconv
social	non-prob., slightly prob., problematic
health	recommended, priority, not recom.

Table 12: The input attributes of the Nursery Database

The drift scenarios that are introduced to the Nursery dataset are similar to the scenarios introduced to the Car Evaluation Dataset. All the drift scenarios are applied to the hours of nursing attribute. In Table 13, drift 0 is the original dataset without any change. On the gradual drifts, we applied drifts scenarios by modifying 10% of the hours of nursing attribute values at each time. When we apply another gradual scenario it means that we change 10% more in addition to the previous gradual drift. For example, suppose we applied the first gradual drift which is drift 1; then, when we want to apply another gradual drift. We apply additional changes to drift1 by 10%. This means that total changes of the first gradual drift and the second one is 20%.

Drift Scenario	Drift point	The influence percentage
Drift 0 (Original)	0	0%
Drift (Gradual)	1296	10%
Drift (Sudden)	6480	50%

Table 13 : Drift Scenarios for Nursery dataset

The other scenario drift that we introduced is sudden drifts. We altered 6480 instances from the original dataset. This type of drift is considered as a sudden and radical change, which influences 50% of the entire dataset. We also divided the drift scenarios into three phases; each phase consists of three drift scenarios. Moreover, each phase must consist of gradual and sudden concept drift.

5.4 Introducing Concept Drifts to the Third Dataset: Contact Lenses Dataset

On August 1st, 1990, Benoit Julien donated the contact lenses dataset, which has been used several times since that date. The number of instances in this database is 24 instances and the number of input attributes is four attributes.

Attribute Name	Input value
Age	young, pre-presbyopic, presbyopic
spectacle prescription	myope, hypermetrope
Astigmatic	no, yes
tear production rate	reduced, normal

Table 14: The input attributes of the contact lenses dataset

The examples in this database are complete and noise free and concern the problem of specifying which contact lenses are appropriate for each customer. In Table 14, we present the attributes names and the input values for these attributes, while in Table 15; we display the target class values which consist of three different concepts.

Class Value	Means
1	The patient should be fitted with hard contact lenses
2	The patient should be fitted with soft contact lenses
3	The patient should not be fitted with contact lenses

Table 15: The target class values of the contact-Lenses dataset

Similar scenarios have been introduced to the Contact lenses dataset are to the scenarios that we have introduced to the Car Evaluation and Nursery datasets. We decided to introduce twelve drift scenarios. Furthermore, we divided the drift scenarios into four phases; each phase consists of three drift scenarios. Each phase must include both gradual and sudden scenario. In Table 16, the drift scenario column refers to the drift type, while the drift point column refers to the number of instances that we have changed. The influence percentage corresponds to the influence percentage of the change to the entire dataset.

Drift Scenario	Drift point	The influence percentage
Drift 0 (Original)	0	0%
Drift (Gradual)	2	10%
Drift (Sudden)	12	50%

Table 16: Drift Scenarios for Contact-Lenses dataset

Table 16 shows the drift scenarios that introduced to the Contact-Lenses dataset. Drift 0 refers to the original dataset and the gradual drift means applying gradual drift on the original dataset by altering 10% from original dataset. All the modifications have been done on Age attribute; the total number of instances that we changed in this case is two instances for each gradual drift. This change influenced 10% of the entire dataset. The sudden drifts means that the numbers of instances that have been changed in the age attribute are twelve instances. Therefore, this change affects 50% of the entire dataset at each sudden drift scenario.

5.5 Introducing Concept Drift to the Fourth dataset: Iris Plants Dataset

This dataset created by R.A. Fisher in July, 1988 is one of the best known database to be found in the pattern recognition and data mining literature [**74**]. This dataset has no missing values. Moreover, there are 150 Instances, four input attributes, and three target classes in this dataset. The data set contains 3 classes of 50 instances each. Each class refers to a type of iris plant. Table 17 shows the input attributes and the description of them.

Attribute Name	Input value
sepal length	sepal length in cm
sepal width	sepal width in cm
petal length	petal length in cm
petal width	petal width in cm

Table 17: The input attributes of the Iris Plants dataset

In Table 18, we describe the target class values, which consist of the three different values namely; Setosa, Versicolour, and Virginica. Those values represent the types of Iris Plant that the classifiers should assign each instance to one of these values.

Class Value	Means
Setosa	The type of Iris is Setosa
Versicolour	The type of Iris is Versicolour
Virginica	The type of Iris is Virginica

Table 18: The target class values of the Iris Plant dataset

Similar drift scenarios have been introduced to the Iris Plant dataset. Twelve drift scenarios have been applied to this dataset divided into four phases. Also, each phase must include both gradual and sudden scenario. In Table 19, the drift scenario column refers to the drift type, while the drift point column refers to the number of instances that we have changed. The influence percentage corresponds to the influence percentage of the change to the entire dataset. However, all the drift scenarios are applied to all attributes.

Drift Scenario	Drift point	The influence percentage
Drift 0 (Original)	0	0%
Drift (Gradual)	15	10%
Drift (Sudden)	75	50%

Table 19: Drift Scenarios for Iris Plant dataset

The gradual drift means applying gradual drift on the original dataset by modifying 10% from the original dataset. The total number of instances that we changed in this case is 15 instances for each gradual drift. This change influenced 10% of the entire dataset. The number of gradual drift that have been applied to this dataset is six drifts. The sudden drift means that the numbers of instances that have been changed are 75 instances. Therefore, this change affects 50% of the entire dataset at each sudden drift scenario. However, there are six sudden drifts that have been applied to this dataset.

5.6 Discussion

This chapter highlighted how we introduced concept drift scenarios into four datasets in order to use them in our experiment in Chapter 6. We introduced concept drift scenarios by using an effective algorithm called Concept Drift Introducer (CDI). Throughout our experiment, we took

advantage of a number of pre-existing datasets for data mining usage [**74**]. The main goal of introducing concept drift scenarios to datasets is to perform DCCD algorithm on these dataset in the next chapter.

CDI splits the drift scenarios into a number of phases and then CDI selects the target attribute that the concept drift scenarios will appear on it. Moreover, CDI specifies the original value which needs to be changed by a new value in order to perform the drift scenarios. The second step of the CDI algorithm is specifying the number of gradual and sudden drift scenarios in the current phase. CDI requires the user to enter the number of gradual and sudden drift scenarios in each phase. At the end, CDI changes the old value to the new value by 10% for the gradual drift or by 50% for the sudden drifts.

5.7 Summary

In this chapter, we discussed introducing concept drift to four dataset by using the CDI algorithm in order to perform DCCD algorithm on these dataset in our experiments. For the experiments in this study, we introduced two types of concept drifts, gradual and sudden, into four datasets. We described these datasets and then introduced drift scenarios really carefully because having a good understanding of the dataset in any concept drift experiment might affect the final results. Moreover, all the introduced drift scenarios are consistent with the dataset. In order to examine our approach's reaction to gradual and sudden concept drifts, we divided the concept drift scenarios into several phases. Each phase consists of gradual and sudden scenarios.

In the next two chapters, we analyze and evaluate the performance of DCCD for each of the two types of concept drifts. In Chapter 6, we present the result analysis of four committees performed on four datasets. The results analysis shows how DCCD can limit the accuracy reduction that is caused by the concept drift when the data distribution changes over time. Chapter 7 provides a comparison between our algorithm (DCCD) and a similar approach, namely MultiScheme.

CHAPTER SIX
EXPERIMENTAL EVALUATION

CHAPTER SIX

EXPERIMENTAL EVALUATION

In this chapter, we provide the experimental evaluation of our DCCD algorithm. The main experimental goal is to examine the ability of the dynamic committee of diverse base classifiers (DCCD) to handle gradual and sudden concept drifts. Another goal is to measure the performance of the base classifiers in different committees against the same datasets affected by concept drifts. Moreover, studying base classifier loss of efficiency as concept drift occurs is the last goal of this experiment. We ran a variety of DCCD committees against several datasets affected by gradual and sudden concept drifts.

6.1 Experiment setup

In this experiment, we select the seven base learning classifiers, which were overviewed in Chapter 3 to be used as members in DCCD's committees. Those base learning classifiers are selected to provide diversity to the committee members and at the same time accurate enough to achieve a good generalisation. Additionally, we use three base classifiers in each DCCD committee, which in fact we can use as many as possible as discussed in section 4.4. However, three base classifiers in a committee provide a good trade-off between the size of the committee and (accuracy and accommodating an acceptable scalability).

In this experiment, we ran four committees of DCCD against the four datasets that we introduced concept drift scenarios into them to show the ability of DCCD to recover from selecting a poor base classifier. Therefore, four committees of DCCD that include different members can provide a good generalization. Throughout this experiment, the four committees of DCCD are trained by the same sequence of drift scenarios that were introduced into the selected datasets for the fairness and consistency of evaluating the DCCD algorithm.

By inspection, we decided to set the accuracy threshold t to -5%, by inspection. If the total lost accuracy for a member becomes more than -5%, DCCD identifies this reduction as a sign of concept drift which must be handled. Furthermore, recall that to evaluate the performance of DCCD, we apply a ten-fold cross validation method and we used the accuracy as the main measure. Also, we use an open-source Java-based machine learning called WEKA to perform this experiment.

6.2 The Results of the First Committee

In this section, we present our analysis of the results of our experiments. The first committee consists of three committee members which are BayesNet, C4.5, and JRip. These members are chosen based on specific criteria. We chose C4.5 to compare it with JRip in terms of accuracy and to measure their performance on different sizes of datasets. Moreover, we chose BayesNet to measure its performance in terms of handling concept drifts because it has an updater that enables BayesNet to learn from new data samples which might be useful for handling concept drifts. This committee was run against four datasets which was discussed in the previous chapter.

6.2.1 The results of the first dataset: Car Evaluation Dataset

As we mentioned in Chapter 5, there are fifteen drift scenarios applied to the car evaluation dataset. Seven of these drift scenarios are gradual and the rest are sudden. Table 20 represents the results of the first committee in details. In addition, it shows where DCCD replaced one of the committee members.

Drift Scenarios		DCCD		
		BayesNet	**C4.5**	**JRip**
Drift 0 (Original)		85.71%	92.36%	86.69%
Phase 1	**Drift 1 (Gradual)**	84.78%	89.93%	84.09%
	Drift 2 (Sudden)	84.84%	87.15%	82.63%
	Drift 3 (Gradual)	83.68%	90.39%	85.59%
Total Loss in accuracy - Evaluation Point		-1.28%	-3.20%	-2.59%
Phase 2	**Drift 4 (Sudden)**	82.70%	86.17%	83.22%
	Drift 5 (Gradual)	84.20%	90.39%	86.46%
	Drift 6 (Sudden)	81.37%	80.21%	76.45%
Total Loss in accuracy - Evaluation Point		-2.95%	-6.77%	-4.65%
Phase 3	**Drift 7 (Sudden)**	83.68%	84.66%	81.59%
	Drift 8 (Gradual)	83.56%	86.92%	81.48%
	Drift 9 (Sudden)	82.06%	83.04%	80.55%
Total Loss in accuracy - Evaluation Point		-2.61%	-7.49%	-5.48%
Replacing JRip by SMO		**BayesNet**	**C4.5**	**SMO**
Phase 4	**Drift 10 (Gradual)**	82.29%	86.23%	87.61%
	Drift 11 (Gradual)	84.66%	89.23%	89.40%
	Drift 12 (Sudden)	84.61%	83.91%	85.65%
Total Loss in accuracy - Evaluation Point		-1.86%	-5.90%	-0.06%
Phase 5	**Drift 13 (Sudden)**	86.05%	89.29%	87.21%
	Drift 14 (Gradual)	84.95%	88.48%	89.12%
	Drift 15 (Gradual)	82.58%	84.43%	86.28%
Total Loss in accuracy - Evaluation Point		-1.18%	-4.96%	0.85%

Table 20: Running the first committee against the car evaluation dataset

C4.5 is the most accurate member in entire phase 1. At the end of phase 1, an evaluation point takes place to evaluate the members' performance. BayesNet lost 1.28%, C4.5 lost 3.20%, and JRip lost 2.59% from their initial accuracy. Recall from Chapter 4, that there are two conditions to replace a member; namely losing more of the accuracy threshold (5% in this experiment) and being the worst member performance. The worst member performance is determined by the weighting mechanism. Based on the weighting mechanism, the most accurate member in each drift gets 1 point in its weight. Therefore, the weight of C4.5 is 4 points while BayesNet's and JRip weight is 0 point. Therefore, all members are successfully handled concept drifts in phase 1 and they will continue to the next phase.

In phase 2, the most accurate member in drift 5 and 6 is C4.5 while BayesNet is the most accurate member in drift 6. At the end of phase 2, BayesNet lost 2.95% and JRip lost

4.65% of their initial accuracy which is still in the acceptable range of the loss in accuracy. On the other hand, C4.5 lost 6.77% of its initial accuracy which is above the acceptable range. The weight of C4.5 is 6 points while BayesNet's weight is 1 point and JRip is 0. Hence, C4.5 lost more than 5% of its initial accuracy but it is not the worst member performance. Therefore, C4.5 will continue to the next phase.

C4.5 is the most accurate member in the entire phase 3. At the end of phase 3, BayesNet lost 2.61% of its initial accuracy which stills in the acceptable range of the loss in accuracy. On the other hand, C4.5 lost 7.49% and JRip lost 5.48% of their initial accuracy which are above the acceptable range. According to the weight mechanism, the weight of C4.5 is 9 points while BayesNet's weight is 1 and JRip is 0. Hence, C4.5 lost more than 5% of its initial accuracy but it is not the worst member performance while JRip lost more 5% of its initial accuracy and is the worst member performance. Therefore, C4.5 will continue to the next phase while JRip will be replaced with a new member which is SMO.

In phase 4, the most accurate member is SMO for the entire phase. At the end of phase 4, BayesNet lost 1.86% and SMO lost 0.06% from their initial accuracy while C4.5 lost 5.90% which is above the acceptable range. According to the weight mechanism, the weight of C4.5 is 9 points while BayesNet's weight is 1 point and SMO is 3. Hence, C4.5 lost more than 5% of its initial accuracy but it is not the worst member performance. Therefore, C4.5 will continue into phase 5. In drift 13, the most accurate member is C4.5 whereas in drift 14 and 15, SMO is the most accurate member. At the end of phase 5, BayesNet lost 1.18% and C4.5 lost 4.96% while SMO lost 0.85% of their initial accuracy which is above the acceptable range. According to the weight mechanism, the weight of C4.5 is 10 points while BayesNet's weight is 1 and SMO is 5.

6.2.2 The results of the second dataset: Nursery dataset

Table 21 shows the results of the first committee in details. Conjointly, it displays where DCCD replaced one of the committee members. There are nine drift scenarios have been introduced to the nursery dataset. Five of these drift scenarios are gradual and four scenarios are sudden.96.98%.

Drift Scenarios		DCCD		
		BayesNet	C4.5	JRip
Drift 0 (Original)		90.33%	97.05%	96.98%
Phase 1	Drift 1 (Gradual)	90.76%	97.45%	97.28%
	Drift 2 (Sudden)	85.99%	90.52%	85.72%
	Drift 3 (Gradual)	88.22%	95.36%	92.65%
Total Loss in accuracy - Evaluation Point		-2.01%	-2.61%	-5.10%
Replacing JRip by SMO		BayesNet	C4.5	SMO
Phase 2	Drift 4 (Sudden)	85.49%	91.75%	86.70%
	Drift 5 (Gradual)	87.05%	92.59%	89.78 %
	Drift 6 (Sudden)	83.86 %	91.17%	88.02%
Total Loss in accuracy - Evaluation Point		-4.86%	- 5.21%	+1.46%
Phase 3	Drift 7 (Gradual)	87.18%	94.79%	91.68%
	Drift 8 (Gradual)	90.22%	97.49%	93.84%
	Drift 9 (Sudden)	87.95%	94.88%	91.81%
Total Loss in accuracy - Evaluation Point		-1.88%	-1.33%	+ 5.74%

Table 21: Running the first committee against the nursery dataset

For the Nursery dataset, C4.5 is the most accurate member in the committee for all phases. At the end of phase 1, BayesNet lost 2.01% and C4.5 lost 2.61% from their initial accuracy while JRip lost 5.10% from its initial accuracy which is above the acceptable rage. Therefore, JRip will be replaced by a new member which is SMO. At the end of phase 2, BayesNet lost 4.86% whereas SMO's accuracy went up by 1.46%. On the other hand, C4.5 lost 5.21% from its initial accuracy which is above the acceptable range. However, C4.5 will continue to phase 3 because C4.5 is not the worst member performance. In phase 3, the accuracy of SMO increases by 5.74% while C4.5 and BayesNet lost 1.33% and 1.88% from their initial accuracy.

6.2.3 The results of the third dataset: Contact-Lenses dataset

Nine drift scenarios have been introduced to the contact-lenses dataset and are divided into four phases. Five of these scenarios are sudden drifts while four of them are gradual drifts. Table 22 shows the results of performing the first committee on the Contact Lenses dataset.

The results in the above table show that C4.5 is the most accurate member in the committee for all phases in the contact lenses dataset. The first evaluation point shows that the accuracy of BayesNet and C4.5 increased by 1.39% from their initial accuracy while JRip's accuracy increased by 2.78% .At the end of phase 2, the accuracy of BayesNet and JRip increase whereas the accuracy of C4.5 remains stable. In the last evaluation point, all members' accuracy has been increased. In this dataset, none of the members was replaced due to the stability and the small size of the dataset.

Drift Scenarios		DCCD		
		BayesNet	C4.5	JRip
Drift 0 (Original)		70.83%	83.33%	75.00%
Phase 1	Drift 1 (Gradual)	70.83%	83.33%	75.00%
	Drift 2 (Sudden)	75.00%	87.50%	79.17%
	Drift 3 (Gradual)	70.83%	83.33%	79.17%
Total Loss in accuracy - Evaluation Point		1.39%	1.39%	2.78%
Phase 2	Drift 4 (Sudden)	70.83%	83.33%	79.17%
	Drift 5 (Gradual)	70.83%	83.33%	79.17%
	Drift 6 (Sudden)	79.17%	83.33%	83.33%
Total Loss in accuracy - Evaluation Point		2.78%	0.00%	5.56%
Phase 3	Drift 7 (Sudden)	75.00%	87.50%	79.17%
	Drift 8 (Gradual)	70.83%	87.50%	87.50%
	Drift 9 (Sudden)	75.00%	87.50%	79.17%
Total Loss in accuracy - Evaluation Point		2.78%	4.17%	6.95%

Table 22: Running the first committee against the contact-lenses dataset

6.2.4 The results of the fourth dataset: Iris Plants dataset

The fourth dataset in our experiment is the Iris Plants dataset. Twelve drift scenarios were introduced into the Iris dataset as mentioned in Chapter 5. Half of these drift scenarios are gradual and half are sudden. Table 23 suggests the results of performing the first committee against Iris Plants dataset.

Drift Scenarios		DCCD		
		BayesNet	C4.5	JRip
Drift 0 (Original)		92.67%	96.00 %	94.00%
Phase 1	Drift 1 (Gradual)	95.33%	95.33%	94.00%
	Drift 2 (Sudden)	95.33%	95.33%	94.67%
	Drift 3 (Gradual)	95.33%	95.33%	94.00%
Total Loss in accuracy - Evaluation Point		2.66%	-0.67%	0.22%
Phase 2	Drift 4 (Sudden)	95.33%	95.33%	94.00%
	Drift 5 (Gradual)	95.33%	95.33%	94.67%
	Drift 6 (Sudden)	95.33%	95.33%	95.33%
Total Loss in accuracy - Evaluation Point		2.66%	-0.67%	0.67%
Phase 3	Drift 7 (Sudden)	94.67%	94.67%	96.00 %
	Drift 8 (Gradual)	94.00%	94.00%	94.67%
	Drift 9 (Sudden)	94.67%	94.67%	93.33%
Total Loss in accuracy - Evaluation Point		1.78%	-1.55%	0.67%
Phase 4	Drift 10 (Gradual)	94.00%	94.00%	94.67%
	Drift 11 (Gradual)	94.00%	94.00%	95.33%
	Drift 12 (Sudden)	94.00%	94.00%	94.00%
Total Loss in accuracy - Evaluation Point		1.33%	-2.00%	0.67%

Table 23: Running the first committee against the Iris dataset

In phase 1, C4.5 is the most accurate member. The first evaluation point shows that C4.5 is the only member that lost from its initial accuracy by 0.67% whereas the accuracy of

BayesNet and JRip increased. Therefore, all members are successfully handled concept drifts in phase 1 and they will continue to the next phase. At the end of phase 2 and phase 3, the accuracy of BayesNet and JRip increases while C4.5 lost 1.55%. In the last evaluation point, C4.5 lost 2.00% of its initial accuracy while the accuracy of BayesNet and JRip increased. Table 24 presents a summary of the performance of the first committee.

	Car Evaluation Dataset	Nursery dataset	Contact Lenses Dataset	Iris Plants Dataset
Drift 0	92.36%	97.05%	83.33%	96.00 %
Drift 1	89.93%	97.45 %	83.33%	95.33%
Drift 2	87.15%	90.52 %	87.50%	95.33%
Drift 3	90.39%	95.36%	83.33%	95.33%
Drift 4	86.17%	91.75%	83.33%	95.33%
Drift 5	90.39%	92.59%	83.33%	95.33%
Drift 6	81.37%	91.17%	83.33%	95.33%
Drift 7	84.66%	94.79%	87.50%	96.00 %
Drift 8	86.92%	97.49%	87.50%	94.67%
Drift 9	83.04%	94.88%	87.50%	94.67%
Drift 10	87.61%			95.33%
Drift 11	89.40%			95.33%
Drift 12	85.65%			94.00%
Drift 13	89.29%			
Drift 14	89.12%			
Drift 15	86.28%			

Table 24: Summary of results of the first committee

6.3 Results of the Second Committee

We present our results and the analysis of the second committee by running it on four datasets. The second committee consists of three committee members: BayesNet, IBK, and C4.5. We select IBK to measure its performance in terms of the accuracy on large datasets that are facing concept drift. Furthermore, we want to examine the distances measures that IBK generate in order to classify new instances. Moreover, we chose BayesNet to test its updater that allows BayesNet to learn from new data samples. During any drift scenario, by inspection, the acceptable range or accuracy threshold, in which the accuracy might decrease, is 5%.

6.3.1 The results of the first dataset: Car Evaluation dataset

Twelve drift scenarios have been introduced on the car evaluation dataset. Six of these drift scenarios are gradual and six are sudden. Table 25 reveals the results of the second committee in

details. Additionally, it points out when DCCD replaced one of the committee members after drift 6.

Drift Scenarios		DCCD		
		BayesNet	IBK	C4.5
Drift 0 (Original)		85.71 %	93.52%	92.36%
Phase 1	Drift 1 (Gradual)	84.78%	89.47%	89.93%
	Drift 2 (Sudden)	84.84 %	85.07%	87.15%
	Drift 3 (Gradual)	83.68%	89.35%	90.39%
Total Loss in accuracy - Evaluation Point		-1.28%	-4.36%	-3.20%
Phase 2	Drift 4 (Sudden)	82.70%	83.97%	86.17%
	Drift 5 (Gradual)	84.20%	90.17%	90.39%
	Drift 6 (Sudden)	81.36%	76.04%	80.21%
Total Loss in accuracy - Evaluation Point		-2.96%	-9.06%	-6.77%
Replacing IBK by PART		BayesNet	PART	C4.5
Phase 3	Drift 7 (Sudden)	83.68%	83.91 %	84.66%
	Drift 8 (Gradual)	83.57%	89.35%	86.92%
	Drift 9 (Sudden)	82.06%	84.66%	83.04%
Total Loss in accuracy - Evaluation Point		-2.61%	2.06%	-3.20%
Phase 4	Drift 10 (Gradual)	82.29%	88.37%	86.23%
	Drift 11 (Gradual)	84.66%	89.29 %	89.23%
	Drift 12 (Sudden)	84.61%	84.61%	83.91%
Total Loss in accuracy - Evaluation Point		-1.86%	3.51%	-3.20%

Table 25: performing the second committee against the Car Evaluation dataset

IBK is the most accurate member in drift 0 while C4.5 is the most accurate member in drifts 1, 2, and 3. At the end of phase 1, all members lost less than 5% which is in the acceptable range for the loss in accuracy. Therefore, all members handled concept drifts successfully in phase 1 and they will continue to the next phase. In phase 2, the most accurate member in drifts 5 and 6 is C4.5 while BayesNet is the most accurate member in drift 6. At the end of phase 2, BayesNet loss 2.96% whereas C4.5 loss 6.77 and IBK loss 9.06% of their initial accuracy which is above the acceptable range. According to the weight mechanism, the weight of C4.5 is 5 points while the weight of IBK and BayesNet is 1. Hence, C4.5 loss more than 5% of its initial accuracy but it is not the worst member performance. Therefore, IBK will be replaced with a new member which is PART while C4.5 will continue to the next phase.

C4.5 is the most accurate member in drift 6 while PART is the most accurate member in drifts 7, and 8. At the end of phase 3, BayesNet loss 2.61% and C4.5 loss 3.20% of their initial accuracy which stills in the acceptable range of lost in accuracy. On the other hand, the accuracy of PART increased by 2.06% from its initial accuracy. Hence, all members are successfully handled concept drifts in phase 3 and they will continue to the next phase. In phase 4, the most

accurate member is PART for the entire phase. At the end of phase 4, BayesNet loss 1.86% and C4.5 loss 3.20% of their initial accuracy while the accuracy of PART has increased by 3.51%.

6.3.2 The results of the second dataset: Nursery dataset

Table 26 provides the results of performing the second committee against the Nursery dataset. There are nine drift scenarios applied to the Nursery dataset; they are divided as five gradual drifts and four sudden drifts.

Drift Scenarios		DCCD		
		BayesNet	IBK	C4.5
Drift 0 (Original)		90.33%	98.06%	97.05%
Phase 1	Drift 1 (Gradual)	98.91%	97.45%	97.45%
	Drift 2 (Sudden)	86.00%	85.60%	90.52%
	Drift 3 (Gradual)	94.69%	94.69%	95.36%
Total Loss in accuracy - Evaluation Point		2.74%	-5.48%	-2.61%
Replacing IBK by PART		**BayesNet**	**PART**	**C4.5**
Phase 2	Drift 4 (Sudden)	85.49%	91.89%	91.78%
	Drift 5 (Gradual)	87.05%	93.98%	92.59%
	Drift 6 (Sudden)	83.86 %	91.54%	91.17%
Total Loss in accuracy - Evaluation Point		-4.86%	0.58%	-5.20%
Phase 3	Drift 7 (Gradual)	87.18%	96.17%	94.79%
	Drift 8 (Gradual)	90.23%	99.20%	97.49%
	Drift 9 (Sudden)	87.95%	96.20%	94.88%
Total Loss in accuracy - Evaluation Point		-1.88%	+5.30%	-1.33%

Table 26: Running the second committee against the Nursery dataset

The accuracy of drift 0, the original dataset, is 90.33% for BayesNet and 97.05% for C4.5, whereas the most accurate member is IBK at 98.06%. Phase 1 begins at drift 1; the most accurate member is BayesNet in this drift. The accuracy of BayesNet is 98.91%, which is an increase from its initial accuracy by almost 9%. C4.5 is the most accurate member in drift 2 and drift 3, whereas IBK and BayesNet lose some of their accuracy during drift 2 and drift 3. After drift 3, DCCD starts the first evaluation point, which shows an increase in the average accuracy for BayesNet, while the average accuracy for IBK and C4.5 decreases by 5.48% and 2.61%. Moreover, DCCD forces IBK to retire and replaces it with a new member, which is PART. IBK is forced to retire because it has lost more than 5% from its initial accuracy and it is the worst member in terms of performance. The new committee becomes BayesNet, PART, and C4.5.

Drift 4 is the first drift in phase two, which shows that the highest accuracy is 91.89% provided by the new member, PART. Also, PART is the most accurate member in drift 5 and

drift 6. PART shows an acceptable performance in phase 2 compared to C4.5 and BayesNet. C4.5 and BayesNet start to lose more of their accuracy during phase two. After drift 6, DCCD starts its second evaluation point, which shows an average accuracy loss for BayesNet and C4.5, while the average accuracy increases for PART of its initial accuracy. BayesNet loses 4.86% of its initial accuracy, whereas C4.5 loses 5.20%. DCCD would not force C4.5 to retire because it is not the worst member in terms of its performance. In addition, DCCD cannot force BayesNet to retire because its average accuracy loss is less than 5%, although BayesNet is the worst member in terms of performance. Therefore, all the committee members will continue to the next phase. The last phase in this dataset is phase three, which consists of drift 7, 8, and 9. The most accurate member in the entire phase is PART.

6.3.3 The results of the third dataset: Contact Lenses dataset

The contact-lenses dataset are divided into four phases which consist of twelve drift scenarios that have been introduced to it. Table 27 shows the results of performing the first committee on the Contact Lenses dataset.

Drift Scenarios		DCCD		
		BayesNet	IBK	C4.5
Drift 0 (Original)		70.83%	70.83%	83.33%
Phase 1	Drift 1 (Gradual)	70.83%	70.83%	83.33%
	Drift 2 (Sudden)	75.00%	66.67%	87.50%
	Drift 3 (Gradual)	70.83%	75.00 %	83.33%
Total Loss in accuracy - Evaluation Point		1.39%	0.00%	1.39%
Phase 2	Drift 4 (Sudden)	70.83%	75.00 %	83.33%
	Drift 5 (Gradual)	70.83%	70.83%	83.33%
	Drift 6 (Sudden)	79.17%	75.00 %	83.33%
Total Loss in accuracy - Evaluation Point		2.78%	2.78%	0.00%
Phase 3	Drift 7 (Gradual)	75.00%	66.67%	87.50%
	Drift 8 (Gradual)	70.83%	62.50 %	87.50%
	Drift 9 (Sudden)	75.00%	66.67%	87.50%
Total Loss in accuracy - Evaluation Point		2.78%	-5.55%	4.17%
Replacing IBK by KStar		**BayesNet**	**KStar**	**C4.5**
Phase 4	Drift 10 (Gradual)	70.83%	79.17%	83.33%
	Drift 11 (Gradual)	79.17%	75.00 %	83.33%
	Drift 12 (Sudden)	79.17%	87.50 %	83.33%
Total Loss in accuracy - Evaluation Point		5.56%	1.39%	0.00%

Table 27: Running the second committee against the Contact Lenses dataset

The most accurate member in phase 1, 2 and 3 is C4.5. At the end of phase 3, the accuracy of BayesNet and C4.5 have increased while IBK loss more than 5% which is the acceptable range for accuracy loss in addition to the lower weight that IBK has. Therefore, DCCD forces IBK to retire and be replaced with a new member which is KStar. In drifts 10 and 11, the most accurate member is C4.5 whereas KStar is the most accurate member in drift 12.

6.3.4 The results of the fourth dataset: Iris Plants dataset

Table 28 presents the results of performing the second committee against fourth dataset. The fourth dataset in our experiment is the Iris dataset. Twelve drift scenarios were introduced into it as mentioned in Chapter 5. Half of these drift scenarios are gradual and half are sudden.

Drift Scenarios		DCCD		
		BayesNet	IBK	C4.5
Drift 0 (Original)		92.67%	95.33%	96.00 %
Phase 1	Drift 1 (Gradual)	93.33%	95.33%	95.33%
	Drift 2 (Sudden)	93.33%	96.00 %	95.33%
	Drift 3 (Gradual)	93.33%	95.33%	95.33%
Total Loss in accuracy - Evaluation Point		0.66%	0.22%	-0.67%
Phase 2	Drift 4 (Sudden)	94.67%	95.33%	95.33%
	Drift 5 (Gradual)	94.00 %	96.00 %	95.33%
	Drift 6 (Sudden)	94.67%	95.33%	95.33%
Total Loss in accuracy - Evaluation Point		1.78%	0.22%	-0.67%
Phase 3	Drift 7 (Sudden)	92.67%	95.33%	94.67%
	Drift 8 (Gradual)	94.00 %	95.33%	94.00%
	Drift 9 (Sudden)	94.00 %	94.67%	94.67%
Total Loss in accuracy - Evaluation Point		0.89%	-0.22%	-1.55%
Phase 4	Drift 10 (Gradual)	89.33%	96.00 %	94.00%
	Drift 11 (Gradual)	92.67%	96.00 %	94.00%
	Drift 12 (Sudden)	92.67%	98.00 %	94.00%
Total Loss in accuracy - Evaluation Point		-1.11%	1.34%	-2.00%

Table 28: Running the second committee against the Iris Plants dataset

In Drift 0, 1, and 3, C4.5 is the most accurate member whereas IBK is the most accurate member in drift 2. The first evaluation point shows that C4.5 is the only member that has lost 0.67% of its initial accuracy whereas the accuracy of BayesNet and IBK has increased. Therefore, all members are handled concept drifts successfully in phase 1 and they will continue to the next phase. In phase 2, C4.5 is the most accurate member in drift 4 while IBK is the most accurate member in drift 5 and drift 6. The second evaluation point shows that C4.5 is the only

member that has lost from its initial accuracy by 0.67% whereas the accuracy of BayesNet and IBK has increased. Therefore, all members will continue to the next phase.

IBK is the most accurate member in drift 7 and drift 8 while C4.5 is the most accurate member in drift 9. The third evaluation point shows that all members will continue to the next phase because none of them lost more than 5% of its initial accuracy. In phase 4, IBK is the most accurate member in the entire phase. Table 29 presents a summary of the second committee's performance.

	Car Evaluation dataset	Nursery dataset	Contact Lenses Dataset	Iris Plants Dataset
Drift 0	93.52%	98.06%	83.33%	96.00 %
Drift 1	89.93%	98.91%	83.33%	95.33%
Drift 2	87.15%	90.52%	87.50%	96.00 %
Drift 3	90.39%	95.36%	83.33%	95.33%
Drift 4	86.17%	91.89%	83.33%	95.33%
Drift 5	90.97%	93.98%	83.33%	96.00%
Drift 6	81.36%	91.54%	83.33%	95.33%
Drift 7	84.66%	96.17%	87.50%	95.33%
Drift 8	89.35%	99.20%	87.50%	95.33%
Drift 9	84.66%	96.20%	87.50%	94.67%
Drift 10	88.37%		83.33%	96.00%
Drift 11	89.29%		83.33%	96.00%
Drift 12	84.61%		87.50%	98.00%

Table 29: summary of the second committee results

6.4 Results Analysis of the Third Committee

BayesNet, IBK, and JRip are the members of the third committee. In this section, we present our analysis of the results of the third committee. These members are chosen based on several reasons. One of the reasons is to measure the performance of JRip compared to C4.5 that was used in the second committee. Moreover, recalling from Chapter 3, C4.5 does not scale very well on large training sets while JRip scale very well on large training sets.

6.4.1 The results of the first dataset: Car Evaluation Dataset

Recalling from Chapter 5, there are fifteen drift scenarios introduced to the car evaluation dataset. Seven of these drift scenarios are gradual and eight of these drifts are sudden. Table 30 provides the results of the first committee in details. Likewise, it points out where DCCD replaced one of the committee members.

Drift Scenarios		DCCD		
		BayesNet	IBK	JRip
Drift 0 (Original)		85.71%	93.52%	86.69%
Phase 1	Drift 1 (Gradual)	84.78%	89.58%	84.09%
	Drift 2 (Sudden)	84.84%	89.18%	82.64%
	Drift 3 (Gradual)	83.68%	90.39%	85.59%
Total Loss in accuracy - Evaluation Point		-1.28%	-3.80%	-2.58%
Phase 2	Drift 4 (Sudden)	82.70%	83.97%	83.22%
	Drift 5 (Gradual)	84.20%	90.97%	86.46%
	Drift 6 (Sudden)	81.37%	76.04%	76.45%
Total Loss in accuracy - Evaluation Point		-2.95%	-9.86%	-4.65%
Phase 3	Drift 7 (Sudden)	83.68%	81.60%	81.60%
	Drift 8 (Gradual)	83.56%	86.52%	81.48%
	Drift 9 (Sudden)	82.06%	82.58 %	80.56%
Total Loss in accuracy - Evaluation Point		-2.61%	-9.95%	-5.48%
Replacing JRip by SMO		BayesNet	IBK	SMO
Phase 4	Drift 10 (Gradual)	82.29%	84.90%	87.62%
	Drift 11 (Gradual)	84.66%	86.81%	89.41%
	Drift 12 (Sudden)	84.61%	82.23%	85.65%
Total Loss in accuracy - Evaluation Point		-1.86%	-8.87%	-0.06%
Phase 5	Drift 13 (Sudden)	86.05%	88.89%	87.21%
	Drift 14 (Gradual)	84.95%	88.31%	89.12%
	Drift 15 (Gradual)	82.58%	82.64%	86.29%
Total Loss in accuracy - Evaluation Point		-1.18%	-6.91%	0.33%

Table 30: Running the third committee against the Car Evaluation dataset

From drift 0 to drift 5, the most accurate classifier is IBK whereas BayesNet is the most accurate classifiers in drift 6. At the second evaluation point, IBK lost 9.86% of its initial accuracy which is more than the acceptable range for losing accuracy. However, DCCD will not force IBK to retire due to the high weight that IBK has in this committee. Based on DCCD weighting mechanism, IBK has weighting points more than JRip and BayesNet. Hence, IBK is not the worst member performance even though it has lost more than 5% of its initial accuracy. Therefore, all the committee members will continue to the phase 3.

BayesNet is the most accurate member among the members of the third committee in drift 7 but IBK is the most accurate member in drift 8 and drift 9. At the end of phase 3, an

evaluation point takes place which shows that BayesNet lost 2.61% of its initial accuracy. Moreover, IBK and JRip have lost more than 5% of their initial accuracy. However, the weight of IBK is higher than JRip; therefore, DCCD replaces JRip with a new member which is SMO.

The most accurate classifier in phase 4 is SMO. At the end of phase 4, an evaluation point takes place which shows that IBK is the only member that lost more than 5% of its initial accuracy. Based on the weighting mechanism, the weight of IBK is 8 points while the weight of SMO is 3 points and the weight of BayesNet is 1 point. Hence, DCCD will not force IBK to retire because it is not the worst member performance; therefore, all members will continue to phase 5. In drift 13, the most accurate member is IBK while SMO is the most accurate member in drift 14 and drift 15.

6.4.2 The results of the second dataset: Nursery dataset

Nine drift scenarios were introduced into the Nursery dataset as we discussed in Chapter 5. The drift scenarios were divided into five gradual drifts and four sudden drifts. Table 31 indicates the results of running the second committee against the Nursery dataset.

Drift Scenarios		DCCD		
		BayesNet	IBK	JRip
Drift 0 (Original)		90.33%	98.06%	96.98 %
Phase 1	Drift 1 (Gradual)	90.76%	98.20%	97.28 %
	Drift 2 (Sudden)	86.00%	90.36%	85.72%
	Drift 3 (Gradual)	88.23%	95.46%	92.65%
Average Loss in accuracy - Evaluation Point		-2.00%	-3.39%	-5.10%
Replacing JRip by C4.5		BayesNet	IBK	C4.5
Phase 2	Drift 4 (Sudden)	85.49%	91.74%	91.78%
	Drift 5 (Gradual)	87.05%	92.90%	92.59%
	Drift 6 (Sudden)	83.86 %	89.11%	91.17%
Average Loss in accuracy - Evaluation Point		-4.86%	-6.81%	0.07%
Phase 3	Drift 7 (Gradual)	87.18%	93.90%	94.79%
	Drift 8 (Gradual)	90.22%	97.82%	97.49%
	Drift 9 (Sudden)	94.88%	97.82%	94.88%
Average Loss in accuracy - Evaluation Point		0.43%	1.02%	0.93%

Table 31: Running the third committee against the Nursery dataset

In phase 1, IBK is the most accurate member in the committee. The first evaluation point shows that JRip has lost more than 5% from its initial accuracy and its weight is 0 point. Therefore, JRip is the worst member performance and will be replaced with C4.5. In phase 2, C4.5 is the most accurate member in drift 4 and drift 6 while IBK is the most accurate member

in drift 5. The second evaluation point shows that IBK has lost more than 5% of its initial accuracy but it has more weight than BayesNet and C4.5. Therefore, IBK will continue to the next phase because it is not the worst member performance. In phase 4, IBK is the most accurate member in the committee in the entire phase. The last evaluation point shows that all members' accuracy has increased.

6.4.3 The results of the third dataset: Contact Lenses dataset

Table 32 presents the DCCD results for the third committee, which consists of BayesNet, IBK, and JRip. This committee is performed on the Contact-Lenses dataset. We introduced twelve concept drift scenarios; half are gradual drifts and the other half are sudden drifts. Moreover, DCCD forces IBK to retire and adds C4.5 instead, which results in an improvement of the committee's performance.

Drift Scenarios		DCCD		
		BayesNet	IBK	JRip
Drift 0 (Original)		70.83%	70.83%	75.00 %
Phase 1	Drift 1 (Gradual)	70.83%	70.83%	75.00 %
	Drift 2 (Sudden)	75.00 %	66.67%	79.17%
	Drift 3 (Gradual)	70.83%	75.00 %	79.17%
Total Loss in accuracy - Evaluation Point		1.39%	0.00%	2.78%
Phase 2	Drift 4 (Sudden)	70.83%	75.00 %	79.17%
	Drift 5 (Gradual)	79.17%	70.83%	79.17%
	Drift 6 (Sudden)	83.33%	87.50%	83.33%
Total Loss in accuracy - Evaluation Point		6.95%	6.95%	5.56%
Phase 3	Drift 7 (Sudden)	75.00 %	66.67%	79.17%
	Drift 8 (Gradual)	70.83%	62.50 %	87.50%
	Drift 9 (Sudden)	75.00 %	66.67%	79.17%
Total Loss in accuracy - Evaluation Point		2.78%	-5.55%	6.95%
Replacing IBK by C4.5		BayesNet	C4.5	JRip
Phase 4	Drift 10 (Gradual)	70.83%	83.33%	75.00%
	Drift 11 (Gradual)	79.17%	83.33%	83.33%
	Drift 12 (Sudden)	79.17%	83.33%	79.17%
Total Loss in accuracy - Evaluation Point		5.56%	0.00%	4.17%

Table 32: Running the third committee against the Contact Lenses dataset

The accuracy for drift 0 is 70.83% for BayesNet and 70.83% for IBK, while the most accurate member is JRip, at 75.00%. JRip continues to be the most accurate member in the entire phase 1. After drift 3, DCCD starts the first evaluation point, which shows that all members will continue to the next phase. In drift 4, JRip is still the most accurate member in the committee; however, JRip and BayesNet are the most accurate members in drift 5. However, IBK becomes the most accurate member in drift 6. Then, DCCD starts the second evaluation

point, which shows an increase in all members' accuracy compared to their initial accuracy. Therefore, BayesNet, IBK, and JRip will continue to the next phase.

Phase three begins at drift 7, which shows from the table above that JRip, again, is the most accurate classifier in addition to drift 8. Moreover, JRip is still the most accurate member in drift 9. Whereas, IBK loses more than 5% of its initial accuracy; therefore, DCCD forces IBK to retire and replaces it with C4.5 in the next phase. In phase four, the committee becomes BayesNet, C4.5, and JRip. This replacement in the committee makes C4.5, the new member, becomes the most accurate member in the committee in drift 10. The accuracy of C4.5 is 83.33% in drift 10; moreover, the accuracy of C4.5 remains the same and continues to be the most accurate member in drift 11 in addition to JRip. C4.5 is also the most accurate classifier in drift 12 while JRip loses almost 4% of its previous accuracy in drift 12. At the end, C4.5 increases the overall accuracy of DCCD.

6.4.4 The results of the fourth dataset: Iris Plants

Table 33 provides the results of performing the third committee against the Iris dataset. There are twelve drift scenarios that have been introduced to the Iris dataset. Six of these drift scenarios are gradual and six scenarios are sudden.

Drift Scenarios		DCCD		
		BayesNet	IBK	JRip
Drift 0 (Original)		92.67%	95.33%	94.00%
Phase 1	Drift 1 (Gradual)	93.33%	95.33%	94.00%
	Drift 2 (Sudden)	93.33%	96.00%	94.67%
	Drift 3 (Gradual)	93.33%	95.33%	94.00%
Total Loss in accuracy - Evaluation Point		0.66%	0.22%	0.22%
Phase 2	Drift 4 (Sudden)	94.67%	95.33%	94.00%
	Drift 5 (Gradual)	94.00%	96.00%	94.67%
	Drift 6 (Sudden)	94.67%	95.33%	95.33%
Total Loss in accuracy - Evaluation Point		1.78%	0.22%	0.67%
Phase 3	Drift 7 (Sudden)	92.67%	95.33%	96.00%
	Drift 8 (Gradual)	94.00%	95.33%	94.67%
	Drift 9 (Sudden)	94.00%	94.67%	93.33%
Total Loss in accuracy - Evaluation Point		0.89%	-0.22%	0.67%
Phase 4	Drift 10 (Gradual)	89.33%	96.00%	94.67%
	Drift 11 (Gradual)	92.67%	96.00%	95.33%
	Drift 12 (Sudden)	92.67%	98.00%	94.00%
Total Loss in accuracy - Evaluation Point		-1.11%	1.34%	0.67%

Table 33: Running the third committee against the Iris Plants dataset

IBK is the most accurate member in phase 1 and phase 2. In phase 3, JRip is the most accurate member in drift 7 whereas IBK is the most accurate member in drift 8 and drift 9. In phase 4, IBK is the most accurate member in the entire phase. In this dataset, DCCD did not replace any member because all members maintained their loss in accuracy above the acceptable range which is 5%. Table 34 summarizes the results of the third committee which performed on the same four datasets.

	Car Evaluation Dataset	Nursery dataset	Contact Lenses Dataset	Iris Plants Dataset
Drift 0	93.52%	98.06%	75.00 %	95.33%
Drift 1	89.58%	98.20%	75.00 %	95.33%
Drift 2	89.18%	90.36%	79.17%	96.00%
Drift 3	90.39%	95.46%	79.17%	95.33%
Drift 4	83.97%	91.78%	79.17%	95.33%
Drift 5	90.97%	92.90%	79.17%	96.00%
Drift 6	81.37%	91.17%	87.50%	95.33%
Drift 7	83.68%	94.79%	79.17%	96.00%
Drift 8	86.56%	97.82%	87.50%	95.33%
Drift 9	82.58%	97.88%	79.17%	94.67%
Drift 10	87.62%		83.33%	96.00%
Drift 11	89.41%		83.33%	96.00%
Drift 12	85.65%		83.33%	98.00%
Drift 13	88.89%			
Drift 14	89.12%			
Drift 15	86.29%			

Table 34: summary of the third committee results

6.5 Results Analysis of the Fourth Committee

The fourth committee includes three members which are C4.5, SMO, and BayesNet. In this section, we present our analysis of the results of fourth committee. The committee's members are selected based on the popularity and diversity of these classifiers. Additionally, we intend to compare between C4.5, SMO, and BayesNet to measure their reaction to concept drifts. Moreover, we want to measure the performance of SMO on different sizes of datasets.

6.5.1 The results of the first dataset: Car Evaluation dataset

Fifteen drift scenarios were introduced into the car evaluation dataset as mentioned in Chapter 5. Seven of these drift scenarios are gradual and the rest are sudden. Table 35 shows the results of the fourth committee in details. Furthermore, it displays where DCCD replaced one of the committee members.

Drift Scenarios		DCCD		
		C4.5	SMO	BayesNet
Drift 0 (Original)		92.36%	93.75 %	85.71%
Phase 1	Drift 1 (Gradual)	89.93%	91.44%	84.78%
	Drift 2 (Sudden)	87.15%	89.76%	84.84%
	Drift 3 (Gradual)	90.39%	89.93%	83.68%
Total Loss in accuracy - Evaluation Point		-3.20%	-3.37%	-1.28%
Phase 2	Drift 4 (Sudden)	86.17%	89.41%	82.70%
	Drift 5 (Gradual)	90.39%	91.38%	84.20%
	Drift 6 (Sudden)	80.21%	81.89%	81.90%
Total Loss in accuracy - Evaluation Point		-6.77%	-6.19%	-2.78%
Replacing C4.5 by PART		**PART**	**SMO**	**BayesNet**
Phase 3	Drift 7 (Sudden)	83.91%	85.01%	83.68%
	Drift 8 (Gradual)	89.35%	87.96%	83.56%
	Drift 9 (Sudden)	84.66%	83.80%	82.06%
Total Loss in accuracy - Evaluation Point		2.06%	-8.16%	-2.61%
Phase 4	Drift 10 (Gradual)	88.37%	87.62%	82.29%
	Drift 11 (Gradual)	89.29%	89.41%	84.66%
	Drift 12 (Sudden)	84.61%	85.65%	84.61%
Total Loss in accuracy - Evaluation Point		-4.94%	-6.19%	-1.86%
Phase 5	Drift 13 (Sudden)	88.19%	87.21%	86.05%
	Drift 14 (Gradual)	91.15%	89.12%	84.95%
	Drift 15 (Gradual)	85.30%	86.29%	82.58%
Total Loss in accuracy - Evaluation Point		-4.15%	-6.21%	-1.18%

Table 35: Running the fourth committee against the Car Evaluation dataset

SMO is the most accurate member in drift 0, drift 1, and drift 3 whereas C4.5 is the most accurate member in drift 3. In phase 2, SMO is the most accurate member in drift 4 and drift 5 whereas BayesNet is the most accurate member in drift 6. The second evaluation point shows that C4.5 and SMO have lost more than 5% of their initial accuracy. According to the weighting mechanism, SMO's weight is 5 points while C4.5 and BayesNet weight is 1 point. Therefore, DCCD forces C4.5 to retire and then adds a new member which is PART. In phase 3, PART is the most accurate member in drift 8 and 9 while SMO is the most accurate member in drift 7. The third evaluation point shows that SMO has lost more than 5% of its initial accuracy. However, DCCD will not force SMO to retire even though its lost is more than the acceptable range because SMO has the highest weight in the committee.

In phase 4, PART is the most accurate member in drift 10 while SMO is the most accurate in drift 11 and drift 12. The fourth evaluation points shows that only SMO has lost more than 5% of its initial accuracy but SMO still has the highest weight in the committee. Therefore, all members will continue to the next phase. In phase 5, PART is the most accurate member in drift 13 and drift 14. In drift 15, SMO is most accurate member in the committee.

6.5.2 The results of the second dataset: Nursery dataset

Five gradual drifts and four sudden drifts were introduced into the Nursery dataset. In Table 36 we present the results of running the fourth committee against the Nursery dataset. Furthermore, Table 36 points out where DCCD replaced one of the committee members.

Drift Scenarios		DCCD		
		C4.5	SMO	BayesNet
Drift 0 (Original)		97.05%	93.13%	90.33%
Phase 1	Drift 1 (Gradual)	97.45 %	93.10%	90.76%
	Drift 2 (Sudden)	90.52 %	88.23 %	85.99%
	Drift 3 (Gradual)	95.36%	91.98 %	88.22%
Total Loss in accuracy - Evaluation Point		-3.20%	-3.37%	-1.28%
Phase 2	Drift 4 (Sudden)	91.78%	86.70%	85.49%
	Drift 5 (Gradual)	92.59%	89.78 %	87.05%
	Drift 6 (Sudden)	91.17%	88.02%	83.86 %
Total Loss in accuracy - Evaluation Point		-6.77%	-6.19%	-2.78%
Replacing SMO by PART		**C4.5**	**PART**	**BayesNet**
Phase 3	Drift 7 (Gradual)	94.79%	96.17%	90.22%
	Drift 8 (Gradual)	97.49%	99.20%	87.95%
	Drift 9 (Sudden)	94.88%	96.20%	90.33%
Total Loss in accuracy - Evaluation Point		-1.33%	1.02%	-0.83%

Table 36: Running the fourth committee against the Nursery dataset

In the fourth committee, C4.5 is the most accurate member in entire phase1 and phase 2. However, the second evaluation point shows that C4.5 and SMO have lost more than 5% of their initial accuracy. DCCD replaces SMO with a new member which is PART because it has lost more than 5% and it is the worst member performance. Replacing SMO by PART increases the committee's accuracy.

6.5.3 The results of the third dataset: Contact Lenses dataset

The fourth committee also performed on the Contact-Lenses dataset. Table 37 presents the DCCD results for the fourth committee, which consists of C4.5, SMO, and BayesNet. We have introduced twelve concept drift scenarios; half are gradual drifts and the other half are sudden drifts.

Drift Scenarios		DCCD		
		C4.5	SMO	BayesNet
Drift 0 (Original)		83.33%	70.83%	70.83%
Phase 1	Drift 1 (Gradual)	83.33%	70.83%	70.83%
	Drift 2 (Sudden)	87.50%	58.33%	75.00 %
	Drift 3 (Gradual)	83.33%	70.83%	70.83%
Total Loss in accuracy - Evaluation Point		1.39%	-4.17%	1.39%
Phase 2	Drift 4 (Sudden)	83.33%	66.67%	70.83%
	Drift 5 (Gradual)	83.33%	70.83%	79.17%
	Drift 6 (Sudden)	83.33%	66.67%	83.33%
Total Loss in accuracy - Evaluation Point		0.00%	-2.77%	6.95%
Phase 3	Drift 7 (Sudden)	87.50%	58.33%	75.00 %
	Drift 8 (Gradual)	87.50%	70.83%	70.83%
	Drift 9 (Sudden)	87.50%	58.33%	75.00%
Total Loss in accuracy - Evaluation Point		4.17%	-8.33%	2.78%
Replacing SMO by PART		C4.5	PART	BayesNet
Phase 4	Drift 10 (Gradual)	87.50%	83.33%	70.83%
	Drift 11 (Gradual)	87.50%	83.33%	79.17%
	Drift 12 (Sudden)	87.50%	83.33%	79.17%
Total Loss in accuracy - Evaluation Point		4.17%	0.00%	5.56%

Table 37: Running the fourth committee against the Contact Lenses dataset

C4.5 is the most accurate member in phase 1, phase 2, and phase 3. The first three evaluation points show that the accuracy of C4.5 and BayesNet increases over time while the accuracy of SMO decreases over time. However, the third evaluation point shows that SMO has lost more than 5% of its initial accuracy in addition to its low weight. Therefore, DCCD forces SMO to retire and adds a new member to the committee which is PART. In phase 4, C4.5 remains the most accurate member in the committee for the entire phase.

6.5.4 The results of the fourth dataset: Iris Plants dataset

This committee is applied to the Iris Plants dataset. There are six gradual drifts and six sudden drifts that have been applied to this datasets. DCCD divides these twelve drifts into four phases. However, DCCD does not replace any of its members on this dataset because none of the conditions that DCCD considers appear. Table 38 presents the results of the fourth committee on

the Iris dataset. Moreover, by inspection, during any drift scenario, the acceptable range that the accuracy might drop or increase to is 5%.

The accuracy of drift 0 in the Iris Plants dataset is 96.00% for C4.5 and SMO, which makes them the most accurate members in the fourth committee. In phase 1, the most accurate member is SMO for the entire phase. The first evaluation point shows that the accuracy of SMO and BayesNet increased slightly whereas the accuracy of C4.5 decreases. C4.5 remains the most accurate member in the committee for the entire phase 2.

Drift Scenarios		DCCD		
		C4.5	SMO	BayesNet
Drift 0 (Original)		96.00%	96.00 %	92.67%
Phase 1	Drift 1 (Gradual)	95.33%	96.67%	93.33%
	Drift 2 (Sudden)	95.33%	96.67%	93.33%
	Drift 3 (Gradual)	95.33%	96.00%	93.33%
Total Loss in accuracy - Evaluation Point		-0.67%	0.45%	0.66%
Phase 2	Drift 4 (Sudden)	95.33%	96.00%	94.67%
	Drift 5 (Gradual)	95.33%	96.00%	94.00%
	Drift 6 (Sudden)	95.33%	96.00%	94.67%
Total Loss in accuracy - Evaluation Point		-0.67%	0.00%	1.78%
Phase 3	Drift 7 (Sudden)	94.67%	93.33%	92.67%
	Drift 8 (Gradual)	94.00%	96.00%	94.00 %
	Drift 9 (Sudden)	94.67%	95.33%	94.00 %
Total Loss in accuracy - Evaluation Point		-1.55%	-1.11%	0.89%
Phase 4	Drift 10 (Gradual)	94.00%	95.33%	89.33%
	Drift 11 (Gradual)	94.00%	95.33%	92.67%
	Drift 12 (Sudden)	94.00%	94.00%	92.67%
Total Loss in accuracy - Evaluation Point		-2.00%	-1.11%	-1.11%

Table 38: Applying DCCD against the Iris Plants dataset

In phase three, C4.5 becomes the most accurate member in drift 7 while SMO is the most accurate member in drift 8 and drift 9. The third evaluation point shows that SMO starts losing of its initial accuracy. Phase 4 shows that SMO is the most accurate member in drift 10 and drift 11. In drift 12, C4.5 and SMO are the most accurate members in the committee. Finally, Table 39 summarizes the results of performing the fourth committee against the previously discussed datasets.

	Car Evaluation Dataset	Nursery dataset	Contact Lenses Dataset	Iris Plants Dataset
Drift 0	93.75 %	97.05%	83.33%	96.00 %
Drift 1	91.44%	97.45 %	83.33%	96.67%
Drift 2	89.76%	90.52 %	87.50%	96.67%
Drift 3	90.39%	95.36%	83.33%	96.00 %
Drift 4	89.41%	91.78%	83.33%	96.00 %
Drift 5	91.38%	92.59%	83.33%	96.00 %
Drift 6	81.90%	91.17%	83.33%	96.00 %
Drift 7	85.01%	96.17%	87.50 %	94.67%
Drift 8	89.35%	99.20%	87.50 %	96.00 %
Drift 9	84.66%	96.20%	87.50 %	95.33%
Drift 10	88.37%		87.50 %	95.33%
Drift 11	89.41%		87.50 %	95.33%
Drift 12	85.65%		87.50 %	94.00%
Drift 13	88.19%			
Drift 14	91.15%			
Drift 15	86.29%			

Table 39: Summary of the fourth committee results

6.6 Discussion

In this section, we discuss our experimental results as obtained by the DCCD algorithm when constructing a variety of committees on four different datasets. Table 40 presents an evaluation of the classifiers that we used in the committees, followed by our analysis. We rate the committee members based on their accuracy using four score levels. When a classifier obtains **Excellent** that means this classifier is always the most accurate classifier in a committee or there is no significant difference when compared to the most accurate members. Moreover, if a classifier that obtains **Very-Good** it implies that the classifier has a competitive performance with the best classifier in a committee. Furthermore, if a classifier never becomes the best classifier in a committee, but its performance is comparable with the most accurate classifier in a committee, this classifier is rated as Good. Finally**,** when a classifier obtains a rating of **Acceptable**, it means that the classifier usually do not perform that well and often has to be replaced by a new classifier.

	Car Evaluation dataset (Medium size)		Nursery dataset (Large size)		Contact lenses dataset (Very small size)		Iris dataset (Small size)	
	Gradual	**Sudden**	**Gradual**	**Sudden**	**Gradual**	**Sudden**	**Gradual**	**Sudden**
C4.5	Excellent	Excellent	Excellent	V-good	Excellent	Excellent	Excellent	Excellent
JRip	Acceptable	Acceptable	Acceptable	Acceptable	Excellent	Excellent	V-good	V-good
BayesNet	Acceptable	Acceptable	Acceptable	Acceptable	Acceptable	Acceptable	V-good	V-good
PART	V-Good	V-Good	Excellent	Excellent	V-good	V-good		
KStar					Acceptable	Acceptable		
IBK	V-Good	Acceptable	V-Good	Acceptable	Acceptable	Acceptable	Excellent	Excellent
SMO	V-Good	V-Good	Acceptable	Acceptable	Acceptable	Acceptable	Excellent	Excellent

Table 40: Performance evaluation of the committees' members

Based on Table 40, C4.5 is the best classifier that handles concept drift on all datasets sizes that we used in this study. **Recall from section 3.1.1**, C4.5 uses the attribute evaluation measure to rate the value of an attribute for predicting the class attribute in order to select the root node. Also, recall that concept drift is often caused by a change in the data distribution, which might lead to a change in the attribute's rating. This explains the high performance that C4.5 obtained in our experiment.

JRip has an acceptable performance on large and medium datasets while its performance on small datasets is excellent. The reason why JRip has only an acceptable performance on large and medium datasets is that it requires a large investment, in terms of time, to build the model and to evaluate the feature set. In addition, highly accurate results can only be achieved by running the algorithm many times. JRip performs excellent on small datasets, because it uses propositional rule learner to ensure that errors are minimal. Also, the phase-by-phase implementation of the algorithm ensures that the overall results are as high as possible. During the rule set growing phase, the condition with the highest information gain is picked to ensure that the rule set is correct.

The performance of BayesNet is acceptable on all datasets sizes except for small datasets, which is satisfactory in general. BayesNet has an updater that enables the classifier to learn from new data samples; this explains the satisfactory performance. However, BayesNet is limited to datasets that have a few variables. Also, BayesNet requires an expert to give domain information for the creation of the network.

The performance of PART is excellent on almost all datasets sizes, which make it comparable to C4.5. The reason behind this excellent performance is PART that it is able to determine concept drift, by using the divide and conquers methodology, in conjunction with

decision trees. Moreover, PART avoids the over-pruning problem of the basic separate and conquer rule learner.

In gradual concept drifts, IBK performs well on medium and large datasets. **Recalls from section 3.6** that IBK assumes that similar instances usually have similar classification. Therefore, IBK classifies instances by comparing them to other pre-classified examples using a distance measures for all the recorded attributes. This mechanism provides to IBK a number of chances to classify the new instances correctly because large datasets provide a large number of pre-classified that will be used to classify the new instances. In contrast, IBK and KStar, the only dataset that KStar ran against, performed acceptably on the very small dataset. **Recalls from section 3.2** that KStar also assumes that similar instances usually have similar classification. This means that small datasets that provide a small number of pre-classified could be used. Hence, IBK and KStar do not perform very well on small datasets, except for the dataset that has high harmonious values such as the Iris dataset. Moreover, IBK performs acceptably in the presence of sudden concept drifts regardless of the datasets size. The sudden concept drift is caused by a radical change in the dataset distribution. This makes the generated distance measures that were legitimate before the presence of sudden concept drifts, illegitimate after.

Lastly, SMO performs well on medium and small datasets while its performance on large and very small datasets is acceptable. SMO does not depend on the size of the dataset, but based on its complexity. Here, the complexity in a dataset includes missing data, the number of altitudes, and the number of target classes. SMO works to ensure that the problems are broken down into the smallest problems, which give SMO the ability to preforms well in most of the large problems.

In summary, the main benefit of the DCCD algorithm is that it detects when to replace bad performing members in the committee by a new member, which increases the accuracy over time more effectively than the regular algorithms. To evaluate our algorithm, DCCD is given sequence of drift scenarios; these sequences are divided by the number of phases. At the end of each phase, an evaluation point will take place to determine whether a member needs to be replaced or not. The member that causes the most decrease in the accuracy will be considered the worst member in terms of performance and therefore will be moved to the blacklist.

6.7 Summary

In this chapter, we evaluated the performance of the DCCD algorithm when sequences of gradual and sudden concept drifts occur. We studied the effectiveness of DCCD by running four committees on benchmark datasets. Furthermore, we measured the reaction of the classifiers, as overviewed in chapter 3, in terms of handling gradual and sudden concept drifts. We provided an evaluation for the classifiers that we used in the committees in term of how well they handle concept drift. Our evaluation suggests that, for the datasets used, the C4.5 decision tree learner performs best against all data set sizes. Chapter 7 provides the results of a comparative experiment performed on the DCCD by comparing it to the state-of-the-art MultiScheme algorithm.

CHAPTER SEVEN

COMPARATIVE EVALUATION

CHAPTER SEVEN

COMPARATIVE EVALUATION

This chapter presents our comparative evaluation between our algorithm (DCCD) and the MultiScheme algorithm. MultiScheme is an ensemble learning approach which is implemented in WEKA and has similar mechanisms to DCCD. The comparison between DCCD and MultiScheme is performed by running them on sequences of gradual and sudden concept drifts. DCCD and MultiScheme enable users to train several of base classifiers and then select the best classifier among them.

7.1 MultiScheme Algorithm

Len Trigg [26] introduced the MultiScheme algorithm, which implements a committee of learning classifiers, in 1999. MultiScheme uses a cross validation mechanism to select the best classifier among a set of diverse classifiers. In the classification, the percentage correctly classified examples is used as the performance measurement. MultiScheme trains the selected classifiers one-by-one by the same training set and sets them by the same testing set. The main idea behind this is to ensure that the performance of the selected classifiers is measured fairly [26].

MultiScheme creates a fixed size array to work as a list of classifiers, as shown in Figure 25. The created array includes the names of the selected classifiers followed by the scheme options. Moreover, the array will be useless without diverse classifiers as will be

discussed in the example in this section. MultiScheme assumes that the first classifier in the array is the best classifier unless another classifier has a lower error rate. If the new classifier has a lower error rate, MultiScheme considers the new classifier as the best classifier [26].

MultiScheme Algorithm

Input:

- $Dataset_i$, The selected dataset;
- $ErrorRate_i$ The error percentage of the current classifier for classification
- ClassifiersList Classifiers to choose from (default ==”null”);
- BestClassifier, The name that has the least error percentage (default ==”null”);
- BestPerformance, The error percentage of the *BestClassifier* (default ==”null”);
- X, Number of folds (default 0, is to use training error);
- S, Random number seed (default== 1);
- B, Classifier specification (the names of the selected classifiers followed by the scheme options. default: "ZeroR”);
- D, Random number seed (default== 1);

Output: Selecting the classifier that had the best performance on training data among various classifiers, using cross validation on the training data or the performance on the training data. Performance is measured based on percent correct (classification) or mean-squared error (regression).

Method:

(1) *Add* $Dataset_i$*;*
(2) *Create ClassifiersList;*
(3) *Add classifiers into the ClassifiersList;*
(4) *Set X, S, B, and D;*
(5) *Prepare* $Dataset_i$ *(if we used 10 cross validation, MS selects 9 folds for training and 1 for testing)*

(6) ***While*** *the ClassifiersList has untrained classifier* ***do***
{
(7) *Select a classifier from the ClassifiersList,*
(8) *Train the current classifier by the same training set ,*
(9) *Calculate the ErrorRate of the current classifier,*

ErrorRate = The error percentage of the current classifier for classification

(10) ***If*** (*ErrorRate of current classifier < BestPerformance)*
{
(11) *BestPerformance == ErrorRate of the current classifier,*
(12) *BestClassifier == current classifier,*
{
(13) ***EndIF;***
}
(14) ***End while;***
(15) ***Return*** *BestPerformance and BestClassifier;*

Figure 25: MultiScheme Algorithm (Classification) [26]

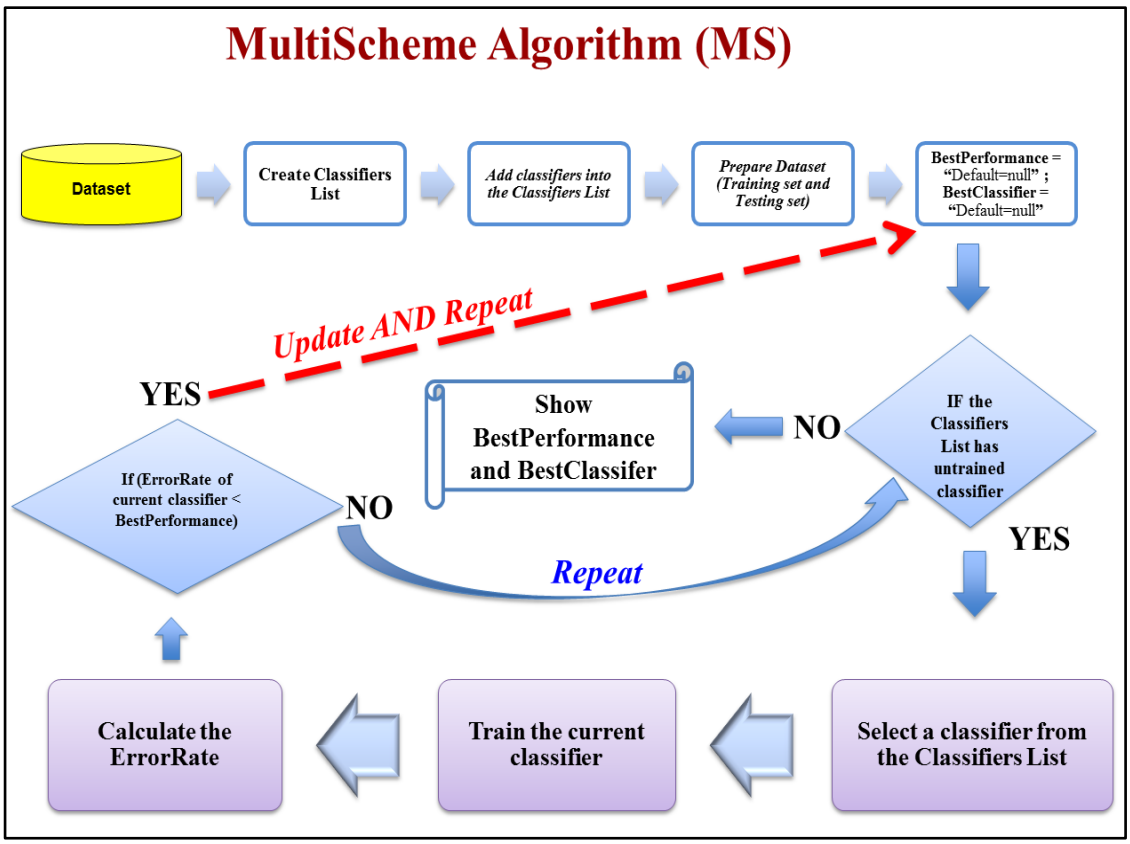

Figure 26: MultiScheme architecture

For example, suppose we performed the MultiScheme algorithm against the Car Evaluation dataset. Also, suppose we selected three diverse classifiers, namely BayesNet, C4.5, and JRip. MultiScheme creates a fixed size array to work as a list of the selected classifiers, used to choose the classifier that had the best performance on training data. The performance is measured based on the error rate of the classifiers. Figure 27 shows the initial phase in the MultiScheme algorithm.

BayesNet	C4.5	JRip

Variable	Values
BestPerformance	Default: "null"
BestClassifier	Default: "null"

Figure 27: The initial phase in MultiScheme

MultiScheme trains the selected classifiers one-by-one by the same training set starting from the first classifier in the array. Suppose the error rate of BayesNet, the current classifier, is 15%. MultiScheme considers BayesNet as the best classifier, if the error rate of current classifier less than the best performance MultiScheme compares the error rate of BayesNet (which is 15%) to the best classifier's performance which is "null" because BayesNet is the first trained classifier ah shown in Figure 28.

BayesNet	C4.5	JRip
15%	Not trained yet	Not trained yet

Variable	Values
BestPerformance	15%
BestClassifier	BayesNet

Figure 28: MultiScheme considers the first classifiers as the best performance

C4.5 is the second selected classifier that MultiScheme will train. Suppose the error rate of C4.5 is 7%. MultiScheme compares the error rate of the best performance, which is 15%, to the error rate of the current classifier, C4.5, which is 7%. Based on the MultiScheme algorithm, when current classifier has **a lower** error rate than the best performance, MultiScheme considers the current classifier as the best performance. Therefore, MultiScheme considers C4.5 as the best classifier, as shown in Figure 29.

BayesNet	C4.5	JRip
15%	7%	Not trained yet

Variable	Values
BestPerformance	7%
BestClassifier	C4.5

Figure 29: MultiScheme considers the current classifier as the best classifier

Finally, MultiScheme repeats the same technique for the third classifier, which is JRip. Suppose the error rate of JRip is 14% which is **higher** than the error rate of the best performance. Therefore, MultiScheme keeps C4.5 as the best classifier as shown in Figure 30.

BayesNet	C4.5	JRip
15%	7%	14%

Variable	Values
BestPerformance	7%
BestClassifier	C4.5

Figure 30: MultiScheme keeps the second classifier as the best classifier

Next, our comparative evaluation is performed by running several committees of DCCD and MultiScheme against the previously introduced four different datasets. Thus, in this chapter, we studied the effectiveness of DCCD by running it on the previously introduced datasets, of different sizes. Furthermore, we measured the reaction of the two approaches in terms of handling gradual and sudden concept drifts.

7.2 Experiment Setup

In this experiment, we ran four DCCD and MS committees against the four datasets that that were affected by gradual and sudden concept drifts. **In this experiment, we use three base classifiers in all DCCD and MS committees, as discussed in section 4.4. Moreover,** comparing the performance of four DCCD and MS committees that has the same members in each committee can provide a good generalization. For the fairness and consistency of the experiment, all DCCD and MS committees are trained by the same sequence of drift scenarios that were introduced into the selected datasets. By inspection, we decided to setup the accuracy threshold *t* by -5% for DCCD Furthermore, to evaluate the performance of DCCD, we apply a ten-fold cross validation method for all experiments and we used the accuracy as the main measure. Also, we use an open-source Java-based machine learning called WEKA to perform this experiment.

7.3 Evaluation of the First Committee

In this section, we compare DCCD to MultiScheme in terms of the accuracy. In Table 41, we present our comparison results between MultiScheme and DCCD on the previously discussed datasets. As you may notice, both approaches have the same initial classifiers; JRip, C4.5, and BayesNet. Additionally, Table 41 lists the results details of the comparison between the MultiScheme and DCCD on the four datasets as shown in Tables (a), (b), (c), and (d).

	DCCD	MultiScheme
Drift 0	92.36%	92.36%
Drift 1	89.93%	89.93%
Drift 2	87.15%	87.15%
Drift 3	90.39%	90.39%
Drift 4	86.17%	86.17%
Drift 5	90.39%	90.39%
Drift 6	81.37%	81.37%
Drift 7	84.66%	84.66%
Drift 8	86.92%	86.92%
Drift 9	83.04%	83.04%
Drift 10	87.61%	86.23%
Drift 11	89.40%	89.23%
Drift 12	85.65%	84.38%
Drift 13	89.29%	89.29%
Drift 14	89.12%	88.48%
Drift 15	86.28%	84.43%

(a) Comparison between the first committee and MultiScheme on the Car Evaluation dataset

	DCCD	MultiScheme
Drift 0	97.05%	97.18%
Drift 1	97.45 %	97.48%
Drift 2	90.52 %	90.51 %
Drift 3	95.36%	95.35%
Drift 4	91.75%	91.77%
Drift 5	92.59%	92.58%
Drift 6	91.17%	91.16%
Drift 7	94.79%	94.79%
Drift 8	97.49%	97.33%
Drift 9	94.88%	94.88%

(b) Comparison between the first committee and MultiScheme on the Nursery dataset

	DCCD	MultiScheme
Drift 0	83.33%	70.83%
Drift 1	83.33%	70.83%
Drift 2	87.50%	75.00%
Drift 3	83.33%	70.83%
Drift 4	83.33%	70.83%
Drift 5	83.33%	70.83%
Drift 6	83.33%	79.17%
Drift 7	87.50%	75.00 %
Drift 8	87.50%	70.83%
Drift 9	87.50%	75.00%

(c) Comparison between the first committee and MultiScheme on the Contact Lenses dataset

	DCCD	MultiScheme
Drift 0	96.00 %	96.00 %
Drift 1	95.33%	95.33%
Drift 2	95.33%	95.33%
Drift 3	95.33%	95.33%
Drift 4	95.33%	95.33%
Drift 5	95.33%	95.33%
Drift 6	95.33%	95.33%
Drift 7	96.00 %	93.33%
Drift 8	94.67%	94.67%
Drift 9	94.67%	94.00 %
Drift 10	95.33%	94.00 %
Drift 11	95.33%	94.00 %
Drift 12	94.00%	94.00 %

(d) comparison between the first committee and MultiScheme on the Iris Plant dataset

Table 41: Comparison between the first DCCD and MultiScheme

Table (a) presents the performance of DCCD and MultiScheme on the sequence of concept drift scenarios that introduced on the Car Evaluation dataset. The difference in the

accuracy is highlighted in Figure 31. From drift 0 until drift 9, the results of DCCD and MultiScheme are identical, but after drift 9 DCCD is always more accurate than MultiScheme.

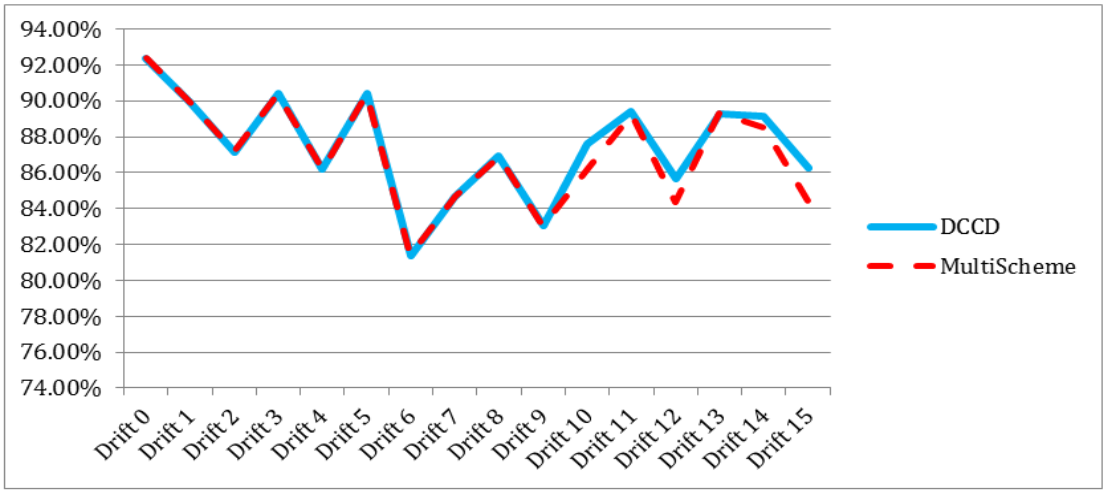

Figure 31: Comparison of the accuracy of the first DCCD and MultiScheme on the Car Evaluation dataset

Table (b) shows that DCCD and MultiScheme have a similar performance for the entire committee that was performed on the nursery dataset. The reason behind that is when DCCD replaced JRip with SMO after drift 6; C4.5 continued to be the most accurate member in the committee. Figure 32 highlights the performance of DCCD and MultiScheme on this dataset which shows that the accuracy of DCCD is equal to the accuracy of MultiScheme.

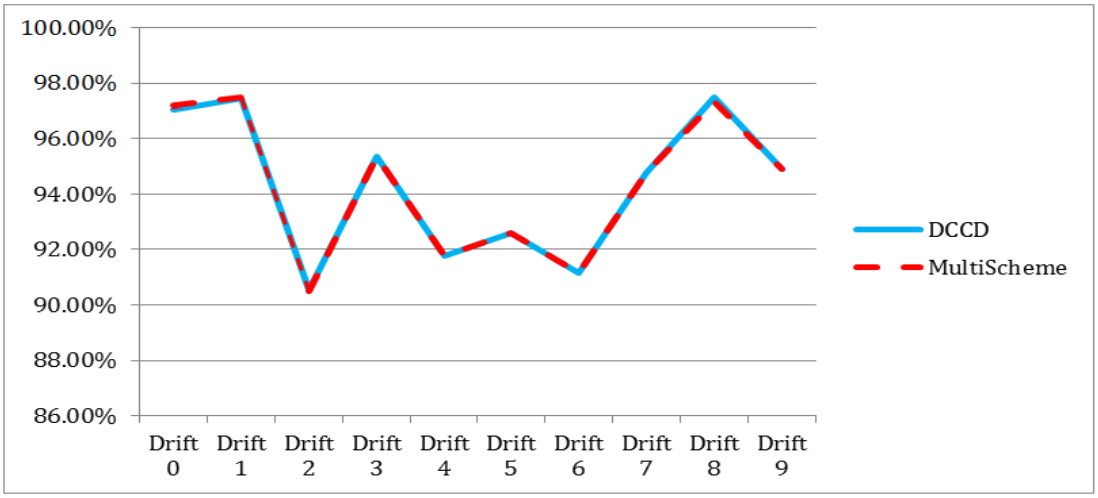

Figure 32: Comparison of the accuracy of the first DCCD and MultiScheme on the Nursery dataset

In Table (c) we provide the detailed of the comparison between DCCD and MultiScheme by performing them on the Contact Lenses datasets. In this committee, DCCD did not replace one of its committee members; however, DCCD maintains higher accuracy than

MultiScheme. This indicates that DCCD has a higher performance even when DCCD did not use its great advantage which is the dynamics that DCCD has. The difference in the accuracy is highlighted in Figure 33, which compares DCCD and MultiScheme on the sequence of concept drift scenarios that were introduced on this dataset.

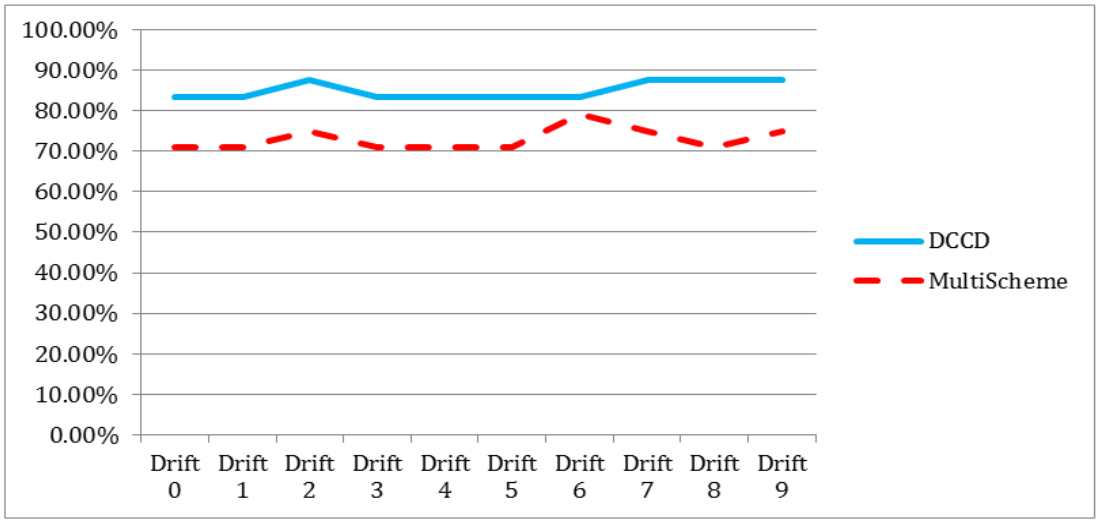

Figure 33: Comparison of the accuracy of the first DCCD and MultiScheme on the Contact Lenses dataset

A comparison between the performance of DCCD and MultiScheme is presented in Table (d). This comparison is performed by running them against sequence of concept drift scenarios that were introduced into the Iris Plants dataset. DCCD and MultiScheme have similar performance until drift 5 as highlighted in Figure 34. The accuracy of DCCD has increased after drift 5 which indicates that DCCD able to limit the accuracy reduction that is caused by the concept drift.

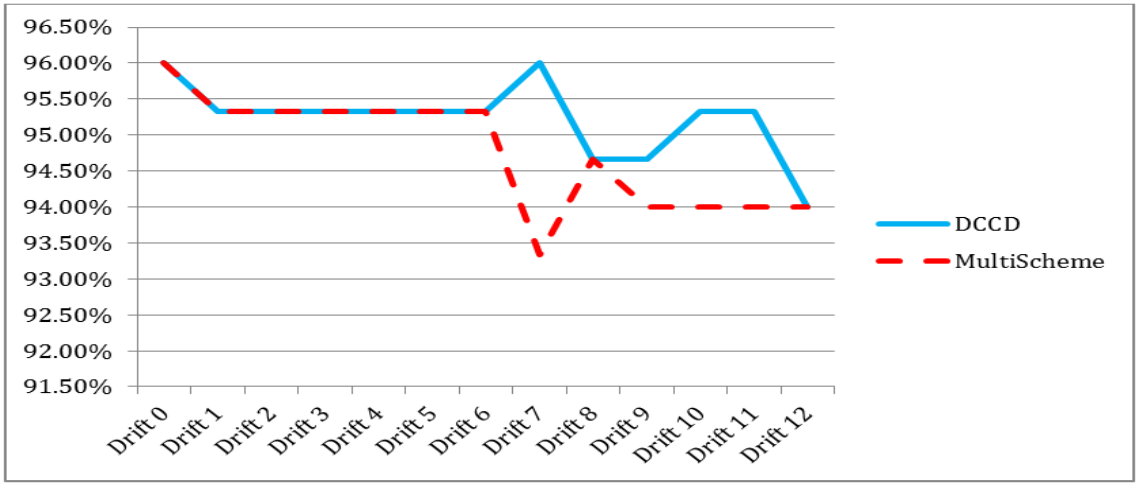

Figure 34: Comparison of the accuracy of the first DCCD and MultiScheme on the Iris Plants dataset

7.4 Evaluation of the Second Committee

In this section, we compare the second DCCD to MultiScheme in terms of the accuracy performance. Notice that both approaches have the same initial committee members; BayesNet, IBK, and C4.5. Moreover, DCCD replaces the worst member with a new member in some of its committees whereas MultiScheme cannot add or remove any of its members. We present the results details of the comparison between the MultiScheme and DCCD in Table 42 (a), (b), (c), and (d).

	DCCD	MultiScheme
Drift 0	93.52%	93.52%
Drift 1	89.93%	89.70%
Drift 2	87.15%	88.95%
Drift 3	90.39%	90.39%
Drift 4	86.17%	86.17 %
Drift 5	90.97%	90.68%
Drift 6	81.36%	76.04%
Drift 7	84.66%	81.60%
Drift 8	89.35%	86.92%
Drift 9	84.66%	82.58 %
Drift 10	88.37%	86.23%
Drift 11	89.29%	89.24%
Drift 12	84.61%	82.23%

(a) Comparison between the second DCCD and MultiScheme on the Car Evaluation dataset

	DCCD	MultiScheme
Drift 0	98.06%	98.06%
Drift 1	98.91%	98.91%
Drift 2	90.52%	90.52%
Drift 3	95.36%	95.36%
Drift 4	91.89%	88.36%
Drift 5	93.98%	90.08%
Drift 6	91.54%	87.26%
Drift 7	96.17%	93.13 %
Drift 8	99.20%	98.74%
Drift 9	96.20%	92.97%

(b) Comparison between the second DCCD and MultiScheme on the Nursery dataset

	DCCD	MultiScheme
Drift 0	83.33%	70.83%
Drift 1	83.33%	70.83%
Drift 2	87.50%	75.00 %
Drift 3	83.33%	70.83%
Drift 4	83.33%	70.83%
Drift 5	83.33%	70.83%
Drift 6	83.33%	79.17%
Drift 7	87.50%	75.00 %
Drift 8	87.50%	70.83%
Drift 9	87.50%	75.00 %
Drift 10	83.33%	70.83%
Drift 11	83.33%	79.17%
Drift 12	87.50%	79.17%

(c) Comparison between the second DCCD and MultiScheme on the contact lenses dataset

	DCCD	MultiScheme
Drift 0	96.00 %	94.67%
Drift 1	95.33%	95.33%
Drift 2	96.00 %	94.67%
Drift 3	95.33%	95.33%
Drift 4	95.33%	94.67%
Drift 5	96.00%	94.67%
Drift 6	95.33%	95.33%
Drift 7	95.33%	95.33%
Drift 8	95.33%	94.67%
Drift 9	94.67%	94.00 %
Drift 10	96.00 %	92.67%
Drift 11	96.00 %	94.67%
Drift 12	98.00 %	97.33%

(d) Comparison between the second DCCD and MultiScheme on the Iris Plants dataset

Table 42: Comparison between the second DCCD and MultiScheme

MultiScheme and the second DCCD are performed on the Car Evaluation dataset as shown in Table (a) and highlighted in Figure 35. The results indicate that the performance of MultiScheme and DCCD are identical until drift 6. After drift 6, DCCD replaces IBK by PART,

which explains the improvement in the performance of DCCD as mentioned in **section 6.2.1**. Figure 35 shows that the accuracy of DCCD increased after the replacement that DCCD performed.

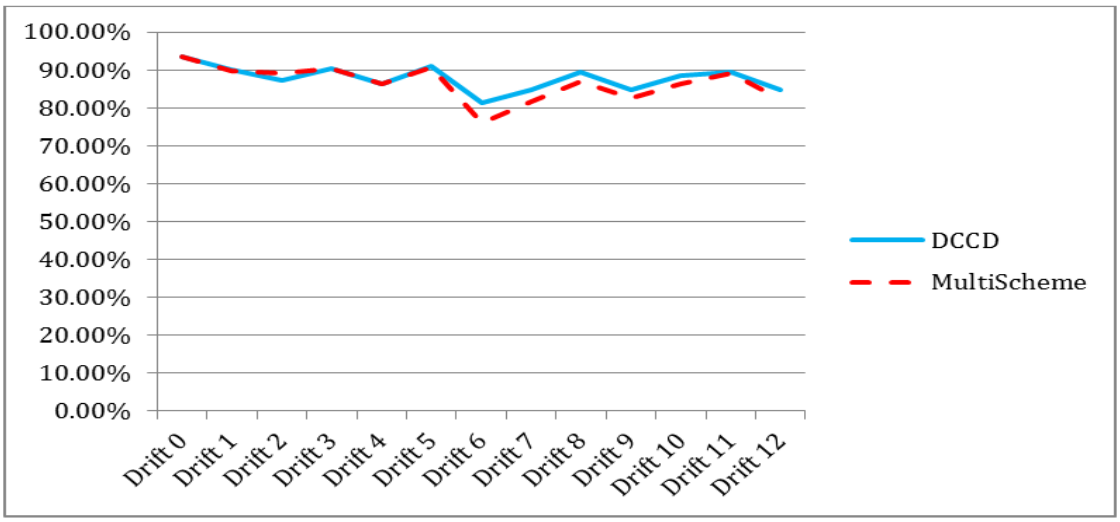

Figure 35: Comparison of the accuracy of the second DCCD and MultiScheme on the Car Evaluation dataset

A comparison between DCCD and MultiScheme on sequence of concept drift scenarios that have been applied into the Nursery dataset is presented in Table (b). Moreover, we highlight the difference in the accuracy in Figure 36. DCCD have the same performance until drift 3 and then DCCD shows an increase in its accuracy due to the dynamic mechanism of replacing the worst member in the committee as mentioned in **section 6.2.2**. Figure 36 shows that the accuracy of DCCD increases when DCCD uses its dynamic mechanism which indicates that the dynamic mechanism in DCCD handles concept drift properly.

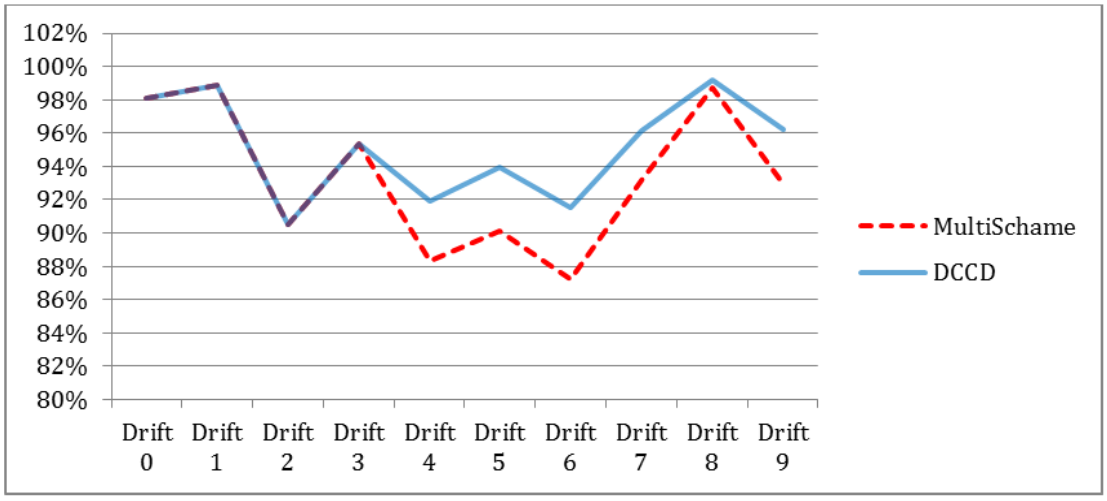

Figure 36: Comparison of the accuracy of the second DCCD and MultiScheme on the Nursery dataset

The comparison between the second DCCD and MultiScheme presented in Table (c) and in Figure 37 which shows their performance against the Contact Lenses datasets. Recall from **section 6.2.3**, DCCD replaces IBK by KStar after drift 9; however, DCCD maintains higher accuracy than MultiScheme before and after the replacement took place. This indicates that DCCD has higher accuracy even when DCCD did not use its dynamical mechanism.

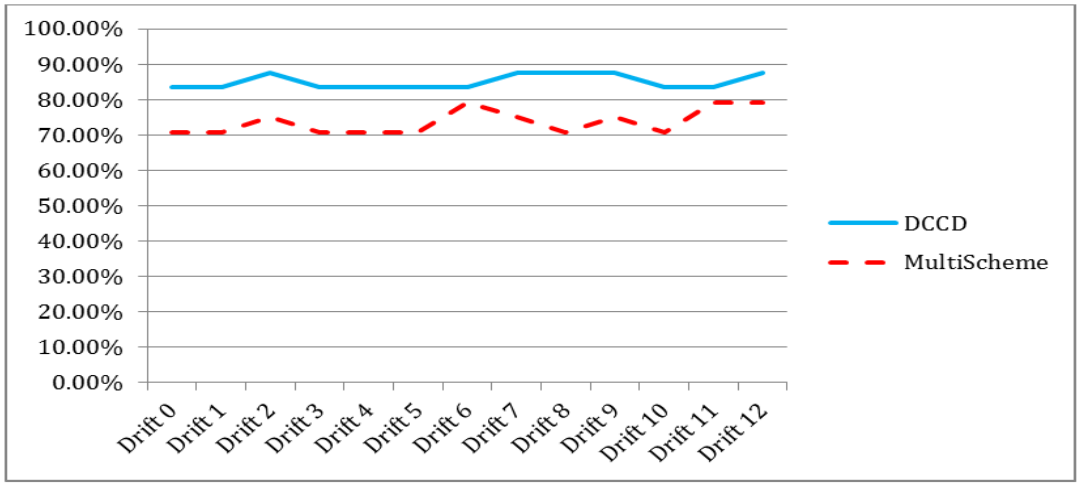

Figure 37: Comparison of the accuracy of the second DCCD and MultiScheme on the Contact Lenses dataset

Table (d) shows that the performance of DCCD is higher than MultiScheme in almost all drifts. In this committee, DCCD did not replace any of its members; however, DCCD is more accurate than MultiScheme. Figure 38 highlights the difference in the accuracy between DCCD and MultiScheme on the sequence of concept drift scenarios that occur on the Iris Plants dataset. Figure 38 points out DCCD ability of limiting the loss in the accuracy that is caused by the concept drifts.

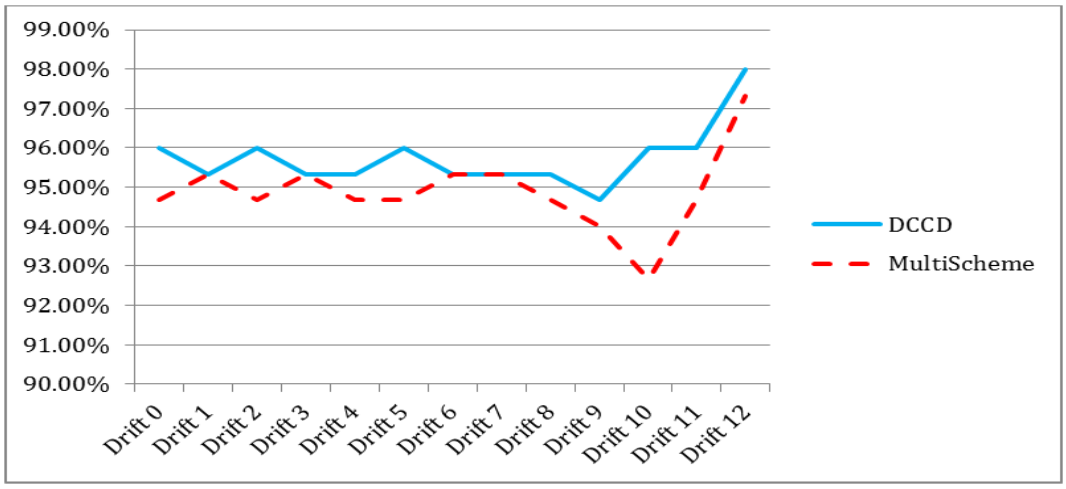

Figure 38: Comparison of the accuracy of the second DCCD and MultiScheme on the Iris Plants dataset

7.5 Evaluation of the Third Committee

This section provides a comparison between the third DCCD and by running DCCD and MultiScheme on the four datasets. Both DCCD and MultiScheme have the same initial classifiers: BayesNet, IBK, and JRip. Table 43 presents the results of the comparative evaluation between DCCD and MultiScheme on the four dataset as shown in Tables (a), (b), (c), and (d).

	DCCD	MultiScheme
Drift 0	93.52%	93.52%
Drift 1	89.58%	89.58%
Drift 2	89.18%	89.18%
Drift 3	90.39%	90.39%
Drift 4	83.97%	84.03%
Drift 5	90.97%	90.97%
Drift 6	81.37%	76.04%
Drift 7	83.68%	81.60%
Drift 8	86.52%	86.52%
Drift 9	82.58 %	82.58 %
Drift 10	87.62%	84.90%
Drift 11	89.41%	86.81%
Drift 12	85.65%	82.23%
Drift 13	88.89%	88.89%
Drift 14	89.12%	88.31%
Drift 15	86.29%	82.97%

(a) Comparison between the third DCCD and MultiScheme on the Car Evaluation dataset

	DCCD	MultiScheme
Drift 0	98.06%	98.06%
Drift 1	98.20%	98.20%
Drift 2	90.36%	90.36%
Drift 3	95.46%	95.46%
Drift 4	91.78%	91.74%
Drift 5	92.90%	92.90%
Drift 6	91.17%	89.10%
Drift 7	94.79%	93.90%
Drift 8	97.82%	97.40%
Drift 9	97.88%	93.86 %

(b) Comparison between the third DCCD and MultiScheme on the Nursery dataset

	DCCD	MultiScheme
Drift 0	75.00 %	75.00 %
Drift 1	75.00 %	70.83%
Drift 2	79.17%	66.67%
Drift 3	79.17%	75.00 %
Drift 4	79.17%	79.17%
Drift 5	79.17%	75.00 %
Drift 6	87.50%	83.33%
Drift 7	79.17%	66.67%
Drift 8	87.50%	62.50%
Drift 9	79.17%	66.67%
Drift 10	83.33%	79.17%
Drift 11	83.33%	70.83%
Drift 12	83.33%	79.17%

(c) Comparison between the third DCCD and MultiScheme on the contact lenses dataset

	DCCD	MultiScheme
Drift 0	95.33%	95.33%
Drift 1	95.33%	95.33%
Drift 2	96.00%	96.00%
Drift 3	95.33%	95.33%
Drift 4	95.33%	95.33%
Drift 5	96.00%	96.00%
Drift 6	95.33%	95.33%
Drift 7	96.00%	96.00%
Drift 8	95.33%	95.33%
Drift 9	94.67%	94.67%
Drift 10	96.00%	96.00%
Drift 11	96.00%	96.00%
Drift 12	98.00%	98.00%

(d) Comparison between the third DCCD and MultiScheme on the Iris Plants dataset

Table 43: Comparison between the third DCCD and MultiScheme

Table (a) shows the performance results of the third DCCD and MultiScheme for the Car Evaluation dataset. The accuracy of DCCD and MultiScheme are the same from drift 0 to drift 5. DCCD has a slightly higher accuracy from drift 6 till drift 15. Recall form **section 6.3.1,**

DCCD replaced JRip by SMO after drift 9, which resulted in a small improvement in the performance of DCCD as shown in Figure 39.

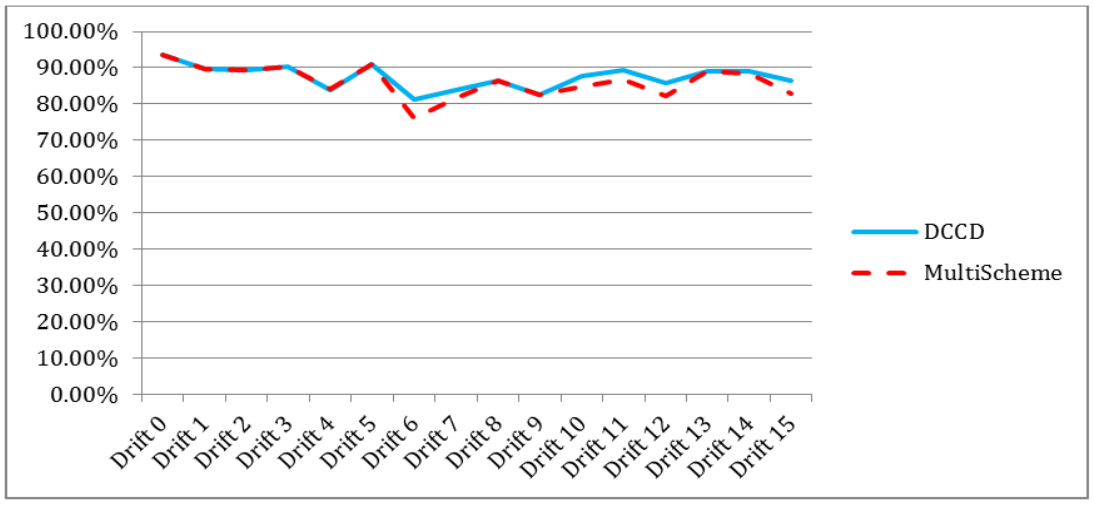

Figure 39: Comparison between the third DCCD and MultiScheme on the Car Evaluation dataset

Table (b) and Figure 40 show the accuracy of the third DCCD and MultiScheme which was performed on the Nursery dataset. As we can see in Figure 40, the third DCCD and MultiScheme have similar performance from drift 0 to drift 5. Even though that DCCD replaced JRip with C4.5 after drift 3 as we mentioned in section 6.3.2. DCCD and MultiScheme continue to have similar performance until drift 5 because the new member did not become the most accurate member until drift 6 which resulted in a slight improvement in its accuracy. Figure 40 shows how DCCD handled the loss in accuracy in drifts 6, 7, 8, and 9.

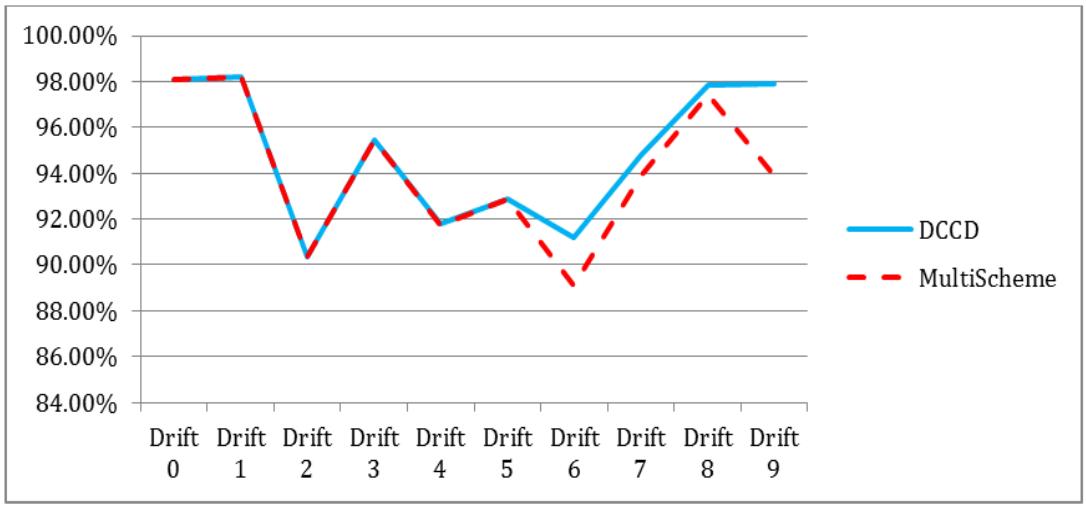

Figure 40: Comparison between the third DCCD and MultiScheme on the Nursery dataset

In Table (c), we present our comparison results between the third committee and MultiScheme on the Contact Lenses datasets. Figure 41 highlights the performance of both which shows that DCCD is always more accurate than MultiScheme. Recall from section 6.3.3, DCCD forces IBK to retire and adds C4.5 instead, which allowed to DCCD to increase its accuracy. Figure 41 shows that the accuracy of DCCD is more accurate in general.

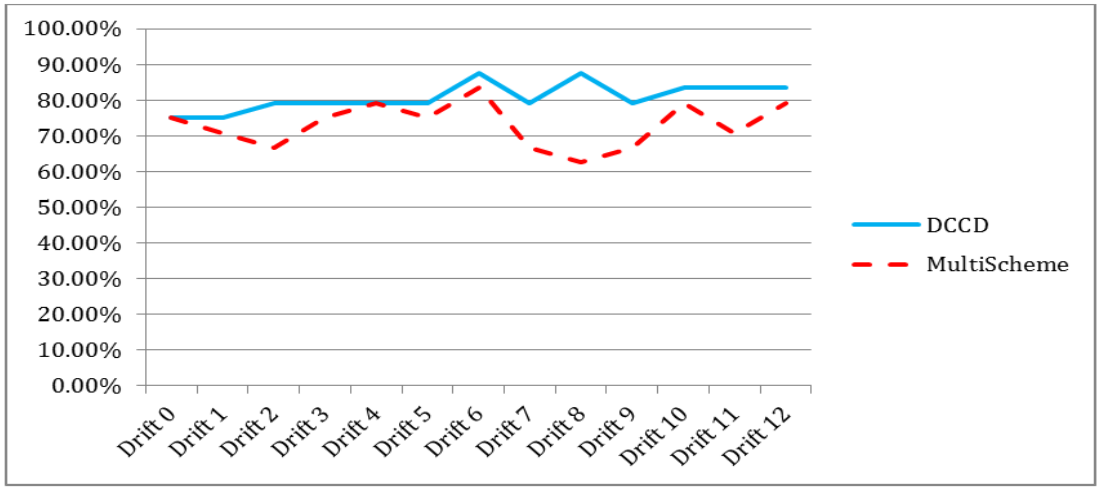

Figure 41: Comparison between the third DCCD and MultiScheme on the Contact Lenses dataset

Finally, we present the comparison results between the third DCCD and MultiScheme on the Iris Plants dataset in Table (d). This table compares the accuracy of DCCD and MultiScheme on sequence of concept drift scenarios that occur on Iris Plants dataset. DCCD did not replace any of its committee members in this committee. As shown in Figure 42, MultiScheme and DCCD have similar performance due to the stability of the dataset.

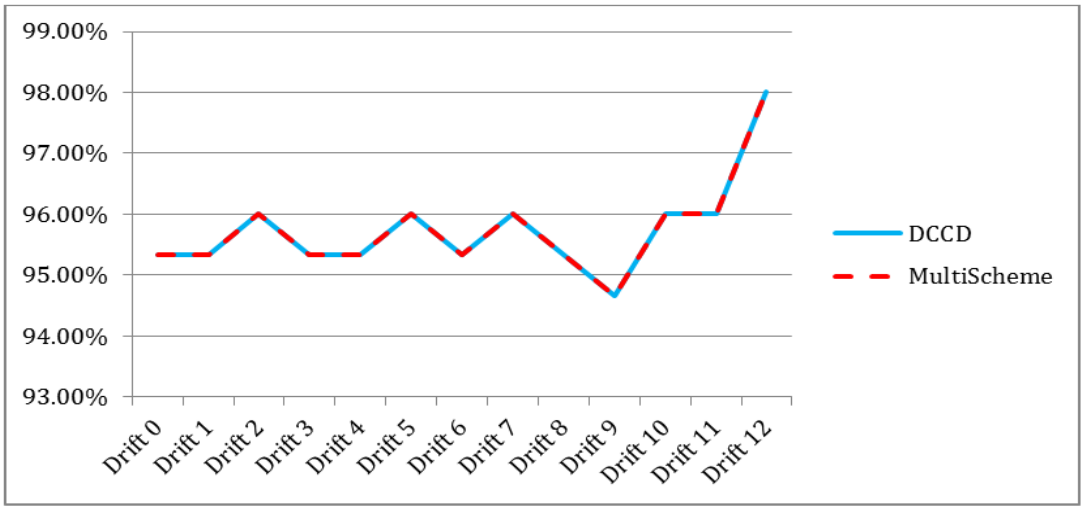

Figure 42: Comparison between the third DCCD and MultiScheme on the Iris Plants dataset

7.6 Evaluation of the Fourth Committee

We compare the fourth DCCD to MultiScheme in terms of the accuracy performance in this section. We present our comparison results between MultiScheme and DCCD by performing them against the four datasets. Table 44 provides the results details of the comparison between the MultiScheme and DCCD as presented in Tables (a), (b), (c), and (d). Notice, both algorithms have the same initial classifiers; C4.5, SMO, and BayesNet.

	DCCD	MultiScheme
Drift 0	93.75%	92.36%
Drift 1	91.44%	89.93%
Drift 2	89.76%	86.86%
Drift 3	90.39%	90.39%
Drift 4	89.41%	87.15%
Drift 5	91.38%	90.39%
Drift 6	81.90%	81.19%
Drift 7	85.01%	84.66%
Drift 8	89.35%	86.92%
Drift 9	84.66%	83.04%
Drift 10	88.37%	86.28%
Drift 11	89.41%	89.01%
Drift 12	85.65%	84.49%
Drift 13	88.19%	89.29%
Drift 14	91.15%	88.31%
Drift 15	86.29%	85.76%

(a) Comparison between the fourth DCCD and MultiScheme on the Car Evaluation dataset

	DCCD	MultiScheme
Drift 0	97.05%	97.05%
Drift 1	97.45 %	97.45 %
Drift 2	90.52 %	90.52 %
Drift 3	95.36%	95.36%
Drift 4	91.78%	91.78%
Drift 5	92.59%	92.59%
Drift 6	91.17%	91.17%
Drift 7	96.17%	94.79%
Drift 8	99.20%	97.49%
Drift 9	96.20%	94.88%

(b) Comparison between the fourth DCCD and MultiScheme on the Nursery dataset

	DCCD	MultiScheme
Drift 0	83.33%	70.83%
Drift 1	83.33%	70.83%
Drift 2	87.50%	70.83%
Drift 3	83.33%	70.83%
Drift 4	83.33%	70.83%
Drift 5	83.33%	70.83%
Drift 6	83.33%	75.00%
Drift 7	87.50 %	70.83%
Drift 8	87.50 %	83.33%
Drift 9	87.50 %	70.83%
Drift 10	87.50 %	70.83%
Drift 11	87.50 %	79.17%
Drift 12	87.50 %	83.33%

(c) Comparison between the fourth DCCD and MultiScheme on the Contact Lenses dataset

	DCCD	MultiScheme
Drift 0	96.00 %	96.00 %
Drift 1	96.67%	95.33%
Drift 2	96.67%	95.33%
Drift 3	96.00 %	95.33%
Drift 4	96.00 %	95.33%
Drift 5	96.00 %	95.33%
Drift 6	96.00 %	95.33%
Drift 7	94.67%	94.67%
Drift 8	96.00 %	94.00%
Drift 9	95.33%	94.67%
Drift 10	95.33%	94.00%
Drift 11	95.33%	94.00%
Drift 12	94.00%	94.00%

(d) Comparison between the fourth DCCD and MultiScheme on the Iris Plants dataset

Table 44: Comparison between the fourth DCCD and MultiScheme

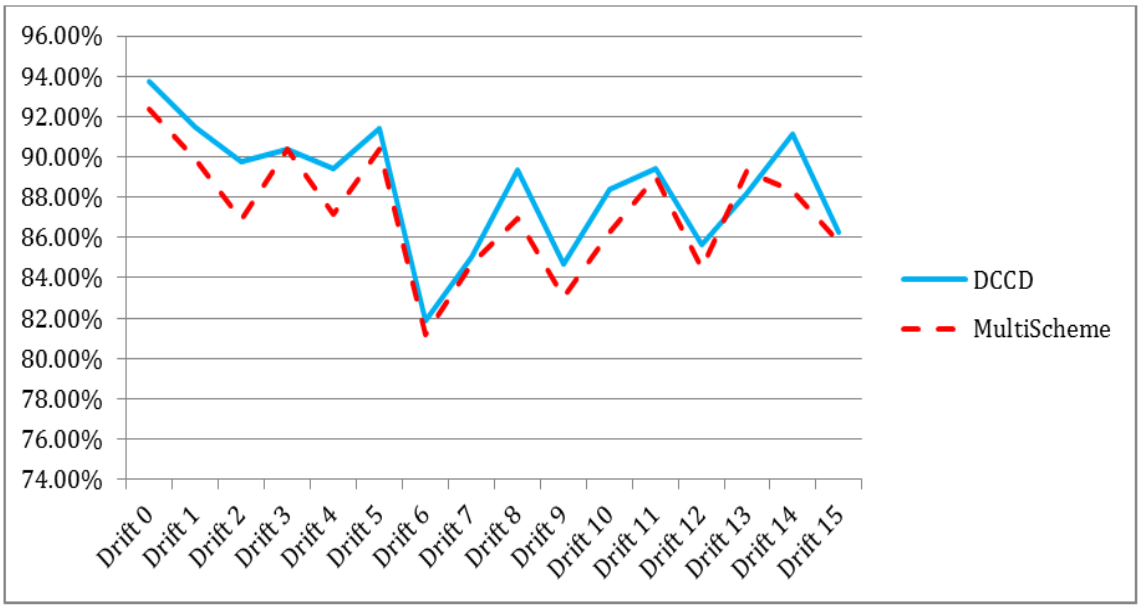

Figure 43: Comparison between the fourth DCCD and MultiScheme on the Car Evaluation dataset

The comparison results between DCCD and MultiScheme on sequence of concept drift scenarios that have been run on the Car Evaluation dataset is presented in Table (a). Figure 43 shows that the accuracy of DCCD is almost always higher than that of MultiScheme.

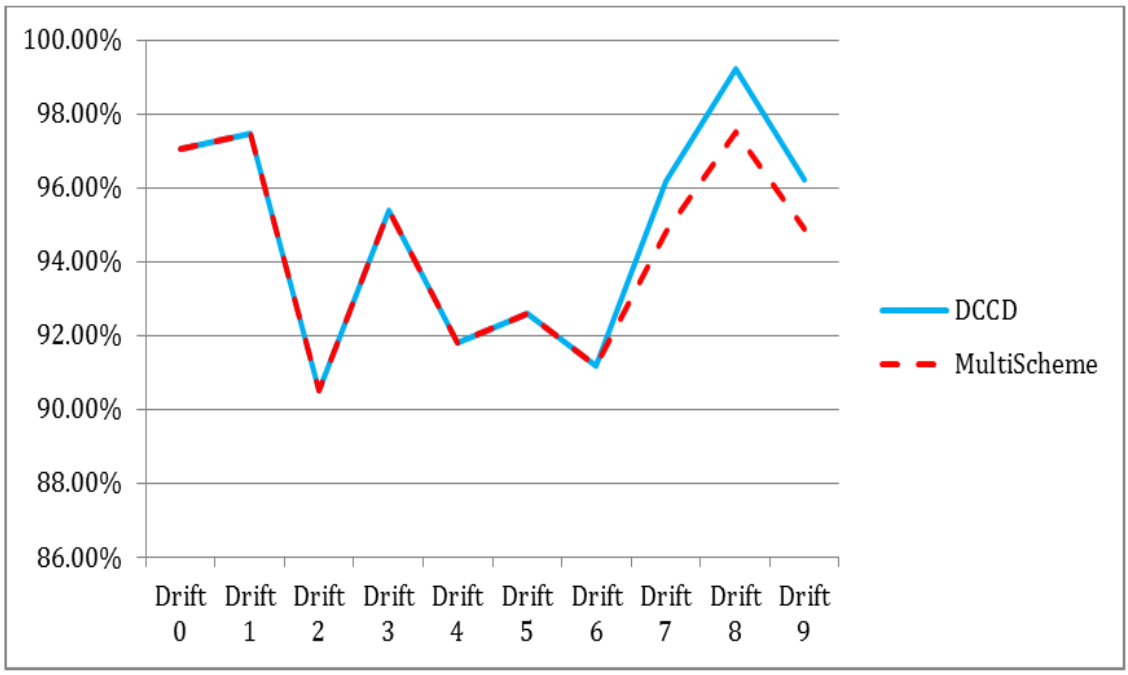

Figure 44: Comparison between the fourth DCCD and MultiScheme on the Nursery dataset

Table (b) donates that DCCD and MultiScheme have similar performance on the Nursery dataset until drift 6. After drift 6, DCCD replaces one of its committee members, which results in increasing the accuracy of DCCD. The difference in the accuracy is highlighted in Figure 44, which compares DCCD and MultiScheme on the sequence of concept drift scenarios that were introduced to this dataset. From drift 0 until drift 6, the performance of DCCD and MultiScheme are identical to each other, but after drift 6 DCCD is almost always more accurate than MultiScheme. Figure 44 shows the increase in the accuracy of DCCD increases after the replacement took place. This significant increase can be used as a balance to the accuracy reduction that concept drifts cause.

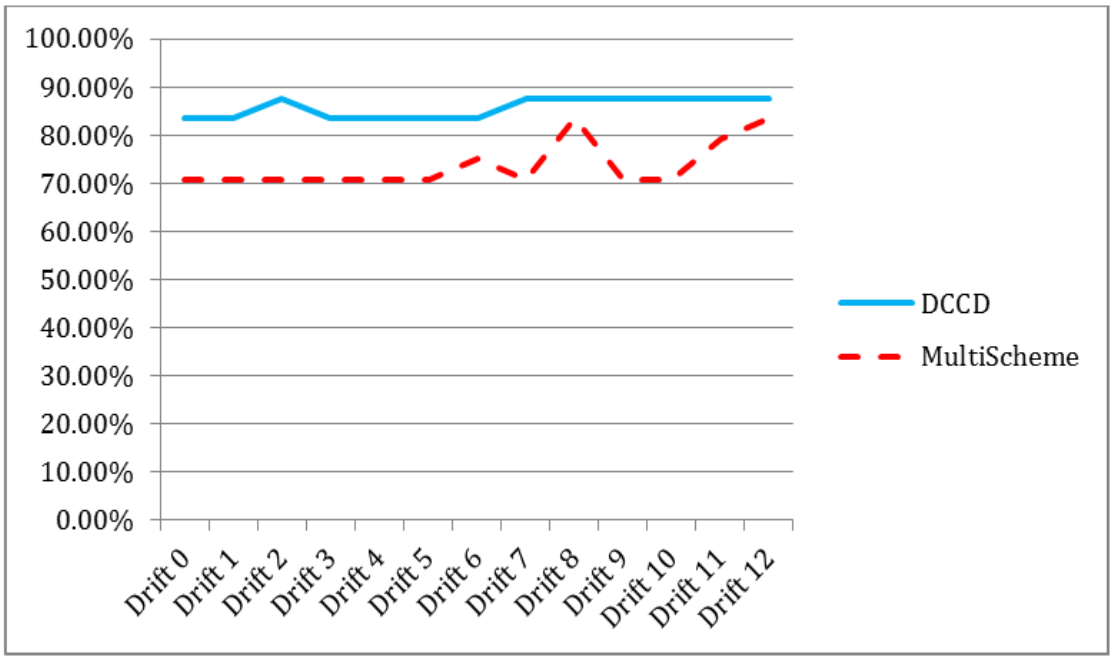

Figure 45: Comparison between the fourth DCCD and MultiScheme on the Contact Lenses dataset

Table (c) donates that the accuracy of DCCD is always higher than MultiScheme the Contact Lenses datasets. After drift 9. DCCD replaced the worst member in the committee with a new member; however, this replacement did not increase the accuracy of DCCD due to the size of the dataset. Based on Table (c), the performance of DCCD is better than MultiScheme even if the replacement did not affect the overall accuracy of DCCD. The difference in the accuracy is exposed in Figure 45, which shows the performance of DCCD and MultiScheme on a sequence of concept drift scenarios that were introduced to this dataset.

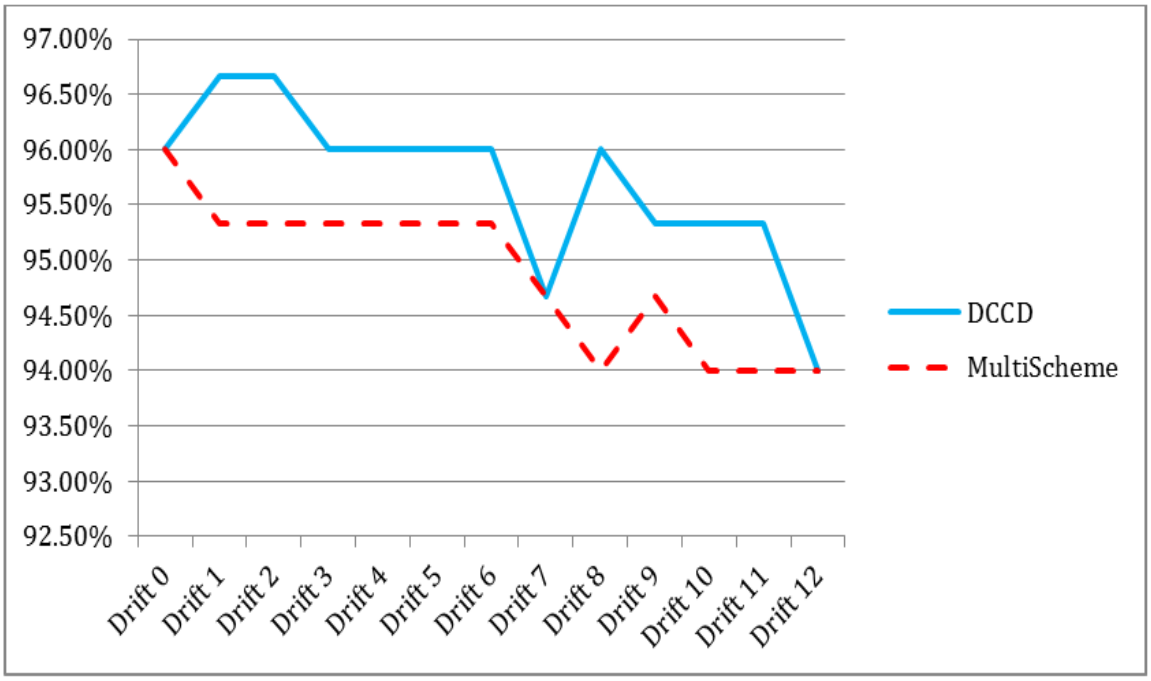

Figure 46: Comparison between the fourth DCCD and MultiScheme on the Iris Plants dataset

We perform the fourth DCCD committee and MultiScheme against the Iris Plants dataset. Table (d) presents the comparison results between of the performance of the fourth DCCD committee and MultiScheme. The difference in the accuracy is highlighted in Figure 46 which shows that DCCD is always more accurate than MultiScheme. Notice that DCCD did not replace any of its committee members as mentioned in section 6.4.4.

7.7 Discussion

In this chapter, we analysed the results of the comparative evaluation between our DCCD algorithm and the MultiScheme algorithm (MS) against the four datasets. The first dataset is the Car Evaluation dataset which is considered as a medium-sized dataset. The Nursery datasets is the second dataset that we used in this study. This dataset is considered as a large-sized dataset. The third dataset is the Contact-Lenses dataset which is a very small dataset. The last dataset that we used is the Iris Plant dataset which is a small dataset. We used diverse committees to measure the performance of DCCD compared to the performance of MultiScheme in terms of accuracy. Table 45 presents the final accuracies of the sixteen experiments that discussed in this chapter. Additionally, this table summarizes the final performance of DCCD and MultiScheme against different sizes of datasets.

		Car Evaluation Dataset (Medium size)		Nursery dataset (Large size)		Contact Lenses dataset (Very small size)		Iris Plant dataset (Small size)	
First Committee		**DCCD**	**MS**	**DCCD**	**MS**	**DCCD**	**MS**	**DCCD**	**MS**
	Final Accuracy	86.28%	84.43%	94.88%	94.88%	87.50%	75.00%	94.00%	94.00 %
	Most Accurate	**DCCD**		**Equal performance**		**DCCD**		**DCCD**	
Second Committee		**DCCD**	**MS**	**DCCD**	**MS**	**DCCD**	**MS**	**DCCD**	**MS**
	Final Accuracy	84.61%	82.23%	96.20%	92.97%	87.50%	79.17%	98.00%	97.33%
	Most Accurate	**DCCD**		**DCCD**		**DCCD**		**DCCD**	
Third Committee		**DCCD**	**MS**	**DCCD**	**MS**	**DCCD**	**MS**	**DCCD**	**MS**
	Final Accuracy	86.29%	82.97%	97.88%	93.86%	83.33%	79.17%	98.00%	98.00%
	Most Accurate	**DCCD**		**DCCD**		**DCCD**		**Equal performance**	
Fourth Committee		**DCCD**	**MS**	**DCCD**	**MS**	**DCCD**	**MS**	**DCCD**	**MS**
	Final Accuracy	86.29%	85.76%	96.20%	94.88%	87.50 %	83.33%	94.00%	94.00%
	Most Accurate	**DCCD**		**DCCD**		**DCCD**		**DCCD**	

Table 45: The final accuracies of the sixteen experiments for DCCD and MultiScheme (MS)

In Table 45, we overview the final accuracies of the sixteen experiments according to the size of our datasets. **For the medium-sized dataset**, DCCD is more accurate than Multischeme in all the committees. Recall from section 6.1.1 that DCCD took advantage of it dynamicity, by replacing the worst member, JRip, in the first committee with SMO. This replacement enabled DCCD to increase its accuracy higher than MultiScheme after drift 9 in this committee as shown in Figure 31. In the second committee, Recall from section 6.2.1 that DCCD replaces the worst member, IBK, with PART which enables DCCD to provide higher accuracy after drift 6 as shows in Figure 35. Moreover, DCCD increases its accuracy higher than MultiScheme after drift 9 in third committee. This increase occurs when DCCD replaces JRip with SMO as mentioned in section 6.3.1 and highlighted the difference in Figure 39. In the fourth committee, DCCD shows more accurate than MultiScheme mainly due to the replacement of C4.5 with PART after drift 6 as highlighted in Figure 43. The results in Table 45, suggests that DCCD provides higher performance more than MultiScheme on the medium-sized dataset.

For the large dataset, DCCD and MultiScheme have a similar performance in the first committee as discussed in section 7.2. Recall from section 6.1.2 that when DCCD replaced JRip with SMO after drift 6; C4.5 continued to be the most accurate member in the committee. Hence,

the advantage of the dynamicity in DCCD did not increase its accuracy. However, DCCD took advantage of its dynamicity in the second, third, and fourth committees which enabled DCCD to provide higher accuracy as shown in Figures 36, 40, and 44. DCCD produces higher accuracy in the large dataset when DCCD uses its dynamicity adequately as emphasized in Figures 36, 40, and 44.

DCCD is more accurate than MultiScheme in **the very small dataset** in all committees. DCCD have replaced the worst member in its committees with a new member in the committees 2, 3, and 4 which reflects in rising the committees' performance as discussed in chapter 6. In contrast, the first committee did not use the dynamicity advantage; however, DCCD maintained higher accuracy more than MultiScheme.

Lastly, DCCD provides higher results in committees 1, 2 and 4, against **the small dataset**. On the other hand, DCCD and MultiScheme have a similar performance in the third committee as shows in Figure 42. Recall from chapter 6 that DCCD did not replace any of its members in all the committees due to the stability of this dataset. Table 45 indicates that DCCD produces higher accuracy when it's performed against the small dataset even though when DCCD does not use its dynamicity advantage. The reason behind DCCD's high results is the training technique. Recall from section 7.1 that MultiScheme trains all the selected classifiers by the same training set and tests them by the testing set. In contrary, DCCD trains and tests each member separately. For instance, suppose MultiScheme trained the selected classifiers against the Iris Plant dataset by using 10 cross validation. MultiScheme divides the dataset into ten equal folds randomly in order to use nine folds for training and one for testing. MultiScheme trains all the selected classifiers by the same nine folds and tests all the selected classifiers by the same one fold. On the other hand, DCCD does not require training and testing its committee members by the same folds.

In this study, eleven experiments took advantage of the dynamicity in DCCD. The comparative evaluation shows that DCCD generally generates higher performance results, especially when the number of concept drifts is large in a dataset. However, the size of the datasets seems to have a significant impact on the accuracy of DCCD. This experiment showed the effectiveness of DCCD in handling gradual and sudden concept drifts. Finally, we can conclude that DCCD has produced highly accurate models on different sizes of datasets that facing gradual and sudden concept drifts. This is our main significant accomplishments in this study.

Table 46 further contrasts DCCD and several algorithms. In this comparison, two features are considered. The first feature is the ability of the algorithms to use more diverse classifiers. The second feature is the ability of the algorithms to replace any of its base classifiers.

	Using diverse base classifiers	**Dynamically** (Adding and removing classifiers)
Adaboost	No	No
Bagging	No	No
Grading	Yes	No
MultiBoostAB	Yes	No
Committee decision tree	No	Yes
MultiScheme	Yes	No
DCCD	Yes	Yes

Table 46: Comparison between DCCD and several algorithms

As Table 46 shows, the only algorithm that has these two features is DCCD. We chose to experimentally compare DCCD to MultiScheme because it has a similar mechanism to evaluate the performance, **as mentioned in section 7.1**, of the used classifiers. In contrast, the other algorithms have very different mechanism to evaluate the performance of their diverse base classifiers. For example, **recall from section 2.3.1**, Grading aims to detect and correct incorrect predictions at the base level by adding into the original dataset a new class labels that encode whether the prediction of the base classifier for each example is correct or incorrect. If we used Grading to deal with concept drifts, the new added class labels into the original dataset to grade each example as correct or incorrect might have legitimate labels at one time and illegitimate labels at another time. Therefore, comparing DCCD to Grading is not reasonable because concept drift will affect its evaluation mechanism.

Additionally, we did not experimentally compare DCCD to MultiBoostAB because the MultiBoostAB's mechanism is based on combining the committee members' outputs to create a single classification from the committee as a whole. **Recall from section 2.3.1,** MultiBoostAB forms and trains a committee to combine the outputs of the committee members in a classification task. MultiBoostAB trains the committee members in the beginning to classify given examples and then MultiBoostAB combines and weights the members' outputs. When a

concept drift occurs, it affects the outputs of the committee members which indeed will affect the combined outputs. For this reason, we avoided comparing DCCD to MultiBoostAB, to ensure the fairness of the experiment.

Recall from chapter 2, Adaboost and Bagging are popular ensemble learning algorithms that can create multiple models of only one base classifier and selects the most accurate model among them. Moreover, Adaboost and Bagging do not have the dynamically feature that makes them able to replace one of created models. Furthermore, Adaboost and Bagging cannot use diverse classifiers at the same time. On the other hand, Grading and MultiBoostAB are ensemble learning algorithms that able to use variety of base classifiers. However, Grading, and MultiBoostAB cannot replace one of its created models without human interaction.

Committee Decision Tree (CDT) is a base learning algorithm that designed to handle concept drift as mentioned in chapter 2. CDT creates multiple Decision Trees and selects the most accurate Decision Trees among them. Moreover, CDT has the dynamically feature that enables it to replace one of created Decision Trees when the accuracy of a model drop below a specific threshold.

In summary, our algorithm DCCD has two features which enables it to create multiple models of one or more base classifier and selects the most accurate model among them. The dynamical nature of DCCD enables it to replace one of base classifiers when one of the base classifiers loss from its accuracy a specific threshold. This feature limits the accuracy reduction that caused by concept drift as we presented in this chapter. Furthermore, this property can reduce the human error in case of selecting inappropriate one of base classifiers for a dataset. Moreover, this characteristic provides one with the flexibility of adding new base classifiers in the future.

7.8 Summary

In this chapter, we presented a comparative evaluation of our approach to that of the MultiScheme algorithm. In this comparative evaluation, DCCD and MultiScheme were run against several datasets. The comparative evaluation confirms that DCCD builds highly accurate models against a number of different sized datasets that have been affected by gradual and sudden concept drifts and that it often outperforms MultiScheme. In chapter 8, we present our conclusion followed by our contributions. Furthermore, we discuss possible future works.

CHAPTER EIGHT
CONCLUSION

CHAPTER EIGHT

CONCLUSION

Concept drift is one of the major issues for the machine learning and data mining society. Regular data mining approaches handle all incoming data as static data represented by a set of features and this, in turn, does not assume the data distribution change that may result from the external environment. A concept may drift due to several motivations such as the changing of the value currency and sensor failure, which can cause serious deterioration of the performance of the current model. Hence, the need of such classifiers that handle concept drift becomes necessary to most users around the world. Extensive work has been conducted on the issue of data mining in an attempt to deal with this complication [7].

Our presented approach, DCCD, reduced the number of lost accuracy, which is needed for any classifier in order to produce good performance results when concept drifts occur. In this thesis, we provided an overview of previous works and research that have been done on the issue of concept drifts. We also provided a detailed discussion of the different types of concept drifts. Furthermore, techniques for committee learning were used in the task of increasing the accuracy of a model and detecting sudden and gradual concept drifts in a dataset. This thesis provided an explanation of several approaches used in the community of data mining that are of interest for our task of handling concept drifts. Furthermore, this thesis provided a detailed explanation of the data preparation and the type of concept drift scenarios that have been applied to each dataset.

We have shown that handling gradual and sudden concept drifts using DCCD requires that the data be prepared first, before DCCD is performed. The steps of preparing the data involve attribute selection and applying the drifts on each dataset. Drifts scenarios will be applied only to attributes that have the largest impact on the entire dataset. DCCD performance is based on two main principles: the average loss in accuracy and the worst member in terms of performance. The detection of concept drift can be evident from the reduction in accuracy that the model loses over time. The worst member performance can be measured by comparing the members' performances in addition to identifying the least accurate member in the committee.

8.1 Thesis Contributions

Handling concept drift using the committee techniques has previously been studied in the research communities [**11**]. There are four main contributions in this thesis. The first contribution is providing a new algorithm called CDI to introduce gradual and sudden concept drifts to datasets. The main advantage of DCI is dividing concept drift into several phases that provide high scalability to concept drift detector by evaluating each phase instead of evaluating each drift scenario.

The second contribution, the creation of an algorithm that can react to the loss in accuracy due to concept drifts. When any type of concept drift appears in a datasets, the used model starts to lose accuracy. Current approaches that handle concept drift focus on how the model should handle the concept drifts or when the model should react to concept drifts. Selecting effective actions alone to deal with concept drifts is not enough. The effective actions must be associated with the perfect timing because focusing on one aspect to resolve this issue and ignoring the other influences the efficiency of the approach. Therefore, DCCD is mainly designed to overcome this gap effectively by selecting appropriate reactions at the appropriate time.

Another contribution of this thesis is providing more member options, which can reduce the amount of inappropriate selection of members that can be made by human mistakes. In other words, suppose a user selected three random members to handle concept drift. However, the selected members are not appropriate for the user dataset. DCCD can overcome this kind of gap effectively by replacing the members, namely the worst members, dynamically. Furthermore, some of the classifiers are the most accurate classifiers in the beginning, but over time this classifier may become less accurate. For example, suppose a user tried several classifiers and

selected the most accurate one among them at a certain time. After the concept drifts occur, the selected classifier loses accuracy and performs worse than it did at the start. DCCD aims to provide insurance to that user by guaranteeing that the user will be able to get the most accurate results every time.

The last contribution of this thesis is the elimination of the worst classifiers by evaluation of their performance in terms of accuracy, and then, replacing them with new classifiers. This procedure can increase the competition between the classifiers, which in turn, will guarantee the users the best results. In other words, suppose a user created a committee consisting of three different classifiers: A, B, and C. Therefore, adding a new member that is as good as B can make the comparison fair. DCCD does this exactly; it provides a fair comparison between the members, which results in an increase of the accuracy.

8.2 Future Work

Our work mainly focused on evaluating how well DCCD performed when gradual and sudden concept drifts appear in datasets. The emphasis was particularly put on "reducing the accuracy loss" in the current model. As far as the author is aware, this is the first study that uses a dynamic committee to handle concept drift, which allows adding and removing diverse base classifiers. The work presented in this thesis may be expanded in many other possible directions. Firstly, DCCD may be extended to consider other concept drift types such as seasonal and re-occurring drifts.

Moreover, it would be possible to adjust DCCD to be able to perform on streaming data rather than just batch data. Another possible direction would be to study the impact of the dataset characteristics on the performance of DCCD. The dataset characteristics could make one classification algorithm more appropriate than another. Several researchers have explored the relationship between the performance of algorithms and the dataset characteristics. However, the impacts of the dataset characteristics on ensemble classifiers need to be examined further. Additionally, it would be worthwhile to study the relationship between the dataset characteristics and concept drift systems.

Another possible direction would be to extend DCCD to work with clustering and compare it to other clustering methods in order to measure its performance. In fact, for our work we only used one measure, which is the accuracy, to evaluate the performance of DCCD. It would be more effective when other measures are included in the criteria for selecting the best

member in a committee. Moreover, it would be interesting to study how other approaches compare when different measures are used in the same datasets and same concept drift types. We considered the accuracy as the main measure due to the complexity of the concept drift issue; it would be more effective to use other measures such as Recall, Precision, and F-measure. Another possible future direction is to run DCCD on more diverse selections of datasets, other than datasets from the UCI repository. For example, an investigation should be considered on the performance of DCCD on synthesis datasets, datasets with noise and outliers, in addition to relational database.

Another suggestion for future work would be to go beyond using base classifiers as members and to allow the members to be ensemble of classifiers. For instance, we can make a committee consisting of Bagging, AdaBoost, and Grading. The main challenge in this particular suggestion is the scalability of DCCD. It would be worthwhile to study how DCCD would perform when given the task of handling concept drifts that occur in a very large dataset. We did not consider incremental base learners such as Hoeffding trees [**91**] or online ensemble approaches such as online Bagging and online Boosting [**37**] in our study. This would be another interesting research direction to pursue.

Bibliography

[1] Cole Burges, "*A Tutorial on Support Vector Machines for Pattern Recognition*" Data Mining and Knowledge Discovery 2.1: 121-168, 1994.

[2] Christopher Bishop, *"Pattern recognition and Machine Learning"*. New York, Springer, 2006.

[3] D. Yeung and P. Chan. (2006). MCS Tutorial [Online]. www.ece.stevens-tech.edu/~hhe/cpe695f09/lecturenotes/

[4] Nils Nilsson. (1996). *Introduction to Machine Learnin: An Early Draft of a Proposed Textbook* [Online]. http://robotics.stanford.edu/people/nilsson/MLBOOK.pdf

[5] Max A. Bramer, "*Principles of data mining"*. Springer. 2007.

[6] Jiawei Han and Micheline Kamber. "*Data Mining : Concept and Techinques*". Morgan Kaufmann, 2006.

[7] M. Kubat G. Widmer, *"Learning in the presence of concept drift and hidden contexts"*. Machine learning 23, 1996.

[8] Sammut C., Horn K. Harries M. *"Extracting hidden context"*. Machine Learning, 32(2), 1998.

[9] Kenneth Stanley, *"Learning concept drift with a committee of decision trees"*. Austin, Texas: Department of Computer Sciences, 2010.

[10] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, *"Handling local concept drift with dynamic integration of classifiers: domain of antibiotic resistance in nosocomial infections"*. In: Proc. 19th IEEE Int. Symposium on Computer-Based Medical Systems CB, 2006.

[11] Indre Zliobaite, *"Learning under Concept Drift: an Overview"*. Lithuania: Faculity of Mayhematics and Informatics.

[12] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen. "*Dynamic integration of classifiers for handling concept drift"*. Information Fusion, 9(1):56-68, 2008.

[13] Tsymbal, A.; Pechenizkiy, M.; Cunningham, P.; Puuronen, S. "*Handling Local Concept Drift with Dynamic Integration of Classifiers: Domain of Antibiotic Resistance in Nosocomial Infections*" Computer-Based Medical Systems, 2006. CBMS 2006. 19th IEEE International Symposium on , vol., no., pp.679-684

[14] L. Breiman, "*Bagging Predictors*," Machine Learning, vol. 24, no. 2, pp. 123-140, 1996.

[15] M. Scholz and R. Klinkenberg, "*Boosting Classifiers for Drifting Concepts*," Intelligent Data Analysis (IDA), Special Issue on Knowledge Discovery from Data Streams, vol. 11, no. 1, pp. 3-28, 2007.

[16] Sašo Džeroski, "*Multi-relational Data Mining: An Introduction*," ACM SIGKDD Explorations Newsletter, 5(1):1–16, 2003.

[17] Feng Muggleton, *Efficient Induction of Logic Program*. Tokyo: Ohmsma, 1993.

[18] Jacek Koronacki. *Advances in machine learning*. Berlin, Germany: Springer Verlag, 2010.

[19] J. C. Schlimmer and R. H. Granger, *"Incremental learning from noisy data,"* Kluwer Academic Publishers, Boston - Manufactured in The Netherlands. 1986.

[20] Alexey Tsymbal, "*The problem of concept drift: definitions and related work*," Technical Report TCD-CS-2004-15. Trinity College Dublin, 2004.

[22] P. Cunningham, A. Tsymbal, and L. Coyle S. Delany, *"A case-based technique for tracking concept drift in spam fltering"*.: Knowledge-Based Systems, 2005.

[23] J. del Campo-Avila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales Bueno M. Baena-Garcia, *"Early drift detection method,"* In ECML PKDD 2006 Workshop on Knowledge Discovery from Data Streams, 2006.

[24] Carsten and Ingrid Renz Lanquillon, *"Adaptive information filtering: detecting changes in text streams"*. Kansas, US: ACM Press, 2000.

[25] Yoav Freund and Robert E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," Journal of Computer and System Sciences, vol. 55(1), pp. 119–139, August 1997.

[26] Len Trigg. (1999). MultiScheme Algorithm [Online] http://www.dbs.ifi.lmu.de/~zimek/diplomathesis/implementations/EHNDs/doc/weka/classifiers/meta/MultiScheme.html

[27] Kim Y. 5. street W., *"A streaming ensemble algorithm (SEA) for large-scale classification,"* In Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining KDD-2001, ACM Press, 2001.

[28] J. Zico Kolter and Marcus A. Maloof, "*Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts,"* Journal of Machine Learning Research 8, pp. 2755-2790, 2007.

[29] Kubat M. Widmer G., *"Effective learning in dynamic environments by explicit context tracking"*.: Proc. 6th European Conf. on Machine Learning ECML-1993, Springer-Verlag, 1993.

[30] Hulten G., Spencer L., and Domingos P., "*Mining time-changing data streams,*" in *Proc.* 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining KDD-2001, ACM Press, pp. 97-106, 2001.

[31] J. Zico Kolter and Marcus A. Maloof. *Dynamic weighted majority: A new ensemble method for tracking concept drift*. In Proc. of the 3rd IEEE International Conference on Data Mining, pp. 123-

130, 2003.

[32] Ralf Klinkenberg, *"Learning drifting concepts: example selection vs. example weighting,"*: Intelligent Data Analysis, Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift, 8 (3), 2004.

[34] Widmer G. Kubat M., *"Adapting to drift in continuous domains"*. Vienna: Tech. Report ÖFAI-TR-94- 27, Austrian Research Institute for Artificial Intelligence, 1994.

[35] K. Yeon, M. S. Song, Y. Kim, H. Choi, and C. Park. *Model averaging via penalized regression for tracking concept drift*. Journal of Computational and Graphical Statistics, 19(2):457-473, 2010.

[36] Marcos Salganicoff, "Tolerating concept and sampling shift in lazy learning using prediction error context switching". AI Review, Special Issue on Lazy Learning, 11 (1-5), pp. 133-155 , 1997.

[37] Oza, N.C.; "*Online bagging and boosting,"* Systems, Man and Cybernetics, 2005 IEEE International Conference on , vol.3, no., pp. 2340- 2345 Vol. 3, 10-12 Oct. 2005

[38] Nowlan N., Delany S.J., Haahr M. Cunningham P., *"A case-based approach to spam filtering that can track concept drift,"*. Proc. ICCBR-2003 Workshop on Long-Lived CBR Systems, 2003.

[39] Jose-Federico Ramirez-Cruz, Olac Fuentes, Vicente Alarcon-Aquino, and Luciano Garcia-Banuelos, *"Instance Selection and Feature Weighting Using Evolutionary Algorithms"*. Proceedings of the 15th International Conference on Computing (CIC'06), 2006.

[40] Ivan Koychev, *"Gradual forgetting for adaptation to concept drift* In Proceedings of ECAI 2000 Workshop Current Issues in Spatio-Temporal Reasoning. Pp. 101-106. 2000

[41] B. John Oommen, Luis Rueda, "*Stochastic learning-based weak estimation of multinomial random variables and its applications to pattern recognition in non-stationary environments*". Pattern Recogn., 39(3):328-341. March 2006

[42] Fang Chu and Carlo Zaniolo, "*Fast and light boosting for adaptive mining of data streams,*" PAKDD 2004, LNAI 3056, pp. 282–292, Springer-Verlag Berlin Heidelberg. 2004.

[43] Mak, L.-O.; Krause, P. ,"*Detection & Management of Concept Drift,*" Machine Learning and Cybernetics, 2006 International Conference on , vol., no., pp.3486-3491, 13-16 Aug. 2006

[44] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavaldà, *"New ensemble methods for evolving data streams"*. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09). Pp. 139-148, ACM, New York, NY, USA. 2009.

[45] Jing Gao; Ding, B.; Wei Fan; Jiawei Han; Yu, P.S.; "*Classifying Data Streams with Skewed Class Distributions and Concept Drifts,*" Internet Computing, IEEE , vol.12, no.6, pp.37-49, Nov.-Dec. 2008

[46] Y. Freund and R. Schapire, *"A short introduction to boosting"*. Journal of Japanese Society for

Artificial Intelligence, 14(5):771-780, September, 1999.

[47] Freund, Y. and Schapire, R., "*Experiments with a New Boosting Algorithm*". Machine Learning: Proceedings of the Thirteenth International Conference, 148–156. 1996.

[48] J. Jiang and C. Zhai, "*Instance Weighting for Domain Adaptation in NLP*," Proc. 45th Ann. Meeting of the Assoc. Computational Linguistics, pp. 264-271, June 2007.

[49] GEOFFREY I. WEBB. "*MultiBoosting: A technique for combining boosting and wagging*". Machine Learning Kluwer Academic Publishers, Boston. Manufactured in The Netherlands 40(2):159–196. 2000

[50] Peng Zhang, Xingquan Zhu, and Yong Shi. *"Categorizing and mining concept drifting data streams"*: In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '08). ACM, New York, NY, USA, 812-820. 2008.

[51] Ross Quinlan, *"C4.5: programs for machine learning"*. Burlington: Morgan Kaufmann, 1993.

[52] Haibo He. (2009) "*Applied Machine Learning*" Stevens Institute of Technology. [Online]. Available: http://www.ece.stevens-tech.edu/~hhe/cpe695f09/lecturenotes/Lecture3.pdf

[53] Ross Quinlan, "*Induction of Decision Trees*": Mach. Learn. , 81-106., (Mar. 1986).

[54] Divine Muhivuwomunda, Data De-Duplication through Active Learning. Ottawa, Ontario : MCS Thesis submitted to University of Ottawa, 2010.

[55] Ian Witten and Eibe Frank, *"Data Mining: Practical machine learning tools and techniques with java implementations"*. San Francisco: Morgan Kaufmann, 2000.

[56] John Cleary and Leinard Trigg, *"K*: An Instance-based Learner Using an Entropic Distance Measure*," New Zealand: Dept. of Computer Science, University of Waikato.

[57] Rivest and Ronald Cormen, *"Introduction to Algorithms"*. Cambridge: MIT Press, 1990.

[58] Robert Gray, *"Entropy and Information Theory"*. New York: NY: Springer-Verlag, 1990.

[59] Paulos, John Allen, "*The Mathematics of Changing Your Mind,*" New York Times (US) , August 5 2011.

[60] Remco Bouckaert, *"Bayesian network classifiers in WEKA"*. United States: University of Waikato, , 1999.

[61] Remco R. Bouckaert, *"Bayesian Network Classifiers in WEKA for Version 3-5-7"*. May 12, 2008.

[62] Maya R. Gupta and Yihua Chen, "*Theory and use of the EM algorithm*," Foundations and Trends in Signal Processing, vol. 4, no. 3, pp. 223-296, 2010.

[63] Blum, A. and Mitchell, T. , "*Combining labeled and unlabeled data with co-training,*": In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT '98)*, pp.

92–100. 1998.

[64] Jensen Finn and Thomas Graven, *"Bayesian Networks and Decision Graphs"*. New York: NY: Springer, 2007.

[65] Alexander K. Seewald and Johannes. “An Evaluation of Grading Classifiers”. In *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis* (IDA '01). Springer-Verlag, London, UK, UK, 115-124.

[66] J. R. Quinlan and R. M. Cameron Jones, "FOIL," *A midterm report In Pavel B Brazdil, editor, Machine Learning ECML, Vienna, Austria, Springer Verlag*, vol. Lecture notes in Computer Science #667, 1993.

[67] William W. Cohen, "*Fast Effective Rule Induction*,": In Proceedings of the Twelfth International Conference on Machine Learning, 1995.

[68] Riel Arthur, *"Object oriented Design Heuristics"*. Addison Wesley, 1996.

[69] Frank, and Witten, Ian. Eibe, *"Generating accurate rules sets without global optimization"*. New Zealand: Hamilton, 1997.

[70] Bremner D, Demaine E, Erickson J, Iacono J, Langerman S, Morin P, Toussaint G , “*Output-sensitive algorithms for computing nearest-neighbor decision boundaries”*: Discrete and Computational Geometry 33 (4): 593–604., 2005.

[71] Kardi Teknomo. (2006) “*K Nearest Neighbors Tutorial*” [Online] Available: http://people.revoledu.com/kardi/tutorial/KNN/index.html

[72] Daniel Larose, *“k-nearest neighbor algorithm”*. Discovering Knowledge in Data: An introduction to Data Mining. John Wiley & Sons, 2005.

[73] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, California, USA, second edition, 2005.

[74] Richard Duda and Hart Peter, *"Pattern classification" 2nd ed*.: David Stork, 2000.

[75] John C. Platt, "*Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*," Microsoft Research , Technical Report MSR-TR-98-14, 1998.

[76] Nocedal, Jorge; Wright, Stephen J., *Numerical Optimization* , 2nd ed. Berlin, New York, US: Springer-Verlag, 2006.

[77] Schölkopf, B., and Smola, A.J. , *Learning with kernels: support vector machines, regularization, optimization, and beyond*.: Cambridge, Massachusetts: MIT Press, 2002.

[78] Berciano, A. and Diaz-Pernil, D. and Kropatsch, W. and Molina-Abril, H. and Real, P, "*Computer Analysis of Images and Patterns*," in 14th International Conference, Caip 2011, Seville, Spain Proceedings. New York: Springer. 2011.

[79] S Abe, "*Support Vector Machines for Pattern Classification,*" London: Springer Publishers , 2010.

[80] Hamel, L, *Knowledge discovery with support vector machines*. Hoboken, New Jersey: John Wiley and Sons, 2009.

[81] Ronald Winder, *"Threshold Logic."*. PhD Dissertation. Princeton University: Princeton press, 1962.

[82] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg., "Top 10 algorithms in data mining," *Knowl. Inf. Syst. 14, 1*, no. 1-37. December 2007.

[83] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); *The WEKA Data Mining Software: An Update*; SIGKDD Explorations, Volume 11, Issue 1.

[84] Frank, A. & Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[85] Petri Myllymäki, *"Advantages of Bayesian Networks in Data Mining and Knowledge Discovery"*.: Academy Research Ph.D,Fellow, Complex Systems Computation Group, Helsinki Institute for Information Technology.

[86] Uusitalo L, *"Advantages and challenges of Bayesian networks in environmental modeling"*. Ecological Modelling, 203, pp.312-318, (2007).

[87] D.N. Barton, T. Saloranta, S.J. Moe, H.O. Eggestad, S. Kuikka, "*Bayesian belief networks as a meta-modelling tool in integrated river basin management - Pros and cons in evaluating nutrient abatement decisions under uncertainty in a Norwegian river basin, Ecological Economics"*, Volume 66, Issue 1, Pages 91-104, 2008.

[88] Padraig Cunningham and Sarah Jane Delany, *"k-Nearest Neighbour Classifiers"*: Technical Report UCD-CSI-2007-4, March 27, 2007.

[89] Michael Kearns. (1988) *"Thoughts on hypothesis boosting"*: Unpublished manuscript [Online] available: http://www.citeulike.org/user/seb1/article/5980850

[90] R. Elwell and R. Polikar, "*Incremental Learning of Concept Drift in Nonstationary Environments*," In Neural Networks, IEEE Transactions. pp.1517-1531, Oct. 2011.

[91] Pedro Domingos and Geoff Hulten. "*Mining high-speed data streams*." In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '00). ACM, New York, NY, USA, pp. 71-80. 2000.

[92] Md. M. Islam, X. Yao, and K. Murase. A constructive algorithm for training cooperative neural network ensembles. IEEE Transactions on Neural Networks, 14(4):820-834, July 2003.