QATAR UNIVERSITY

COLLEGE OF ENGINEERING

BAGGED RANDOMIZED CONCEPTUAL MACHINE LEARNING METHOD

BY

MOHAMED ABDALHAKEM TAHA ALI

A Thesis Submitted to

the Faculty of the College of

Engineering

in Partial Fulfillment

of the Requirements

for the Degree of

Masters of Science in Computing

June 2018

COMMITTEE PAGE

The members of the Committee approve the Thesis of Mohamed Abdalhakem Taha Ali defended on 13/05/2018.

Dr. Ali Jaoua
Thesis/Dissertation Supervisor

Dr. Somaya Al-Maadeed
Thesis/Dissertation Supervisor

Dr. Engelbert Mephu Nguifo
Committee Member

Dr. Tamer Elsayed
Committee Member

Dr. Abdelkarim Erradi
Committee Chair

Approved:

Khalifa Al-Khalifa, Dean, College of Engineering

ABSTRACT

ALI, MOHAMED, A., Masters: June: 2018, Masters of Science in Computing

June : 2018, Masters of Science in Computing

Title: Bagged Randomized Conceptual Machine Learning Method

Supervisors of Thesis: Ali, M, Jaoua and Somaya, A, Al-Maadeed.

Formal concept analysis (FCA) is a scientific approach aiming to investigate, analyze and represent the conceptual knowledge deduced from the data in conceptual structures (lattice). Recently many researchers are counting on the potentials of FCA to resolve or contribute addressing machine learning problems. However, some of these heuristics are still far from achieving this goal. In another context, ensemble-learning methods are deemed effective in addressing the classification problem, in addition, introducing randomness to ensemble learning found effective in certain scenarios. We exploit the potentials of FCA and the notion of randomness in ensemble learning, and propose a new machine learning method based on random conceptual decomposition. We also propose a novel approach for rule optimization. We develop an effective learning algorithm that is capable of handling some of learning problem aspects, with results that are comparable to other ensemble learning algorithms.

DEDICATION

I dedicate this thesis to my family who stayed beside me in every stage of my life surrounding me with love, to all of my friends who supports me and encouraged me to continue, and to my supervisors and teachers for always being supportive during my master's thesis.

ACKNOWLEDGMENTS

First and last we thank Almighty God for his grace. Second, I would like to express my deepest gratitude to my supervisors Dr. Ali Jaoua and Dr. Somaya Al-Maadeed for their support, guidance and patience. And I would like to express my gratitude to my family and friends who supports me in this challenge. I am eternally grateful to my instructors for the quality of knowledge I got from them during my study in Qatar University.

This contribution was made possible by NPRP grant #07- 794-1-145 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOTATIONS

$\mathbb{E}(.)$	Mathematical expectation of possible values of (.) – Summation or integration of the expected values.
$\mathbb{I}(.)$	Indicator function, returns 1 if (.) is true, and 0 when (.) is false.
$h(.)$	Classifier (hypothesis).
$f(.)$	Target function, returns the correct value of variable (.).
$H(.)$	Set of Classifiers (hypotheses)
$err(.)$	Error function returns the error caused by (.).
$\arg max\,(.)$	The argument of maxima, return the maximum value of (.).
$P(.)$	Probability of (.).
γ, λ	Galois Operators, used for obtaining Galois closure.
$Occurence(.)$	Returns the instances where (.) is observed.
$\|.\|$	Cardinality, measures the number of elements in (.)

ABBREVIATIONS

FCA	Formal Concept Analysis.
RCL	Random Conceptual Learner.
$B\text{-}RCL$	Bagged Random Conceptual Learner.
BFC	Boosting Formal Concepts.
BNC	Boosting of Nominal Concepts.
DNC	Dagging Nominal Concepts.
CNC	Classifier Nominal Concept
$Bagging$	Bootstrap Aggregating.
$RMCS$	Recommender-based Multiple Classifier System.
$Dagging$	Disjoints samples Aggregating.
$Wagging$	Weight Aggregation.
$Vogging$	Variance Optimized Bagging.
RF	Random Forests.
$SPFC$	Selecting Plausible Formal Concepts.
IRP	Incremental Rule Production.
$FIPR$	Fuzzy Incremental Production Rule.
$SAMME$	Stagewise Additive Modeling using a Multi-class Exponential loss function.

CHAPTER 1: INTRODUCTION

This thesis studies a new conceptual learning paradigm—an innovative machine learning approach for handling the learning problem based on Formal Concept Analysis, random conceptual coverage, and ensemble learning. In this chapter, we introduce supervised learning by taking a glance at its importance and applications, and by introducing Formal Concept Analysis. Through these introductions, we will start to understand our motivations behind using this approach. In addition, we will elaborate on the main problem in machine learning and define some of the popular learning problem approximations. Finally, we will examine the research questions and discuss our main objectives.

1.1 Supervised Machine Learning

Supervised learning is an active research area as its importance is reflected in several fields. As is well known, it plays an important role in many computing practices such as spam filtering[1], text, speech recognition, and image processing[2], as well as many applications that have a direct impact on people's lives, such as crime data analysis, medical research[3] and much more. Supervised learning is usually a discrete process; it may process continuous data such as streaming data learning[4]. Machine learning algorithms can be defined as computational structures that learn from given data and apply the knowledge deduced from the data to decision-making[5]. Supervised learning algorithms learn from given training examples that fall in specific categories. Generally, supervised learning is oriented toward discovering the hidden patterns that construct these categories or classes, in order to use these patterns to classify future data. All the existing learning algorithms are proposed as an approximation to one problem with different aspects, and this problem is called the learning problem [5]. However, have these algorithms succeeded? The answer is not that simple. It is not easy to judge the performance of a classification model, and it is not possible to point to a certain method that solves the learning problem completely. Some tradeoffs need to be done in order to achieve an acceptable resolution. In order to create a successful learning algorithm, there are many aspects to consider, such as the learning techniques and tools, classification method, the proper ways to handle the data, and tradeoffs. In the present, there are many supervised learning algorithms; some are successfully implemented and used for different applications. This is due to their abilities to handle classification tasks properly. In fact, most of the existing learning algorithms are built on top of statistical approaches, such as decision and naïve Bayesian; some others may use different tools.

In our scenario, we are going to build our learning system by utilizing a mathematical approach known as Formal Concept Analysis (FCA). FCA is based on set theory and discrete mathematics as well as exploitation of some existing statistical methods. We will use this approach to achieve our goal, which will be explained later in more details. In the beginning, we are going to discuss the learning problem. Then, we are going to show our motivations behind adopting Formal Concept Analysis for this task, and explore why we think it is suitable for this mission.

1.2 Problem Description

In general, any classification model is considered acceptable if it satisfies the user's requirements. These requirements may vary from one user to another, while each user has his or her own measures to determine the quality of the produced classification results. However, it is common to evaluate a classification model based on the performance of that model. Typically, it is necessary for a classification model to achieve a level of generalization, so later; it could properly perform the classification process of the unseen data. A machine-learning model is the process of generalizing the information deduced from the training examples and intersecting that information with data in the unseen records. A general model is more likely to perform the classification task accurately, with a minimal prediction error rate[6], [7]. Since the quality of a model relies on how general it is, we might say a good classifier is a model that has a low generalization error rate, meaning most of the predicted classes are the correct classes of the given records. The calculation of the generalization error relies on how it will perform with the future data; however, it is not possible because the process is discrete. Many approximations have been used to predict the model's abilities to perform in the future, more specifically to validate the learning potentials and to verify the adequacy of the training data. In the next subsections, we will focus on the formulation of the learning problem; briefly discuss various approximations for addressing the learning problem, and tradeoffs to make.

1.2.1 Learning Problem

As mentioned in the previous section, the learning problem[5] is represented as the learner's ability to produce a classification model that is able to predict the future data correctly (i.e. a model with a low generalization error). The process of learning is formulated as follows: for a given training set $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, the instances x_i drawn independently and identically distributed from D, where D is the distribution over instance space X. We have $f(x_i) = y_i$, where f a target function that produces the correct class of instance x_i. The aim is to form a hypothesis h that minimizes the generalization error, given by:

$$err(h) = \mathbb{E}_{x\sim D}[\mathbb{I}(h(x) \neq f(x))] \tag{1.1}$$

The notation $\mathbb{E}_{x\sim D}$ is the Mathematical Expectation (summation, or integration) of the outputs of $\mathbb{I}(.)$, the later is an indicator function; returns 1 if (.) is true, and 0 when (.) is false. However, the generalization error is composed of three kinds of errors: (i) error because of bias, (ii) error because of variance, (iii) irreducible error or noise. Bias error occurs because of the learning algorithm's inability to build a strong hypothesis that can be used to classify the future data; such an error occurs because of weaknesses in the learning algorithm itself (see figure 1). In contrast, an error caused by variance reflects the ability of the learning algorithm to learn efficiently from the existing data; however, the produced model may fail at estimating the classes of future data. This is usually caused by the lack of representative examples in the complex learning algorithm. The third error category is caused by noise, which is probably out of hand and irreducible. However, to overcome the error, we should sacrifice some bias and vice versa. This tradeoff will be discussed in the next subsection.

1.2.2 Bias-Variance Tradeoff

To understand the Bias-Variance tradeoff [8], let us assume that we have training data with overlapped data points, and we consider that we have only two classes (O, X) that represent the whole feature space. Fitting a linear model on this data may result in poor performance, due to the overlap between the data points, where some of data points that belong to class O share some features with data points that belong to the other class,

X. The model will include many data points that belong to the other class. This kind of error is caused by a biased hypothesis that is not flexible enough to separate between the data points and results in poor performance or under-fitting.

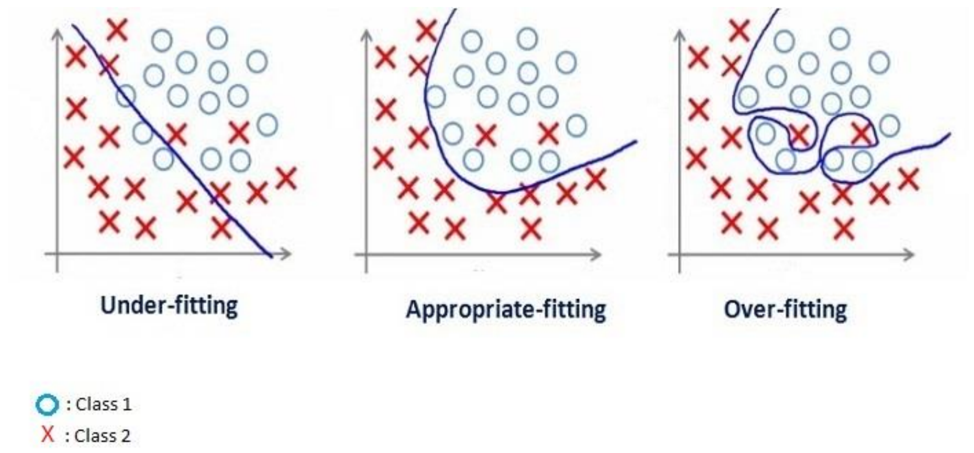


Figure 1. Different versions of models fitting

On the other hand, if we assumed that we were able to find a flexible hypothesis that fit tightly on the training data, we may face another problem when it comes to classifying unseen records with a different distribution over space, because the model is tightly fitted on the training examples and unable to generalize. Figure 1 shows the differences between model over-fitting and under-fitting. As we can see, the decision boundary is precise in an over-fitted module. It only describes the properties of the training examples. It will probably fail to predict unseen records with different distributions. However, using a linear model may result in high bias that leads to a high error rate. In order to design a learning algorithm, we should tradeoff some bias for the sake of variance (see figure 2).

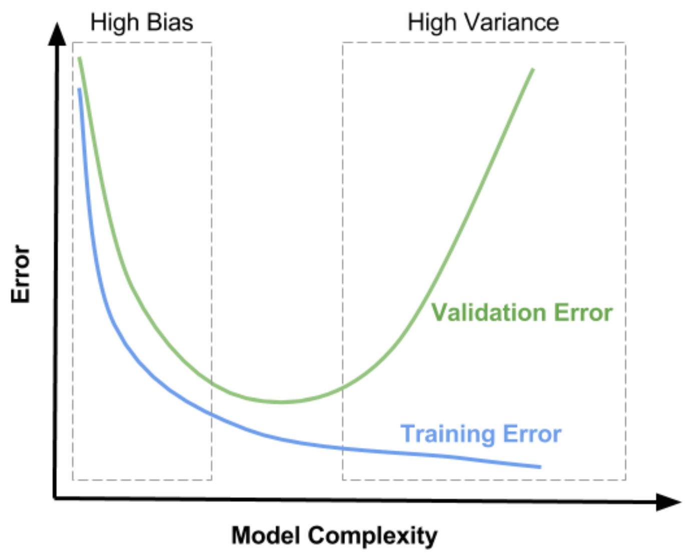


Figure 2. Bias-variance tradeoff

The goal is to achieve a level in which the model is able to keep both the variance and bias as low as possible. Many techniques have been created to handle some aspects of the learning problem; we are going to introduce some of them in the next section.

1.2.3 Problem Approximation Techniques

In literature, we can find many approaches that minimize the effects of the learning problem, concerning predicting performance. Some techniques are designed to average the variance, such as data partitioning methods, sampling, random subspace, and pruning[6]. Others work to produce unbiased hypotheses, such as boosting. Variance reduction techniques tend to hide some of the data from the learner, to avoid over-fitting or revising the produced hypotheses, such as rules/decision tree pruning. Other methods aim to average the variance while keeping the bias as low as possible, not by hiding the data, but by building multiple models on top of different subsets sampled from the data. In the presence of such a large number of approximations, it becomes difficult to decide what kind of practice is more suitable to adapt. So far, the empirical results favor the ensemble learning approaches. Before discussing ensemble learning, we will cover our perception of how to utilize Formal Concept Analysis in supervised learning as well as the main research questions deduced from the problem, and our motivations for using FCA.

1.3 Research Questions and Motivations

Based on the stated problem, the main question of the thesis is: how is it possible to create an effective learning procedure using a randomized rule extraction method based on Formal Concept Analysis? To answer this question, we need to answer two sub-questions:

1. How we can utilize FCA to minimize the learning bias?
2. Is it possible to produce models with minimal variance relying on FCA alone, and if not, what is the best practice to use along with FCA to achieve that goal?

In this thesis, we aim to implement a FCA-based learning method. Therefore, we will focus on the best practices and the strategies in the literature that we can be exploited to achieve our goals. Generally, FCA is used in many computing-related fields such as feature extraction, categorization, data reduction, human language processing, and knowledge discovery. In addition, FCA is used in different machine learning practices, including ensemble learning, with promising outcomes[2], [9]–[12]. Creating an effective FCA-based learning system means that we should develop learning methods with acceptable results in terms of classification performance while taking into account the other design considerations concerning the learning problem.

1.4 Purpose of the Study

The goal of this thesis is to design, implement, and evaluate a FCA-based learning system that contributes effectively to the learning problem. To achieve this goal, we will utilize the recent advances and the best practices in machine learning and Formal Concept Analysis to satisfy the following objectives:

- Design an FCA-based learning algorithm with minimal possible bias.
- Adapt a suitable ensemble-learning paradigm to minimize the variance aspect.
- Utilize random conceptual coverage as a practice for lowering model variance.
- Implement the design and assess the quality and performance of the model by comparing the model with other existing methods, using known benchmark databases.
- The new approach should be effective, with acceptable and comparable performance.

1.5 Contribution of the Thesis

In this thesis, we show the effect of using ensemble machine learning alongside Formal Concept Analysis in order to contribute in solving the learning problem. The principal contributions of this thesis are:

- A new concepts decomposition method based on random coverage; this method is introduced to use within ensemble learning environment to enforce the diversity between the classifiers.
- A novel rule enhancement method to reduce the bias of the conceptual classification rules and to improve the overall performance of the system.
- To contribute to the learning problem of variance we exploit Bagging ensemble learning paradigm alongside our proposed conceptual learner in order to achieve better performance.

1.6 Summary and Thesis Structure

In this chapter, we give a glance at supervised learning, its importance, and its applications. We introduce FCA and its contributions to the supervised learning area in section 1.1, and in in section 1.2 we define the learning problem how affects the classification modeling. In that section, we also mention in this chapter various approximations to the learning problem, but we focus on ensemble learning, as it serves the goal of this thesis. We follow that explanation with the research questions and discuss the main questions to be answered in this study in section 1.3 and in section 1.4, we state our objectives based on the research questions, and the learning problem that we want to tackle.

In the next chapter, we discuss different ensemble learning and their main properties. In chapter 3 we will introduce Formal Concept Analysis and we will focus on its application in machine learning, specifically in ensemble learning. In chapter 4 we will propose our method and we will explain how it works. In chapter 5 we will experiment our method and show our results, and finally conclude and discuss the future work.

CHAPTER 2: ENSEMBLE LEARNING

In this chapter, we review ensemble learning as an approximation to the learning problem. First, we discuss the reasons behind considering the ensemble method for learning. Then, we discuss the common ensemble learning approaches and methods used for achieving diversity and the techniques used for aggregation in ensemble learning.

2.1 Why Ensemble Learning?

In supervised learning, the objective is to build a strong model h that minimizes the generalization error $err(h)$, which can be decomposed to variance/bias errors. Ensemble learning was introduced to achieve this goal. We can define ensemble learning as a set of finite models$\{h_1, h_2, \dots, h_t\}$ referred to as base classifiers, produced from a homogenous or heterogeneous learning process and aggregated together using a combination method to form a strong classifier $H(x)$. It is expected to outperform single models if it achieves a level of diversity between the base classifiers[13]. $H(x)$is a diverse ensemble classifier if the base classifiers produce uncorrelated errors. The diversity of an ensemble classifier is reflected in the performance of that model. Many studies show that it is more likely to reduce the bias of a learning system by combining multiple weak learners in a dependent sequential manner; other proposed methods work to reduce the variance while maintaining the bias as low as possible by aggregating the decisions made by multiple independent models[6].

In the next section, we will review the common ensemble methods and their characteristics and limitations.

2.2 Ensemble Methods

Different taxonomies are reported in the literature for classifying ensemble learning methods. Some are based on ensemble fusion topologies [14]; others are based on their learning functionalities [6]. In this chapter, we will review different ensemble learning methods and discuss both their contributions to the learning problem and their common implementations.

2.2.1 Sequential Learning Paradigm

In sequential ensemble learning, the learning process is performed in a cascading manner. The learning algorithm will be executed N times, to learn from the training data. Each learning phase depends on the output of the previous phase (see figure 3). Each phase consists of a validation process to identify learning errors. Sequential methods generate a weighted classification model in each running phase. The output of each learning phase is considered weighted training data; the weights are distributed based on the validation errors; such that higher weights are assigned to the instances that are misclassified. Generally, the aim of such procedures is to boost the performance of a weak learning algorithm and reduce the bias by correcting the errors made in the prior execution.

The first sequential ensemble procedure was proposed by Schapire[15].The idea behind this method is to boost the learning ability of a weak learner through a cascading process to enhance classification decisions of the produced model. The author's argument is built on the notion of "weak and strong learners". Weak learners produce high training errors, and consequently generate biased models with performances that are only slightly better than random guessing, while strong learners produce models with very low bias.

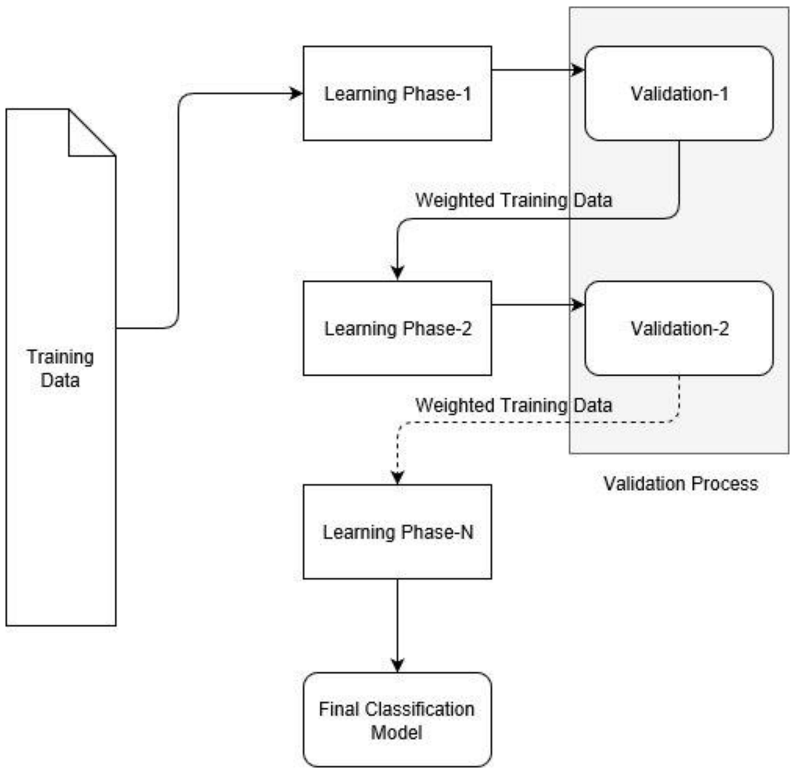


Figure 3. General sequential ensemble learning process

Adaptive Boosting, or AdaBoost, is a binary ensemble learning family, introduced by Freund and Schapire[16] based on boosting. AdaBoost learner will start by assigning equal weights in the training examples, which indicates the importance of each record. By increasing the weights on the misclassified examples in the validation process, the algorithm raises their priorities, so instances with higher weights get more attention in the next iteration. At the base level, it requires a learning algorithm that is able to handle the weights properly, and it is possible to use any learning algorithm that is able to perform this task. The AdaBoost algorithm controls the processes of distribution and weighting. The objective of this approach, as in the case of all sequential learning ensembles, is to reduce the bias produced by a weak learning algorithm. However, AdaBoost shows an increased over-fitting behavior [17] as long as the optimization process is ongoing. Over-fitting can be controlled by limiting the number of learning iterations over the training data.

Another implementation of adaptive boosting, called boosting with re-sampling [16], [18], does not require a base learning algorithm with the ability to handle weighted instances. In this case, the weights are handled through a sampling process; the ensemble learner draws N samples from the training data with replacements; the probability of selecting each instance is proportional to the weight of that instance. Two advantages can be drawn from this approach:

1. The first advantage is that the ensemble learner is no longer limited to base learners that can handle the weights.
2. The second advantage is that the ensemble shows some resistance to over-fitting through the re-sampling process [19].

Generally, AdaBoost generates M classification models along with M learning iterations performed; each model is weighted based on the errors produced by that model in the validation process. In the classification phase, the final decisions are based on weighted majority voting criteria that involves all the generated models through the learning process. In literature, boosting ensembles, underwent several optimizations over time, including resample boosting. Other extensions have been added to the algorithm to enhance the predictive abilities, such as the multi-class extension by Hastie[20], which allows AdaBoost to act on multi-class problems. Gradient boosting [21] is another popular variety of the boosting algorithm; it shares the same concept with AdaBoost, however, gradient boosting does not reweight the training instances. Instead, the next weak learner will learn from the remaining errors (pseudo-residuals) of the previous learners. In addition, the weighting criteria for the generated model in each phase is constructed through a gradient descent optimization process.

Finally, we can conclude that most of the sequential learning ensembles are designed using the same concepts and to serve the same goal, which is to reduce the bias produced by weak learning algorithms. However, models that are produced using these procedures are sensitive to variance, especially when the number of learning iterations is very large. In the next section, we are going to review the second ensemble-learning paradigm and discuss the main objectives and characteristics of the parallel ensembles.

2.2.2 Parallel Learning Paradigm

In the previous section, we discussed the sequential ensemble methods as an approximation to highly biased learners. In contrast, the parallel ensemble methods objective is to reduce the variance of unstable learning algorithms. We can define the model stability referring to the tradeoff discussed earlier in Chapter 1. Unstable learning algorithms [21] are the algorithms that use complex hypotheses to learn from the training data, and they produce a very low training bias. Unstable learners are very sensitive to changes in the data; any small change in the training data will produce different classification hypotheses. In contrast, small changes in the training data will not affect stable learners. Unstable classifiers tend to over-fit the training data, and produce classification models with high error rates when it comes to classifying unseen or future data.

In the parallel paradigm, the learning algorithm will be executed multiple times in an independent fashion. Each iteration is not dependent on the previous one, and in each learning phase, the learner will work on different samples of the training data. This method aims to achieve some normalcy between variance and bias, as the traditional approaches, such as data partitioning or early stopping, will limit the learner's ability to learn from the data, which will increase the bias produced by the learning algorithm. The theory behind this approach is to enable the unstable learner to learn at a high capacity from multiple samples taken from the training data; these samples represent the scope of the information in the training data, but from different perspectives. Generally, parallel ensemble learning approaches creates S samples from the training data, and release the base unstable learning algorithm to learn from all the data generated independently; an independent classification model will result from each iteration (see figure 4).

The first ensemble approach utilized for the parallel learning paradigm is Bagging (**B**ootstrap **agg**rega**ting**) predictors [22]**.** The idea behind bagging is to generate multiple bootstrap samples from the training data; those samples are generated randomly with replacements. Each of the generated samples is used to generate a separate classification model. Later, in the classification phase, all the classifiers will vote, and the most voted class will be used as a final decision. Bootstrap sampling ensures significant variety among the examples selected for each sample, and this leads to the production of classification models with lower correlation.

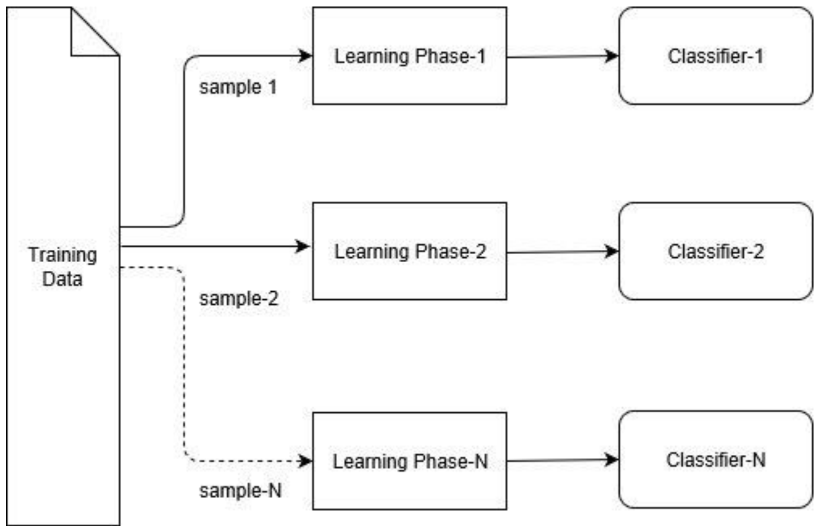


Figure 4. General parallel ensemble learning structure

Bagging achieves diversity and independence if it is used with unstable base learners, such as decision tree; if base classifiers built using stable base learner, the generated models will be quite similar, they will produce the same decisions repeatedly, and no improvements will be achieved, in terms of reducing the variance of the model. Disjoint sample aggregating (Dagging) is similar to bagging except that it uses stratified sampling rather than bootstrapping [23]. Other variants of bagging are wagging [24] and vogging[25]. Wagging uses random weighting distribution to omit some instances from the training data on each run; here, instances with weights equal to zero will be neglected. However, wagging requires a base learner with the ability to handle the weights distribution. On the other hand, vogging (variance optimized bagging) introduced to further reduce the variance of the classification models using Markowitz Mean-Variance Portfolio to produce low variance model while retaining the bias at the same level.

Decision Forests[26] is another parallel learning approach, which build multiple decision trees by dividing the features space into random subsets rather than sampling the data as in bagging.

Random forests is another popular variation of parallel ensemble learning introduced by Breiman[7]**.** A collection of CART decision tree base classifiers will be constructed using bootstrap sampling and a high level of variety is enforced between the trees by adding another level of randomization on the base learner, by utilizing the notion of random subspace [26]. The main difference between traditional bagging and random forests is that bagging uses all the features at each node to split while the random forests algorithm utilizes random sub features, where it selects $m < M$ random features among the best M features to split the decision tree nodes. The aim of such levels of randomization is to reduce the correlations between produced classifiers, which empirically reduce the variance of the resulted model. Similar to bagging, random forests adapt majority voting for selecting the right class for a specific object. Breiman introduced several methods for random features selection. Forest-RI was used for splitting by randomly selecting one variable from the best possible options to split. In another method, the selection was determined by generating linear combinations of inputs and the number of combinations was predefined. The default selection method for random forests is to select $m < M$ features from each instance, where the size of m is equal to the square root of the total number of features in each instance[7]. Moreover, random forests introduced a new error estimation metric known as the out-of-bag error estimator, which relies on the training instances that are not selected while constructing a certain base classifier. Rotation Forests algorithm[27]**,** proposed to enhance the accuracy and diversity of base classifiers. It carries similar concepts exists in random forests while claiming to introduce an enhanced model of randomization using rotation matrix and Principal Component Analysis (PCA) for feature extraction.

Generally, parallel ensembles are designed to exploit the learning abilities of unstable learners to aggregate multiple classification models and enhance the diversity among them. The classification models produced using parallel ensemble learning preserve their predication abilities and show a resistance to variance, especially when the number of learning iterations is increased. In the next section, we are going to review the third ensemble-learning paradigm, and we will discuss the main objectives and characteristics of the stacking ensembles.

2.2.3 Stacked Generalization Learning Paradigm

Stacking ensembles [28] consist of two levels of learners: level-1 learners, and a combiner or “meta-learner”. Similar to bagging and boosting, each of the level-1 learners is trained using a different set of sampled/same data; then the outputs from the first layer are used as inputs for the lower-level meta-learner. Stacking ensembles usually consist of “heterogeneous” sets of level-1 learners. Unlike boosting and bagging, it can be considered as a hybrid approach; in the upper level, the learners are trained independently on parallel; then the meta-learner uses the outputs produced by the validation process to learn about the generated models itself. The meta-learner uses the outputs from each classification model to learn which model is better than the others for classifying certain instances; figure 5 shows the learning phase in stacking models. This is achieved by creating a new training set, which contains information about how each one of the produced classifiers performs better in certain instances to utilize the confidences of each decision. As in other ensemble techniques, stacking aims to improve the accuracy classification, and it achieves a higher level of accuracy than individual classifiers**.**

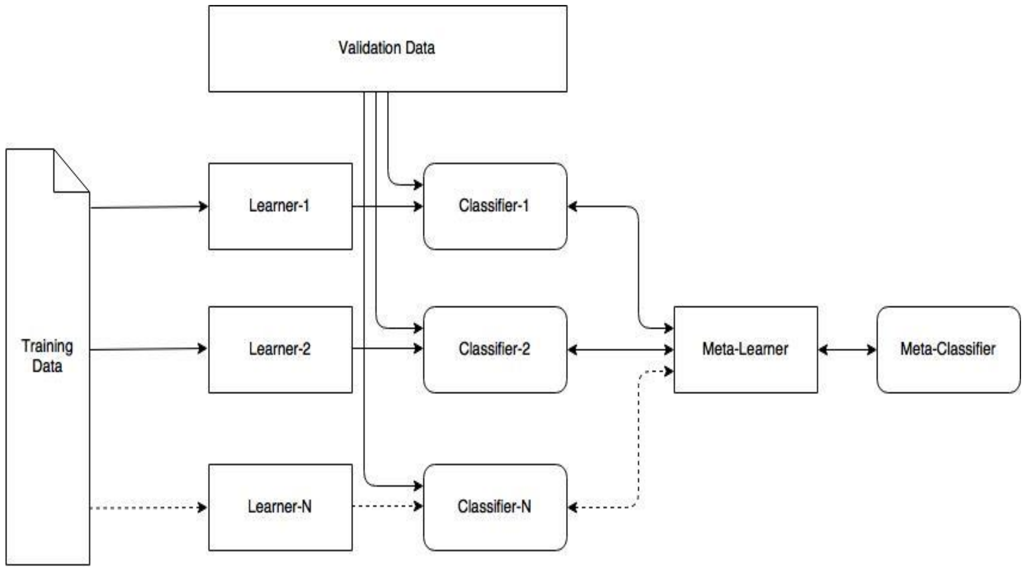


*Figure 5***.** Stacked Generalization learning process

Variant learning methods that are constructed with the same concepts of stacking are reported in various pieces of literature. Because of the generality of stacking, these methods may vary from each other based on different factors. Such as the types of upper-level learners and meta-learners [29], evaluation methods [30], meta-data set construction methods, and the types of metadata [31], [32]. Generally, stacking ensembles are prone to the biases of the selected learner; therefore, to achieve the best possible performance, different considerations should be taken into account, such as performance of upper-level and meta-level learners, the number of upper-level learners, and the metadata generation approach [13]. Stacking is more efficient in terms of variance reduction, and this is because stacking ideally works with different samples of the training data by using multiple heterogeneous learning algorithms.

2.3 Distribution Methods

In this section, we will review the common methods that are used to create diversity among the base classifiers in ensemble learning; we will discuss the main characteristics of these methods and their use.

2.3.1 Weights Distribution

The weighting approach is commonly used in popular sequential ensembles, such as boosting learners, and is rarely used in parallel ensembles. Generally, the weights are used to reflect the importance and difficulty of the training instances; this will force the learning process to concentrate on highly weighted instances. For example, AdaBoost algorithm will set the new weight D_{new} for training example $x$in the training set using the following equation:

$$D_{new}(x) = \frac{D_{old}(x)}{Z} \times \begin{cases} e^{-\alpha} & if\ h(x) = f(x) \\ e^{\alpha} & if\ h(x) \neq f(x) \end{cases} \tag{2.1}$$

where α is the weight of the classifier $h(x)$and indicates the performance of the model on the training data (training error), and $f(x)$ is targets function that yields the correct class value, and Z is a normalization factor (for example sum of all weights). What can be deduced from this equation (2.1) is that the weight D indicates whether the training example x will be in the spotlight in the next learning phase or not. The exponential function returns a fraction value if x is correctly classified, and a value of 1 or greater if x is misclassified. Using these weights of that particular sample can be increased or decreased when multiplying it with the previous weight. The resulted classification models have some level of diversity due to the different types of training instances they deal with.

2.3.2 Random Subspace

The random space method [33] indicates that the learner will work on a random subset of the features each time it iterates over the training data, to emphasize the diversity among the produced models. There are different approaches to sampling the features space, such as random sampling, where the learning algorithms use different numbers of features each time, while running consistently. Some other approaches sample the features space with a restrictive manner, where the number of random features used in every execution is static [7]. Unlike sampling, the feature subspace approach projects only parts of attribute values used to train certain models.

2.3.2 Probability Sampling

Various sampling techniques are adopted for ensemble learning, including bootstrap sampling[7], sampling without replacement [27], and stratified sampling [23], where each has its own characteristics. Bootstrap sampling draws random samples from the original population with size equal to the original population, with the probability of repetition (i.e. each element has a chance to be taken again and added to the current sample), where each element has a probability of not being selected equal to:

$$\left(1-\frac{1}{N}\right)^{N} \tag{2.2}$$

while in sampling without replacement, each element has a chance to be selected only once. Therefore, the sample size n should be maintained as $n<N$, where N is the size of the original population. Otherwise, the sample will be exactly the same as the original set. On the other hand, stratified sampling splits the population into disjoint groups (strata); in supervised learning, these groups could be the class values. Sampling is done with respect to strata size i.e. from each stratum k samples will be drawn based on the following procedure:

$$k = S \times \frac{n}{N} \tag{2.3}$$

where n is the number of elements to be sampled randomly, and S is the size of the stratum; and N is the total number of elements in the population.

Sampling in ensemble learning methods is used to average the variance in classification by generating more datasets from the training set, by using random sampling with replacement. In some cases, sampling is used to produce multiple training sets of the same size as the original dataset, and in other scenarios, the repetition will only occur across multiple training sets and not within the same training data. However, increasing the number of the training sets will not reduce the bias or increase the model's prediction ability; it will average the variance, by using a combination method, such as majority vote of the outcomes from each classification model, leading to a reduction in the errors caused by the variance. Consequently, to obtain the best performance, we should have a strong learner at the base level.

2.4 Combination Methods

The combination methods and functions play an important role in terms of classification performance; however, this role depends heavily on the diversity of the base classifiers. Combining identical classifiers will not boost the performance of the ensemble, but rather produce the same performance as a single one. Therefore, combining multiple classifiers, each trained with uncorrelated and reweighted samples of the data, will improve the prediction performance. The sequential and parallel ensemble methods rely on un trainable combiners (majority vote, weighted majority vote) which produce their decisions by averaging the upper layer predictions. This is not the case in stacking, where the combiner learns from these predictions.

2.4.1 Majority Vote

Majority vote is a popular combination method; specifically in parallel learning, as in[7], [22]. Let us assume we have a set of finite models$\{h_1, h_2, \dots, h_t\}$, the output of each model is given by $h_t(x) = \mathrm{y} \in Y,$ the possible number of class values. And $\mathbb{I}(.)$ is an indication function that returns $\{0,1\}$ if the condition is satisfied or not.

$$H(x) = arg \max_{y \in Y} \sum_{i=1}^{t} \mathbb{I}(h_i(x) = \mathrm{y}) \tag{2.4}$$

The function $arg \max_{y \in Y}$ will return the value of y that has the maximum number of occurrences because of the classification hypothesis $h_i(x)$.

2.4.2 Weighted Majority Vote

In another variation of majority vote, each classification model is assigned a weighting factor that generally represents the model accuracy [15], [16], [20]. With the same assumption made for the majority vote in (2.4); by introducing $w_i$as the weighting coefficient for the classification model h_i, the weighted majority vote is given by:

$$H(x) = arg \max_{y \in Y} \sum_{i=1}^{t} w_i(h_i(x) = \mathrm{y}) \tag{2.5}$$

2.4.3 Bayesian Probability

Bayesian Combination is based on Bayesian rule; in ensemble learning, the final decision is based on the class value that maximizes the probability function:

$$P(y, h(x)) \tag{2.6}$$

$P(y, h(x))$ is the probability of instance x to be in class $y \in Y$. The Bayesian rule is used to approximate the probability function by:

$$P(y, h(x)) = \frac{P(y, h(x))}{\sum_{i=1}^{t} P(y_i, h(x))} \tag{2.7}$$

In ensemble learning, the probability of class value y is counted across multiple classification models:

$$H(x) = arg \max_{y \in Y} \sum_{i=1}^{t} P(y, h_i(x)) \tag{2.8}$$

Bayesian combination might be altered based on the implementation requirement for a certain ensemble-learning model. Different ensemble learning methods adopt the

Bayesian combination method, such as decision forests [33], and rotation forests [27].

2.4.4 Meta-Combiners

Meta-Combiners are trainable combination methods, which are widely implemented in stacking approaches. The meta-learner selects the best decision based on a certain hypothesis deduced from the upper-level classifiers. Table 1 is an example of what a meta-learner dataset may look like.

Table 1

Example of Meta-combiner dataset

	$Classifier_1$	$Classifier_2$	$Classifier_n$	Y
1	1	1	0	0
2	1	0	1	1
3	0	1	1	1
4	0	0	1	0

2.5 Chapter Summary

In this chapter, we discussed different ensemble learning approaches that contribute to the learning problem. We reviewed the reasons behind using ensemble learning as an approximation to the learning problem and that ensemble learning is superior among the other approaches. We reviewed different ensemble implementations, their characteristics, and how they treat the bias-variance tradeoff. We discussed how each learning paradigm is used to achieve some levels of diversity among the base classifiers, and their limitations in terms of the types and numbers of base learners that can be used. Moreover, we reviewed some of the common combination methods used in ensemble learning. Based on the objectives of this thesis, we are going to introduce an FCA learning approach to ensemble learning. Therefore, in the following chapter, we will review Formal Concept Analysis, and the learning approaches that are based on FCA.

Table-2 summarizes different characteristics of the reviewed ensemble learning approaches. These characteristics are the focus area, which represents the method's main contribution to the learning problem. The learning paradigm defines how the learning process is carried on. The Base learner is the suitable learning algorithm to be used with a certain ensemble method. Learning method indicates whether the method involves a single base learning algorithm, or a set of different learning algorithms. Moreover, the diversity generation method defines the approaches used to achieve diversity between the produced classifiers, and finally the combination method used to combine the predictions from the classifiers in the ensemble in order to achieve the final decision.

Table 2

A comparison of different reviewed ensemble approaches

Ensemble Method	Focus Area	Learning Paradigm	Base Learner	Learning Method	Diversity Generation Method	Combination Method
Decision Forests	Variance	Independent/parallel	Decision Tree	Homogenous	Features Subspace	Bayesian Probability
Bagging Predictors	Variance	Independent/parallel	Any Unstable Learner	Homogenous	Bootstrap Sampling	Majority Vote
Dagging	Variance	Independent/parallel	Any Unstable Learner	Homogenous	Stratified Sampling	Majority Vote
Wagging	Variance	Independent/parallel	Weights handling unstable base learner	Homogenous	Weights Distribution	Majority Vote
Vogging	Variance	Independent/parallel	Any Unstable Learner	Homogenous	Bootstrap Sampling	Sharpe Ratio
Random Forests	Variance	Independent/parallel	Decision Tree	Homogenous	Bootstrap Sampling & Features Subspace	Majority Vote
Rotation Forests	Variance	Independent/parallel	Decision Tree	Homogenous	Bootstrap Sampling & Features Subspace	Weighted Majority Vote
Adaboost	Bias	Dependent/sequential	Weights handling Weak base learner	Homogenous	Weights Distribution	Weighted Majority Vote
Resampling Adaboost	Bias	Dependent/sequential	Weak Base learner	Homogenous	Weighted Samples	Weighted Majority Vote
Gradient Boosting	Bias	Dependent/sequential	Weak Base learner	Homogenous	Pseudo-residuals	Weighted Majority Vote
Stacking	Variance	Hybrid	Any	Heterogeneous	Sampling	Meta-Combiner

CHAPTER 3: FORMAL CONCEPT ANALYSIS IN MACHINE LEARNING

In the previous section, we covered a general overview of supervised learning. In this section, we define the Formal Concept Analysis and its applications in machine learning, focusing in ensemble methods carried on using FCA.

3.1 Why FCA?

Formal Concept Analysis [34]is a scientific approach and a discipline that aims to investigate and analyze the data, and represent and manage the knowledge extracted from the data in conceptual structures known as lattices. An FCA lattice helps to visualize these concepts, enabling better methods to interpret and understand these concepts, and also find patterns and regularities. In this manner, FCA has been adopted in many computing-related practices, such as features extraction[35]–[37], categorization, data reduction, linguistics, knowledge discovery[38], and machine learning. Recently, many researchers have counted on the potentials of FCA to resolve or address many computational problems, mainly for its efficiency and capability to accommodate and address these problems.

Recently, many FCA research projects were carried out to address various machine-learning aspects. Before we move on to them, we formally introduce and define Formal Concept Analysis.

3.2 Definitions

In this section, we will formally define Formal Context and Formal Concept, the constituent parts of the approach.

Definition 1 (Formal Context)

Let us say we have $O = \{o_1, o_2, \ldots\ldots o_n\}$, which is a finite set of objects, we have $A = \{a_1, a_2, \ldots\ldots a_n\}$ a finite set of attributes, and R is a binary relation between $O\ and\ A$, where$R \subseteq O \times A$, (O, A, R) represents a formal context (see Table-3).

Let us have $X \subseteq O$,$\gamma(X) := \{a \in A\ |\forall o \in X: (o, a) \in R\}$, and also $Y \subseteq A$, $\lambda(Y) := \{o \in O\ |\forall a \in Y: (o, a) \in R\}$. $\gamma(X)$ is defined as a set of all attributes of objects in X, while $\lambda(Y)$is the set of all objects sharing all the attributes in Y. Operators γ and λ define the Galois connection between sets X and Y. The closure operators are γoλ, and λoγ.

Table 3

Formal Context Example

$O - \mathrm{A}$	a_1	a_2	a_3	a_4
o_1	1	1	0	0
o_2	1	0	1	0
o_3	0	1	0	1
o_4	0	0	1	1

Definition 2 (Formal Concept)

A pair $(X, Y)$is a formal concept of (O, A, R) if and only if $X \subseteq O, Y \subseteq A$, $\gamma(X) = Y\ and\ \lambda(Y) = X$. Set X is called the extent (domain) of the formal concept, and set Y is the intent of the formal concept (co-domain). All formal concepts $\{(X_1, Y_y), (X_2, Y_2), \dots., (X_n, Y_n)\}$ extracted from the formal context(O, A, R) can be organized in conceptual lattices (see figure 6). The formal concepts of a given context are naturally ordered by the sub-concept; the super-concept relation is defined by:$(X_1, Y_1) \leq (X_2, Y_2)$ if and only if $(X_1 \subseteq X_2)$ and $(Y_2 \subseteq Y_1)$.

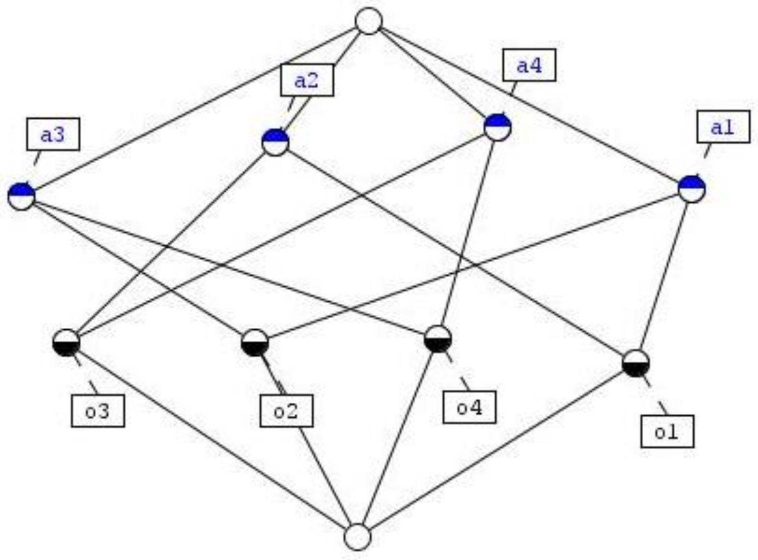


Figure 6. Conceptual Lattice of the formal context in table 3

However, representing a large number of concepts is not effective in terms of space complexity; thus, the ideal solution is to represent the whole space using a minimum number of concepts. In addition, obtaining optimal concepts is challenging as it is considered an NP-Hard problem[39], but on the other hand, many heuristic contributions were introduced, providing approximate solutions in order to tackle this problem, and many of these heuristics have been deemed useful in data analysis and reduction. In the next section, we are going to review some of the heuristics that aim to decompose optimal formal concepts from contexts.

3.3 FCA Conceptual Decomposition Methods

In FCA, we can use the notion of concepts to obtain rule-based classification systems. However, generating concepts from a context is fully dependent on how we decompose the context to find the optimal points for conceptual coverage. In addition, the method used to obtain the concepts has an impact on the complexity of the system. Many heuristics have been introduced to obtain the optimal concepts from a given context. The fringes decomposition [40], [41] approach is used to obtain a set of isolated points which belong to only one concept in the binary context; the decomposition method is performed by applying Riguet's difunctional relation recursively in the context, and the resulting points will be used to construct a set of formal concepts. Another approach used is Shannon's Entropy, to decompose the context and obtain the minimal formal concepts that describe a given context [10], [11], [42]. Moreover, some approaches perform the coverage based on the knowledge gained from the produced concepts; such approaches tend to evaluate the produced concepts using certain quality measures, such as confidence and correlation [43]. Furthermore, lazy-coverage [44] is a method that aims to obtain only desirable concepts from a context; this approach is usually practical in classification.

The quality of the decomposed concepts affects the overall system performance; therefore, it is important to obtain the best and minimal concepts that describe the whole context. In the next section, we are going to review FCA practices in machine learning. We will discuss the common FCA-based learning algorithms and the variation of ensemble learning methods that utilize FCA-based learning algorithms.

3.4 FCA-Based Learning

There are several FCA-based algorithms for supervised learning, based on Trabelsi taxonomy [45] of the existing supervised classification methods. FCA-based classification methods can be divided into two categories: exhaustive and combinatory. The first category involves the ensemble learning algorithms that are based on FCA. The second category contains the learning methods that exploit the ensemble learning paradigms.

Some of the exhaustive learning methods are construct and use the conceptual lattices, such as, GALOIS and Selecting Plausible Formal Concepts(SPFC)[45]. The limitations of these approaches are time and resource complexities, because the construction of the lattice is an exponential process. Other approaches, like Fuzzy Incremental Production rule (FIPR)[46] and Incremental Rule Production (IRP)[47], are constructed using only part of the lattice. Several FCA learning methods extract pertinent concepts, such as IPR, however, these methods require some quality metrics to evaluate or control the concept extraction processes [9]. There are many characteristics for each; complete lattice-based classifiers are usually prone to exponential complexity in terms of time and resources, but conversely, they are more precise in terms of extracted knowledge. Sub lattice-based methods only use part of the conceptual lattice, which can reduce the complexity and operate more efficiently for resource consumption. It also reduces the redundancy in the generated concepts, keeping only the most relevant options among them. However, using only sub lattices may cause information loss affecting the overall accuracy of the algorithm.

Combinatory approaches, or FCA learners that utilize ensemble learning, were introduced to improve the overall performance of learning. Boosting formal concepts (BFC)[11] and Boosting nominal classifier (BNC)[10] are considered sequential ensemble classifiers; the base learners in BFC and BNC use the basis of Formal Concept Analysis to extract classification rules from the training set. BNC is similar to BFC; the only difference is that BNC was built to handle nominal data, mainly to avoid the complexity that follows transforming the data to the binary representation. The two algorithms cover only parts of the conceptual lattice, which is relevant to the samples drawn from the training set. Both BFC and BNC are based on AdaBoost.M2 ensemble learning paradigm.

BFC operates by forming classification rules based on the discovered concepts. The processes are divided in two phases, the learning phase and the classification phase. In the learning phase, the algorithm extracts concepts from the data based on conceptual decomposition. In the classification phase, a class is assigned to the unseen objects based on the extracted concepts from the data. The learning phase is done in sequential process, learning one classifier at a time based on a subset of the training set. The authors in [11] claim that boosting a weak classifier allows improvement on overall performance; in addition, it improves the error rate, and reduces over-fitting. Moreover, the decisions obtained using multiple classifiers are more reliable than the ones produced using a single classifier. Given any training set, the algorithm will assign equal weights to all training samples, and then, it will randomly draw samples from the training set and build the first classifier using this sample. Moving forward, it will repeat this process T times on the data set. In each iteration, the algorithm will modify the weights and construct new classifiers based on the new weighting criteria.

The classifier in this case is a classification rule extracted from a formal concept. In each iteration, the algorithm works on a different subset of the data (based on the weights). It will extract formal concepts by selecting the attribute (ex. binary attribute) that satisfies specific criteria (ex. minimize Shannon entropy). In a case where multiple attributes have the same entropy value, the algorithm will select the one that has more support. Next, after selecting the attribute, the algorithm will obtain the formal concept by applying Galois closure on the attribute. After that, the classification rule is obtained by the intent of the concept, and the class is obtained by the majority vote associated with the extent of the concept. Then, the rule is returned to the main algorithm AdaBoost.M2, which calculates Pseudo-loss on the classifier, and uses the Pseudo-loss to calculate the error and update the weights of the training examples according to the error and a normalization factor (ex. to have total weights = 1). In iteration T, the weight of the correctly classified examples will decrease, while increasing the weights of those misclassified, in order to provide more chances to select the next classifier.

BNC was considered an extension for the previous method, BFC. Minor changes were introduced to the previous algorithm, while working under AdaBoost.M2 method. BNC aims to improve the performance of BFC by calculating the information gain on the attribute itself, without resorting to its modality. BFC works only in binary context; it transforms the nominal data into binary data, which leads to high resource consumption and poor accuracy. BNC handles the nominal attributes directly without resorting to its modalities, using the information gain for selecting the best attribute, and then returning a set of classification rules, rather than a single one, as with BFC.

Another text[10] proposed a new classification method based on the dagging approach and FCA, called Dagging (Disjoint sample aggregating) Nominal Classifier (DNC). The method constructs multiple parallel classifiers; in this case, each classifier is constructed using the same learning algorithm. In order to perform bagging or dagging in a set of base classifiers, you need to prove that these classifiers are unstable. It has been proven that bagging will not improve the performance of stable classifiers, and in most cases, it will result in similar classification rules, since the stable classifiers will not be affected by changes in the training sets.

However, the authors prove the instability of the base classifiers by reporting the standard deviation of error rates for multiple data sets. They demonstrate this by running classifications with cross-validation on them. The proposed base learner (CNC), or Classifier Nominal Concept, runs in a similar way to BNC. It receives stratified samples from the distributer (Ensemble Paradigm. CNC will select the best attribute based on information gain evaluation measure, and then the algorithm will derive the formal concept using this attribute. The rule is deduced from the intent, and the class is deduced from the extent using majority vote. Later, a majority vote is conducted to select the appropriate class for the unseen data. The authors prefer dagging over the bagging method, as they claim it is more efficient. The dagging technique creates stratified subsets from the original dataset; each subset is used to create a classifier (classification rule in this case), and it is used mainly to reduce the chances of model over-fitting, and to reduce the effects of Noise. Using dagging on unstable base classifiers will ensure proper distribution, and it will result in several different classifiers with a low correlation between them. DNC algorithm is implemented on an existing method (dagging) to extend the notion of parallel ensemble by using FCA, in an attempt to tackle the limitation of a single classification. In addition, in such a scenario, validating the stability of the base classifier helps determine to which extent this approach is applicable. Moreover, two algorithms are introduced: one for concept induction, and the other to implement dagging and combine the classifiers deduced using the first one.

Furthermore, Kashnitsky and Ignatov[12] proposed Recommender-based Multiple Classifier System (RMCS), an FCA-recommender system that recommends the best classifier to use for specific objects among a set of different base classifiers. The base classifiers in this case could be heterogeneous or homogenous sets of classifiers (i.e. produced using different learning algorithms such as SVM, NB, Decision tree, etc.). The idea is more relevant to stacking, where level-1 classifiers send their decisions to the meta-learner, which learn from these decisions and select which classifier to use for specific object classification. The recommender system is based on Formal Concept Analysis, where it builds its recommendation criteria based on the notions of upper neighbors and lower neighbors of formal concept.

Given a training set, the recommender initializes by performing a cross validation on that training set, where all the classifiers are trained using leave-one-out cross-validation. Then a classification context (example: Table-1) is created using the results of the classifier, representing which classifier classified object x_i correctly. After constructing the classification context, the algorithm will train the set of classifiers with all the training examples, forming another table called a prediction table that contains each x_i from the test set and its label as well as the classifier that selected this label. For every object from the testing set, the algorithm finds its k nearest neighbors (using Euclidean distance) from the training set according to a specific metric, and then the algorithm searches for a concept in the classification context that has a maximal intersection with the set neighbors. After selecting the concept from the classification table, this concept will have a set of classifiers on its intent (or one).

These classifiers are considered suitable for classifying that object. In the case of several classifiers in the intent, a majority vote will be used to choose the majority labels selected by the recommended classifiers. The upper top concept and its neighbors are created by a function that uses the classification context. The concepts are obtained using the Close-By-One algorithm after modification. This means the sub-lattice used only the uppermost concept and its neighbors; here, these neighbors usually consist of multiple objects (instance number) and one attribute (classifier number). The upper concept is usually ignored; in most cases, it contains all the objects and empty sets of classifiers.

The algorithm performs training two times among n classifiers. The first time, it creates a classification table using leave-one-out cross-validation; the second time, it creates the prediction table where each object is associated with the classification decisions made by each classifier. Regardless of the time it takes to obtain the uppermost concepts, this approach is equal to the stacking method in terms of time complexity.

3.5 FCA-based Rules-Induction Methods

For a classification task, it is possible to obtain the conceptual classification rules using any conceptual coverage heuristic, or it can be obtained using the whole conceptual lattice; the produced concept can be used later for classification rules. Conceptual coverage is obtained using the Galois closure/connection. For example let us assume that the optimal points are $\{a_1, a_4\}$ in the context (table 4); calculating Galois closure from each point using the closure operators defined in chapter 3:

$$\text{Concept } c_1 = \{\lambda(\{a_1\}), \lambda o \gamma(\{a_1\})\} = \{\{o_1, o_2\}, \{a_1, a_2\}\}$$

$$\text{Concept } c_2 = \{\lambda(\{a_4\}), \lambda o \gamma(\{a_4\})\} = \{\{o_3, o_4, o_5\}, \{a_3, a_4\}\}$$

Table 4

Formal Context with Class labels

$O - \mathrm{A}$	a_1	a_2	a_3	a_4	$Class$
o_1	1	1	1	0	1
o_2	1	1	0	0	1
o_3	0	1	1	1	2
o_4	0	0	1	1	2
o_5	0	1	1	1	1

The rules can be obtained from a concept by taking majority vote of the class labels assigned to objects sub-set. For the previously obtained concepts; we will have the following rules:

$$IF a_1 \ and \ a_2 Then \text{ Class} = 1$$

$$IF a_3 \ and \ a_4 Then \text{ Class} = 2$$

What can be deduced from the second rule is that the concept is not always precise; where the attributes(a_3, a_4) are not always associated with class label 2, which may result in imprecise classification rules. Accordingly, we can introduce some weighting criteria to produce concepts such as confidence or support. Another approach is to perform oriented conceptual coverage by starting attributes/class pairs, or to start from the best point using certain evaluation methods.

3.6 Chapter Summary

In this chapter, we introduced Formal Concept Analysis, its applications and reasons behind adapting this approach in many computing related applications. We defined the main principles that constitute this approach including formal context, formal concept, conceptual lattice, closure operators and Galois connection. In section 3.3 we reviewed the conceptual decomposition heuristics and its importance to avoid the exponential complexity of constructing the conceptual lattice. In addition, we discussed different FCA learning methods whether they are exhaustive (single classifier) or under a certain ensemble learning paradigm and the conceptual decomposition approaches adopted by these methods. Finally, we defined the FCA rule induction methods, defining the common approach using a toy example.

As we mentioned earlier, we are going to adopt FCA in order to implement a learning algorithm, alongside an ensemble-learning paradigm to achieve our objectives with respect to the learning problem. The function of FCA and ensemble learning will be explained with more details in the next chapter.

CHAPTER 4: PROPOSED METHOD

In the previous chapter, we passed through FCA-based learning methods and their main properties. Therefore, in this chapter, we introduce a novel approach to induce rules using Formal Concept Analysis, in addition a new method that minimizes classification rule bias, and we introduce our argument on how we select the suitable ensemble-learning paradigm for obtaining better classification performance.

4.1 Random Conceptual Coverage Learner

The proposed approach is based on random conceptual coverage. Unlike other FCA coverage methods that select the best attribute to obtain the conceptual coverage, our algorithm performs the selection randomly from a given training data: $D = \{(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)\}$, where $x_i \in X$ a set of features, and $y_i \in Y$ represents the class label associated with set x_i. In the learning phase, the algorithm will iterate over N rows randomly selecting random number of features: $a_i \subseteq x_i$ without repetition. The selected features will not be selected again in the next iterations. The method of selecting random numbers of attribute values was to obtain different classification rules by utilizing the notion of random coverage in the model aggregation scenario. In order to obtain random values, the algorithm will set a random integer b; the value of b will change randomly in each iteration i of the training data D and vary from one to the length of the features set x_i.

For example, let us assume that we have attributes values describing object $x_i = \{AV_1, AV_2, AV_3\}$ with size = 3, and b is an integer with random value between 1 and 3 (size of x_i) for iteration i, let us say the random integer here is 2 then $a_i \subseteq x_i$ is a set of attributes values with size = b, for example $a_i = \{f_1, f_3\}$, and the values in a_i are selected randomly without repetition.

Algorithm 1

Random Conceptual Learner (RCL)

```
INPUT: Training Examples with class labels $D = \{(x_1, y_1), (x_2, y_2), \ldots\ (x_n, y_n)\}$
OUTPUT: Set of Classification Rules: $Rules$
PROCESS:
1. $Rules = \{\,\}$
2. For $i = 1, \ldots\ldots, n$:
3.     Randomly Select without replacement $b$ attribute values: $a_i \subseteq x_i$, $b$ random integer:
                  $a_i = sample(x_i, b), 1 \leq b \leq |x_i|$
4.     Calculate Galois Closure for $a_i$ to generate a Formal Concept: $c_i = \{\lambda(a_i), \lambda o \gamma(a_i)\}$
5.     Calculate the majority Class $vote$ associated with $\boldsymbol{\lambda}(a_i)$in $D$.

6.     Classification Rule: $rule_i = (\lambda o \gamma(a_i), vote)$
7.     Calculate the confidence $w_i = \frac{|Occurence(rule_i)|}{|Occurence(\lambda o \gamma(a_i))|}$ as a weight for the obtained rule.
8.     If $\boldsymbol{w_i = 1.0}$ : // rule covers only one category
9.         $Rules.add(rule_i)$
10.    Else :// rule covers more than one category
11.        Remaining attribute values that are not selected: $rem = Occurence(\lambda o \gamma(a_i)) - \lambda o \gamma(a_i)$
12.        $Rules.add(\boldsymbol{Enhanced\_Rule}(Occurence(\lambda o \gamma(a_i)), rem)$ )//execute algorithm-2
13. End
CLASSIFICATION:
    • For classification if more than one rule is triggered, the algorithm will assign
      the class value that has the maximum number of votes
```

Selected features will be used to obtain the formal concepts by calculating Galois closure $\lambda(a_i), \lambda o\gamma(a_i)$, where $\lambda(a_i)$ is a closed set of objects, and $\lambda o\gamma(a_i)$ is a closed set of attributes. The next step is to calculate the class label associated with $\lambda(a_i)$, and this is done through majority voting of the class labels associated with the closed set of objects $\boldsymbol{\lambda}(a_i)$ in the training data.

However, relying on voting may produce biased classification rules, as many instances that belong to other categories will be neglected. Therefore, the algorithm will calculate the confidence of the rule as weighting criteria, which is based on how many times the rule appeared in the training data associated with the most voted class over the total number of appearances in training data. The cardinality $|x|$, returns the number of instances in x:

$$w_i = \frac{|Occurence(rule_i)|}{\left|Occurence\big(\lambda o\gamma(a_i)\big)\right|} \tag{4.1}$$

The function $Occurence(x)$ returns all the instances in the training data D that are containing x. Rules with weight equal to 1.0 will be added directly to the classification rule set. On the other hand, rules with weights less than 1.0 will undergo an enhancing process that aims to minimize the classification rule bias caused by inadequate attribute values that construct the rule. To establish an optimal classification rule, we need to construct classification rules with minimal properties that trigger records that are belong to only one category. To achieve that, we propose the Rules Optimization Method; this method performs recursively to optimize biased classification rules. The proposed enhancement method (see Algorithm-2) is also based on FCA; the algorithm will iterate over the remaining (rem) attributes calculated in Algorithm-1. The remaining are the attributes that are not used in the obtained rule:

$$rem = Occurence(\lambda o\gamma(a_i)) - \lambda o\gamma(a_i) \tag{4.2}$$

The $Occurence(\lambda o\gamma(a_i))$ returns a set of instances in D containing $\lambda o\gamma(a_i)$**,** the remaining rem is a set of attributes that are not used in $\lambda o\gamma(a_i)$**.** From each remaining attribute, the algorithm will generate a formal concept and each concept will be weighted using the same weighting approach performed previously. Algorithm-2 will run recursively and produces two types of classification rules: The first type of rules—if the rule confidence is equal to 1.0 (Optimized Rule), the second type of rules—if the rule already has the maximum set of properties and cannot "grow" more. Once the $rem = \{\}$ the algorithm will stop the execution.

The rules resulting from this operation are optimized; many of them will describe only one category. However, some of the optimized rules may describe more than one category, and this is due to overlapping in the features of some datasets, where multiple records share the same properties but fall in different categories; these kinds of errors are irreducible. Similar to other classification methods like decision tree and naïve Bayesian, the proposed FCA-based learner will perform properly when it's applied to nominal attributes or intervals. Therefore, to obtain the best performance, we propose an embedded static discretizer, to create intervals from a given continuous attribute as a pre-learning process.

Algorithm 2

Rule Optimization Method ($Enhanced\_Rule$)

```
INPUTS: $D^* = Occurence(\lambda o \gamma(a_i)) \subseteq D; rem$// $D^*$ is a sub-context from $D$, $rem$ are the attributes in $D^*$ not used to construct the biased rule
OUTPUT: Set of Classification Rules: $Rules$
PROCESS:
1. $Rules = \{ \}$
2. For $i = 1, \dots \dots, n$:
3.     Calculate Galois Closure for $a_i \in rem$ to generate a Formal Concept: $c_i = \{\lambda(a_i), \lambda o \gamma(a_i)\}$
5.     Calculate the majority Class $vote$ associated with $\boldsymbol{\lambda}(a_i)$, in $D^*$

6.     Classification Rule: $rule_i = (\lambda o \gamma(a_i), vote)$
7.     Calculate the confidence $w_i = \frac{|Occurence(rule_i)|}{|Occurence(\lambda o \gamma(a_i))|}$ as a weight for the obtained rule.
8.     If $w_i = 1.0$ : // rule covers only one category
9.         $Rules.add(rule_i)$

10.    Else If $w_i < 1.0$ and $\lambda o \gamma(a_i)$ contains the maximum number of attributes:
11.        $Rules.add(rule_i)$
12.    Else:
13.        Remaining attribute values that are not selected: $rem = Occurence(\lambda o \gamma(a_i)) - \lambda o \gamma(a_i)$
14.        $Rules.add(\boldsymbol{Enhanced\_Rule}(Occurence(\lambda o \gamma(a_i)), rem))$ // recursive execution
15. Return($Rules$)
```

As an example, let us assume that we are learning from the dataset in table 5, and Algorithm-1 selects randomly attributes (a_3), then it will generate a concept based on the selected using Galois Closure $c = \{\lambda(a_3), \lambda o \gamma(a_3)\} = \{\{o_1, o_3, o_5\}, \{a_2, a_3\}\}$. The majority class associated to the set of objects $\{o_1, o_3, o_5\}$ is class 1, with confidence 0.66; which means that the classification rule $\{a_2, a_3\} \rightarrow Class\ 1$ will be triggered by instances that belongs to other classes, as the confidence of the rule is not equal to 1.0. Consequently, this rule will be enhanced using Algorithm-2, which will start the coverage from the remaining attributes values $rem = \{a_1, a_4\}$ and the sub-context D^* in table 6. Algorithm-2 will iterates over rem and generates two classification rules with confidence equal to 1.0; $\{a_1, a_2, a_3\} \rightarrow Class$ 2 and $\{a_2, a_3, a_4\} \rightarrow Class$ 1. Algorithm-2 will be executed recursively N times until the stopping conditions satisfied; rem becomes an empty set.

Table 5

Dataset Example ($\boldsymbol{D}$)

O|A	a_1	a_2	a_3	a_4	$Class$
o_1	0	1	1	1	1
o_2	1	1	0	0	1
o_3	1	1	1	0	2
o_4	0	0	0	1	2
o_5	0	1	1	1	1

Similar to other classification methods, such as Decision Tree and Naïve Bayesian, RCL will performs better when applied to scaled attributes. Therefore, it is important to discretize continuous values and transform them into intervals before applying RCL. Therefore, we embedded a Gini-Based discretizer to be used alongside our algorithm; this method will be discussed with more details in the next section.

Table 6

Sub-context $\boldsymbol{D}^{*}$*generated by Algorithm-1*

O|A	a_1	a_2	a_3	a_4	$Class$
o_1	0	1	1	1	1
o_3	1	1	1	0	2
o_5	0	1	1	1	1

4.2 Gini-Based Discretizer

Supervised discretization involves the processes of scaling the continuous values into ranges/intervals with respect to class values. These intervals might be labeled to form categories, which will be used for learning. The importance of discretization of continuous values is reflected in many aspects of supervised learning, mainly boosting the classification performance, and cutting down processing costs by acting as a reduction method. In addition, some of the learning algorithms deal only with scaled features or perform better in the presence of scaled attributes [48]. Industry literature is rich with various supervised discretization techniques; all are targeting the minimal information loss and the maximum selection accuracy. However, generating the best intervals has been labeled an NP-complete problem. Discretizes may be static, in which they perform the scaling before classification forming, or dynamic, where the process occurs at the same time of classification model creation, such as CART. In addition, some algorithms perform discretization on each attribute separately (univariate) or by considering all the attributes each time it looks for the best split point (multivariate).

In general, supervised discretization algorithms involve sorting, split point evaluation mechanisms, and stopping conditions. There are many evaluation mechanisms, however, the most popular and effective techniques are derived from information theory, such as entropy and Gini index, followed by Chi-Square and ChiMerge statistical techniques.

As mentioned previously, FCA-based classifiers require scaled attributes to perform properly. In this section, we propose an embedded supervised Gini-Gain discretizer (Algorithm-3), which is based on the Gini index evaluation metric. The reasoning supporting this measure will be discussed later in this section. The proposed approach is static, as it does not involve with in the learning procedures and will be executed before the learning phase. The main characteristic of this approach is that it simultaneously considers multiple features (multivariate). The method utilizes Gini index of impurity as an evaluation measure for creating intervals from given data. These intervals allow the model to be more predictive in the presence of continuous data, also acting as a data reduction mechanism and reducing the size of the training data. Though it is not easy to decide which evaluation measure is more effective than others; each of them has its own characteristics and applications. However, a comparative study by García[48] shows that evaluation measures based on information theory, such as entropy and Gini index, are among the best, based on empirical studies that involve other measures. Moreover, a theoretical comparison conducted by Raileanu[49] found that there are slight differences between Shannon entropy and Gini index, in terms of finding the best split points, as they suggest that differences will not achieve more than 2% by experimenting with the frequency of disagreement between the two approaches.

However, the Gini evaluation method performs faster as it does not involve any logarithmic calculation. Moreover, choosing evaluation criteria is not an objective by itself for this study; thus, any approach with acceptable performance is also applicable.

With the given training data, $D = \{(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\}$, the discretizer will operate recursively until the stopping conditions are satisfied, and it returns a set of bins that will be used later for creating intervals in the training data. The algorithm starts with all attribute values, and greedily checks for the best split point over all possible split points in the training data. For all possible split points, the algorithm will split the data into two groups using the function$Grouping()$, which returns binary groups, $Groups = \{g_1, g_2\}$, and evaluate each group using the Gini evaluation measure:

$$\text{Gini}(g_j) = 1 - \sum_{\boldsymbol{k}=1}^{\boldsymbol{l}} (P_{g_j}^k)^{\mathbf{2}} \tag{4.3}$$

$$where, \qquad P_{g_j}^k = \frac{count(y_k)}{size_{g_j}} \tag{4.4}$$

The evaluation is based on how many distinct class values are in each group. The best Gini indication is achieved when each group contains one class value, where the function Gini(g_j) = 0 for the two groups. Consequently, it will minimize the gain function:

$$Gain = \sum_{j=1}^{m} \text{Gini}(g_j).\frac{size_{g_j}}{size_D} \tag{4.5}$$

Also, consequently, each group that contains more than one class value will be divided again until satisfying the stopping condition. All selected split points with minimal gain will be added to the bins list to be used later to create intervals on the training data.

To explain the discretizer with more details, we will take the dataset in Table 7. This dataset contains 4 attributes; two of them are numeric while the other two attributes are categorical, with 2 class values. Before feeding the data to the Gini-Discretizer, the categorical attributes (outlook, windy) will be labeled to numerical coefficients (i.e. the attribute value overcast will be labeled as 1.0, rainy as 2.0 and sunny as 3.0), and the same for windy. The discretizer will start by evaluating all the attributes values in the dataset to find the first split point; this is done by calculating the Gini-Gain (equation-4.5) for each unique attribute value. The binary discretizer will search for the point that can split the dataset into two groups where each group belongs to one class category; in the ideal scenario, it will return zero as a score. The value with minimal Gini-Gain will be selected as the best split point, in this scenario the algorithm selected Outlook value = 1.0 (overcast) as the first split point and this is done through the following process:

- Grouping the data into two groups; where group-1 contains instances with outlook value less than or equal to 1.0, group-2 with instances with outlook values larger than 1.0.
- Calculating the probability score P for each group by counting the number of

class values of a certain class in each group over the size of the group:

- Group-1: $P_{g_1}^{0} = \frac{count(y=0)}{size_{g_1}} = \frac{0}{4} = 0$, $P_{g_1}^{1} = \frac{count(y=0)}{size_{g_1}} = \frac{4}{4} = 1$
- Group-2: $P_{g_2}^{0} = \frac{count(y=0)}{size_{g_2}} = \frac{5}{10} = 0.5$, $P_{g_2}^{1} = \frac{count(y=0)}{size_{g_2}} = \frac{5}{10} = 0.5$

* Calculating the Gini score for each group using equation (4.4):
    - Group 1: $\text{Gini}(g_1) = 1 - (0)^2 + (1)^2 = 0$
    - Group 2: $\text{Gini}(g_2) = 1 - (0.5)^2 + (0.5)^2 = 0.5$.
* Calculating the Gain using equation (4.5):
    - $Gain = \text{Gini}(g_1).\frac{size_{g_1}}{size_D} + \text{Gini}(g_2).\frac{size_{g_2}}{size_D} = 0.\frac{4}{14} + 0.5.\frac{10}{14} = 0.357$.

The discretizer will continue iterating over the rest of attributes value searching for the best split point. For this scenario the best split point is outlook value = 1.0 (overcast), with minimal gain against others, and this point will be added to the bins list to be used later for creating intervals in the dataset.

Algorithm 3

Gini-embedded discretizer

```
Gini_Evaluation($Groups, Y$):
PROCESS:
 // Calculate the score for each group:
1.  For $g_j \in Groups$:
2.      For each class value $y_k \in Y$:
3.          $P_{g_j}^{k} = \frac{|y_k|}{size_{g_j}}$
4.          Gini($g_j$) $= 1 - \sum_{k=1}^{l} (P_{g_j}^{k})^2$
    // Calculate Gini Gain:
5.  $Gain = \sum_{j=1}^{m}$ Gini$(g_j) . \frac{size_{g_j}}{size_D}$
6. RETURN($Gain$)
```

```
Binary_Splitter($D$):
PROCESS:
1.Gini_Gain = {}
2. For $i = 1, \dots \dots, n$:
3.     For each attribute value $a_k \in x_i$:
4.         // perform binary grouping based on split point  $Split_{Point} = a_k;\ index = k$ :
           Groups$_{a_k}$ = Grouping($Split_{Point}, index, D$) // binary grouping data on $Split_{Point}$
5.         Gini_Gain.$add$(Gini_Evaluation (Groups$_{a_k}$, $Y$)):
  // Return the groups and split point that minimizes Gain function:
6. RETURN(Groups$_{av_k}$, $Split_{Point}$)
```

```
Gini_Discretizer:
INPUTS: Training Examples with class labels $D = \{(x_1, y_1), (x_2, y_2), \dots\ (x_n, y_n)\}$
OUTPUT: Set of bins: $bins$
PROCESS:
1. $bins$ = {}
2. $Groups, Split_{Point}$ = Binary_Splitter($D$)
3. $bins.add(Split_{Point})$
 // Check if each group contains only one class:
4. For $g_i \in Groups$:
5.     IF $|Y_{g_i}| > 1.0$:
6.         $bins$ += Gini_Discretizer($g_i$)
7. RETURN($bins$)
```

Table 7

Golf dataset before scaling.

Outlook	Temperature	Humidity	Windy	Play
overcast	83	86	TRUE	yes
overcast	64	65	TRUE	yes
overcast	72	90	False	yes
overcast	81	75	TRUE	yes
rainy	70	96	TRUE	yes
rainy	68	80	TRUE	yes
rainy	65	70	TRUE	no
rainy	75	80	TRUE	yes
rainy	71	91	TRUE	no
sunny	85	85	TRUE	no
sunny	80	90	TRUE	no
sunny	72	95	TRUE	no
sunny	69	70	TRUE	yes
sunny	75	70	FALSE	yes

Table 8

Golf dataset after scaling.

Outlook	Temperature	Humidity	Play
Outlook <= 1	Temperature >70	Humidity >80	yes
Outlook <= 1	Temperature <=65	Humidity <=80	yes
Outlook <= 1	Temperature >70	Humidity <=80	yes
Outlook > 1	Temperature <=70	Humidity >80	yes
Outlook > 1	Temperature <=70	Humidity <=80	yes
Outlook > 1	Temperature <=65	Humidity <=80	no
Outlook > 1	Temperature >70	Humidity <=80	yes
Outlook > 1	Temperature >70	Humidity >80	no

The obtained bins will be used to create intervals (see table 9) in the datasets before the learning process carried on. There is no split points found in windy attribute as it is not containing useful split information for the algorithm, and it will be neglected in the intervals generation process. Finally, the dataset will be reduced after merging the attributes value inside these intervals as we can see in table 8 cases, the dataset is reduced from 14 instances to only 8 instances after dropping the duplicates. RCL will benefit from this phase as it allows the learner to perform faster, removing the unnecessary attributes and scale the data so it can be used effectively for learning. Now we have a group of instances that belong to one category, this is meaning that group-1 satisfies the stopping condition and no longer involved in the searching process. The algorithm will be executed recursively on Group-2 to create more sub groups until it satisfies the stopping condition, and on each execution, the best split point will be added to the bins list.

Table 9

Bins obtained by the discretizer.

Attributes	Outlook	Temperature	Humidity	Windy
Bins	1.0 (overcast)	70.0, 65.0	80.0	n/a

The proposed methods (Algorithms: 1, 2, 3), are designed to minimize the error caused by biased classification rules. Algorithm-2 enhances the conceptual coverage in order to produce rules with minimal bias by adding more properties, while creating intervals on the training data. This reduces the chance of a rule becoming too focused on a specific value. However, as we mentioned in Chapter 1, the learning error is not limited to the model bias; the variance may produce a high error rate in the presence of a model that learns deeply from the training data, reducing the model’s ability to perform in the presence of new data. Therefore, in the next section, we are going to explain the proposed method for variance averaging.

4.4 RCL Ensemble Paradigm

In much of the literature, there are many techniques proposed for variance reduction, such as rule pruning, tree pruning, data partitioning, scaling, and ensemble learning. The last of these options was found to be more effective in different scenarios, as previously discussed in chapters 1, 2, and 3. However, it exists in multiple paradigms. Arbitrarily selecting the ensemble paradigm for a certain learner might be tricky, and may lead to a drop in the classification performance. Sequential ensemble methods aim to boost the accuracy of weak learning algorithms, which tends to under-fit the data and produce biased classification hypotheses. Conversely, parallel ensembles tend to reduce the variance of unstable learning algorithms, which over-fit and produce complex hypotheses, while producing high error rates when it comes to unforeseen instances.

In order to choose the best ensemble paradigm, it is essential to measure the stability of the base learner. Unstable classifiers have a high variance, as they over-fit the training data. The training error of the unstable classifiers is very low, while the validation error tends to be high. On the other hand, stable classifiers, have low variance and will probably exhibit high bias. In the model aggregation scenario, unstable classifiers produce different results if small changes occur in the data, while stable classifiers tend to agree in general, even if some changes are applied to the training examples. Sequential ensemble methods are used to boost the weak classifiers and reduce the bias, but parallel ensembles average the variance of the unstable classifiers. Therefore, to ideally utilize ensemble-learning paradigms and to decide which ensemble method best fits our objectives; we have to verify which category the proposed method falls within. The stability of the learning algorithm can be experimented by using various techniques, such as the variance/standard deviation in classification errors, training data injection, or simply by verifying the effect of sampling on the accuracy when aggregating multiple models together. Empirically, bagging was found to be effective for variance reduction. Therefore, to reduce the variance caused by RCL's expected tendency to over-fit the training data, we propose fusing RCL with Bagging paradigm (see algorithm-4) to utilize its ability to average the variance. To verify its efficiency in reducing the variance of RCL we would show the experiment results of the effect of bagging in chapter-5.

Using Bagging averages the variance by generating multiple samples from the training data; the samples are generated using bootstrap sampling. Each of the generated samples will be used to produce a different classification model. The discretizer will be used locally (see Figure 7) to further enforcing the diversity between the produced classifiers and to reduce the effects of noise on the discretization process.

Algorithm-4: RCL-Bagging Paradigm

```
INPUT: Training Examples with class labels $D = \{(o_1, y_1), (o_2, y_2), \ldots (o_n, y_n)\}$;
       N: Number of base classifiers
OUTPUT: $H(x) = arg \max_{y \in Y} \sum_{i=0}^{N} \mathbb{I}(h_i(x) = y)$
PROCESS:
1. For i = 1,….N:
2.     Randomly Select with Replacement $M$ instances from $D$, and create $D^*_i$ bootstrap sample;
       M = size($D$); $D^*_i \neq D$.
3.     $h_i = \boldsymbol{RCL}(D^*_i)$ // ith classification model generated using random conceptual learner.
4. End
```

Sampling reduces the correlation between the produced models, which allows the learner to produce a different hypothesis with minimal correlation based on the sampled data. Each sample size is equal to the size of the original training set; however, it only contains two-thirds of the training instances of the original training set because of the repetition chances.

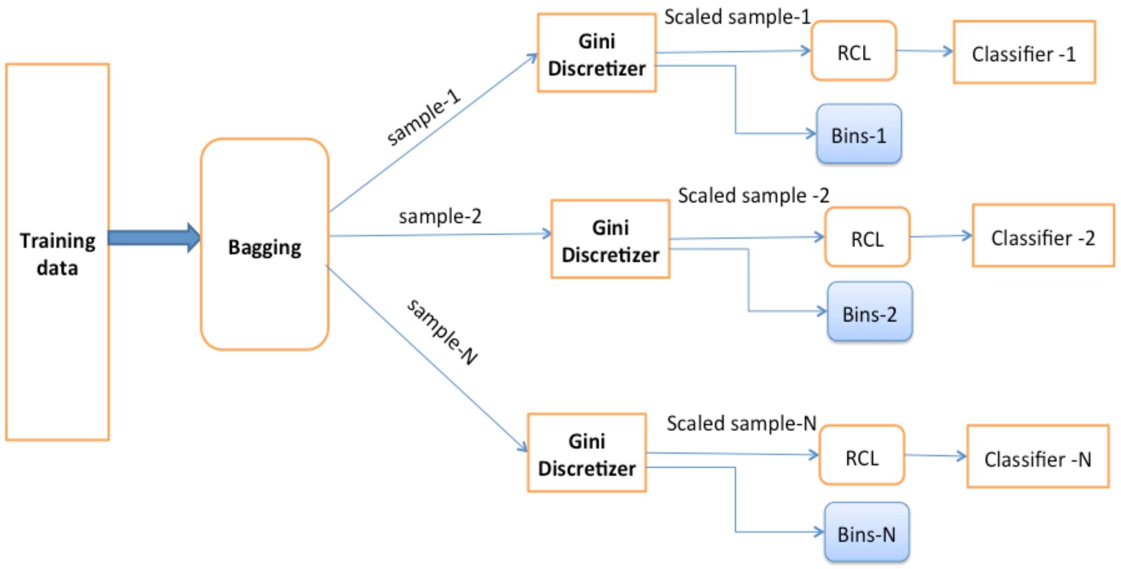


Figure 7. Bagged RCL learning schema

In the classification phase (see Figure 8), the same validation data will be tested by each classifier and the final decision is based on aggregating various decisions made by individual models using majority voting.

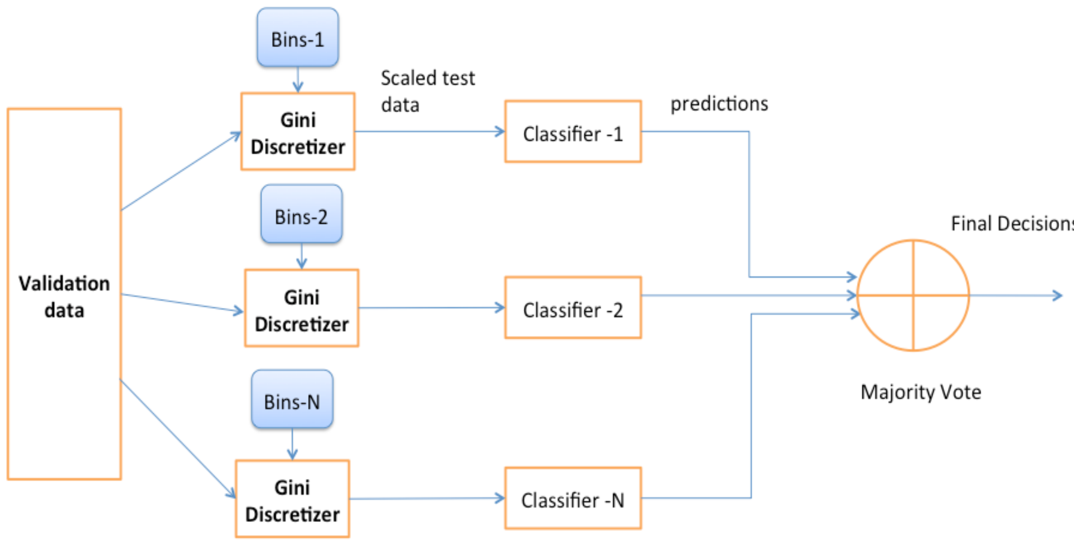


Figure 8. Bagged RCL classification schema

4.5 Bagged RCL vs. other FCA-based Ensemble Learning Methods

Previously in chapter 3, we highlighted some of the learning methods that are based on FCA and utilize certain ensemble paradigm, each learning method has its own characteristics in terms of the conceptual structure they use, the conceptual coverage method, type of inputs and the classification method. Base learners in BFC[11], BNC[42], and DNC[10]start the conceptual coverage by selecting certain attribute/s that satisfy a specific quality measure such as Shannon entropy, in addition these approaches only constructs and utilize part of the conceptual lattice.

RMCS meta-learner[12] performs the conceptual coverage based on distance measure (Euclidean Distance), in addition, RCMS constructs the complete conceptual lattice. Our proposed method Bagged RCL (B-RCL), only constructs and uses part of the lattice and performs the coverage in random fashion as explained previously, in addition it utilize an embedded static Gini-gain discretizer. Table 10 highlights the main characteristics of the reviewed FCA-based learners.

Table 10

FCA-Based Learners main characteristics

Learning Method	Conceptual Structure	Inputs	Coverage Method	Classification	Discretization Method
B-RCL	Sub-lattice	Nominal	Random Coverage	Majority Vote	Embedded Gini-gain discreteizer
RMCS	Lattice top concepts	Binary	Euclidean Distance	Defined by base classifier.	Depends on the used set of base classifiers
BFC	Sub-lattice	Nominal	Shannon Entropy	Weighted Vote	n/a – external
BNC	Sub-lattice	Nominal	Information Gain	Weighted Vote	n/a – external
DNC	Sub-lattice	Nominal	Information Gain	Majority Vote	n/a – external

4.6 Chapter Summary

In this chapter, we introduced new method for learning using FCA. The first algorithm proposed to mine the concepts in a random fashion to improve the diversity between the generated classifiers. The second method is proposed to reduce the bias of the classification rules obtained through the process of random conceptual coverage. In addition, we exploited a scaling method based on Gini Index evaluation criteria, the reasons behind using this approach stated earlier. In addition, we proposed our argument regarding the selection of the best ensemble-learning paradigm, which will be verified in the following chapter. Based on that, in chapter 5, we will experiment the stability of the proposed learning algorithm to verify our choice. In addition, we will conduct an empirical comparative study to compare our proposed method against other classification approaches.

CHAPTER 5: EXPERIMENTS

Next, we compare our proposed method against other traditional ensemble methods, Bagging, Random Forests and Adaboost. The experiments would demonstrate the performance of our method and its validity. The comparison illustrates the differences between our method and traditional algorithms. In section 5.1 we present the used datasets properties to have an overview of what datasets are used and their suitability for use in experiments. In section 5.2, we will experiment the stability of the RCL, in section 5.3 we will define the experiments setup of the comparative study, including the testing environment, and we will explain the reasons behind the used configurations for each experimented algorithm. In section 5.4 we will show and discuss the results obtained from the comparative experiments.

5.1 Datasets Characteristics

In order, to test our method we will use (23) datasets as illustrated in Table-11, these datasets are downloaded from UCI-Repository[50] and from other sources[51], [52]. The characteristics of the used datasets are vary, where some of them are containing only categorical attributes, others are continuous and some of them contain different types of features. Two reasons behind using these datasets: the first reason is the popularity and the usability of these datasets; in literature they have been commonly used to validate classifiers accuracy such in[7], [10], [12], [16], [20], [42], [53]. The second reason is the variance between these datasets in terms of features types (categorical, continues, or both), number of classes in each dataset and the total number of instances. In addition, this allows us to experiment our work with many options to see how they reflect on the performance of our algorithm.

Table 11

Datasets Characteristics

Dataset	Features	Instances	Data Types	Classes
Iris	4	150	Continuous	3
Breast Cancer	9	683	Continuous	2
Sonar	60	208	Continuous	2
Glass	9	214	Continuous	6
Vowel	10	990	Hybrid	11
Ionosphere	34	351	Hybrid	2
German credit	24	1000	Hybrid	2
Ecoli	7	336	Continuous	8
Hayes-Roth	4	160	Categorical	3
Car	6	1728	Categorical	4
Zoo	16	101	Categorical	7
Liver	6	345	Hybrid	2
Wine	13	178	Continuous	3
Heart	13	270	Hybrid	2
Balance	4	625	Categorical	3
MPG	11	234	Hybrid	7
Immunotherapy	7	90	Continuous	2
Cryotherapy	6	90	Continuous	3
Waveform	40	5000	Continuous	3
Twonorm	20	7400	Continuous	2
Letters	16	20000	Continuous	26
Sat-image	36	6435	Continuous	6
Ringnorm	20	7400	Continuous	2

Table 11 illustrates the used datasets and their properties, based on the number of features, the size of each dataset and the data types; whether it is continuous or categorical or a combination of both.

5.2 The Stability of the RCL

To measure the stability of RCL, we set up an experiment using 5 datasets from UCI-Repository[50]. The experiment is executed using a single training set (80% of the data) and 4 testing sets; 3 of them are sampled from the remaining 20% of the data that was not used in training, the 4th is the same data used for training. We preferred this scenario rather than cross-validation because we are interested to see whether bootstrap sampling enhance the accuracy in all cases as well as to run the experiment in a before/after fashion. The first stage is to learn the model in the training data and test the training error using the same training data for validation giving an apparent error.

Table 12

Training Error vs. Testing Error using RCL (%)

Dataset	$Train^{err}$	${Test^{err}}_1$	${Test^{err}}_2$	${Test^{err}}_3$
Iris	0.0	26.0	6.67	20.0
Hayth-Roth	12.5	31.25	37.5	34.37
Ecoli	0.0	33.8	35.29	35.29
Wine	0.0	13.88	19.44	22.22
IMtherapy	0.0	11.1	5.5	38.8

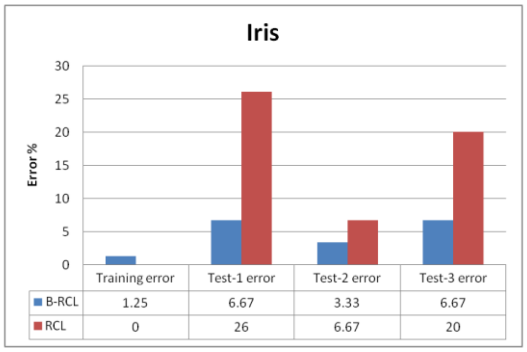
Then, we run the tests using the three sample testing sets to verify the variance in prediction errors. The obtained results (see table 12) show that the RCL has a very low training error in most of the used datasets. However, the error increases dramatically when the generated model is validated using unseen data. These results suggest that the RCL is unstable as it tends to over-fit the training data, and this is due to complexity in the learning procedures. Consequently, we propose bagging as an approximation for this problem. In the next subsection, we are going to explain how to utilize the parallel paradigm to average the variance produced by RCL.

The experiment is executed using the same training data and the same 3 sampled test sets from the previous experiment without any changes. The algorithm is setup with N = 10; generating 10 samples with replacements from the training data and feeding them to RCL to produce 10 classifiers. In the testing phase, the same testing samples are used to validate the aggregated model. Table 13 shows the training/testing errors for each dataset. Empirically, bagging was found to be effective for variance reduction. Therefore, to verify its efficiency in reducing the variance of RCL we will repeat the previous experiment, using B-RCL (see algorithm-4).

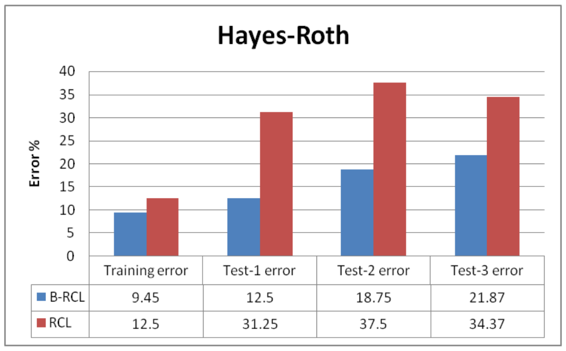
Table 13

Effect of Bagging-RCL on Training/Testing Errors (%)

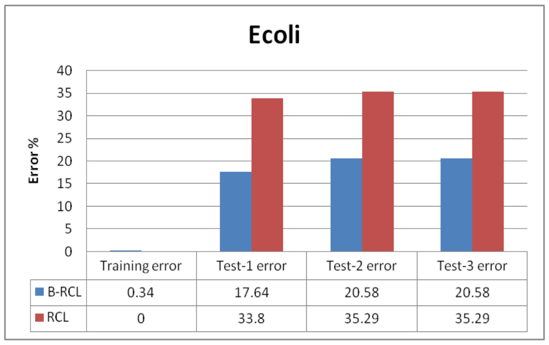
Dataset	$Train^{err}$	${Test^{err}}_1$	${Test^{err}}_2$	${Test^{err}}_3$
Iris	1.25	6.67	3.33	6.67
Hayth-Roth	9.45	12.5	18.75	21.87
Ecoli	0.34	17.64	20.58	20.58
Wine	1.06	5.5	11.1	11.1
IMtherapy	0.69	11.1	5.5	11.1



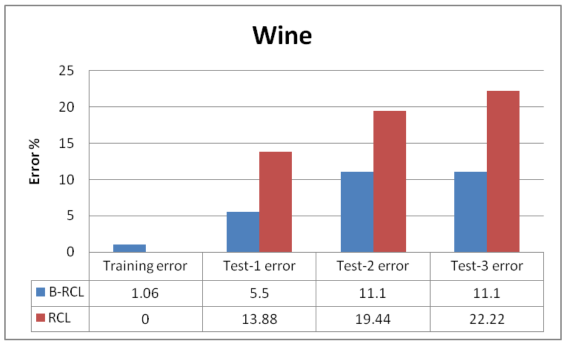
	Training error	Test-1 error	Test-2 error	Test-3 error
🟦 B-RCL	1.25	6.67	3.33	6.67
🟥 RCL	0	26	6.67	20



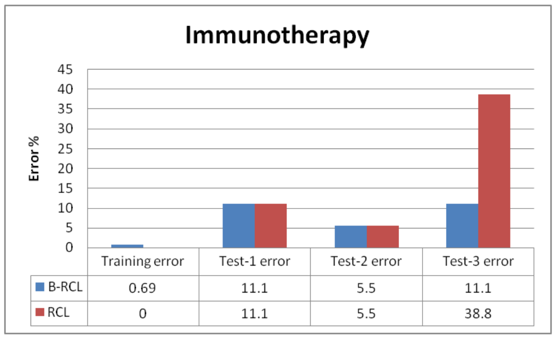
	Training error	Test-1 error	Test-2 error	Test-3 error
🟦 B-RCL	9.45	12.5	18.75	21.87
🟥 RCL	12.5	31.25	37.5	34.37



	Training error	Test-1 error	Test-2 error	Test-3 error
🟦 B-RCL	0.34	17.64	20.58	20.58
🟥 RCL	0	33.8	35.29	35.29



	Training error	Test-1 error	Test-2 error	Test-3 error
🟦 B-RCL	1.06	5.5	11.1	11.1
🟥 RCL	0	13.88	19.44	22.22



	Training error	Test-1 error	Test-2 error	Test-3 error
🟦 B-RCL	0.69	11.1	5.5	11.1
🟥 RCL	0	11.1	5.5	38.8

Figure 9. The stability of RCL.

Results from the bagging experiments show that the testing errors are averaged in most of the testing samples (see figure 9), while few testing errors remain the same, as with IM therapy $\boldsymbol{Test^{err}}_{\mathbf{1}}$ and $\boldsymbol{Test^{err}}_{\mathbf{2}}$. However, we can notice a slight increase in the training error, and this is due to the absence of some representative examples from the training data because of the sampling. In general, these results verify the instability of the base learner; therefore, it will perform better with the parallel ensemble-learning paradigm. Moreover, it is important to state that the randomness in conceptual covering also contributes to producing less correlated classifiers.

5.3 Comparative experiments setup

The experiments setup is unified among all the ensembles; we use 10-folds cross validation for each dataset, in addition each experiment was repeated 10-times and the performance will be averaged from the 10 executions. We experiment our algorithm against scikit-learn[54] implementations of Bagging Predictors, Random Forests, Gradient Boosting and AdaBoost. To experiment Bagged-RCL ability to learn from small proportion of the data, we setup the experiment on the large datasets to use 1 out of 10 folds to train the learner, and the 9 folds to be used for testing only. These datasets are: Sat-image, Letters, Ringnorm, Twonorm and Waveform.

For Bagged-RCL (B-RCL) we adjust the ensemble parameters to produce 40 classifiers and there are two reasons to select this number. The first reason comes from verifying the effect of the number of classifiers on the average accuracy (see figure 10). This experiment is conducted using train/test split approach and the reason is to hold the exact train and test data without changes while adjusting the number of classifiers parameter incrementally. For each parameter, repeat the training and testing 10 times to reduce the effect of randomness on the results. The second reason is to reduce the learning time by not going very far to a large number while there are no significant changes on the accuracy. Figure 10 shows the experiment results on 4 datasets, representing the variation on accuracy with respect to the number of classifiers. We can see that the accuracy level stabilizes at 20 classifiers or earlier, which indicates that 40 classifiers is suitable to run the comparative experiment.

In addition, we configure the Gini-Discretizer to its default configuration; where it will perform recursively until it satisfies the stopping condition; the maximum number of classes in the last interval should equal to one. In addition, the feature selection method is based on sampling with replacement each instance (row); where the sample size varies from instance to another, starting from one and limited to the number of features in each instance. On the other hand, we are interested in unifying the experiment against the other approaches, therefore all other ensemble methods are trained using the same parameter (40 classifiers). With different tuning possibilities, we will have a very large number of possible combinations for all datasets, and each possible combination might generate different accuracy with respect to others.

Bagging and Random forests are based on CART approach, in addition, each of the trees are grown to limit, without pruning to ensure the minimum bias the can be produced by each tree as well as to have a fair comparison against our approach. We used the default features selection method for Random Forests; where each node split is based on sampled number of features given by the square root of the possible number of features to split on. Furthermore, bagging configuration only varies from Random Forests in term of feature selection; bagging algorithm uses the whole features to define the best split point. Gradient Boosting and Multi-Class AdaBoost(SAMME) parameters are set to default with learning rates equal to 0.1 and 1.0 respectively; constructing and testing 40 trees for each of them.

All experiments executed in DELL and Macintosh Notebooks, with Core-i7, 8

GB RAM,and Core-i5, 4 GB RAM respectively. We used Anaconda Navigator Python 3.6 to implement Bagged-RCL and for testing B-RCL against others.

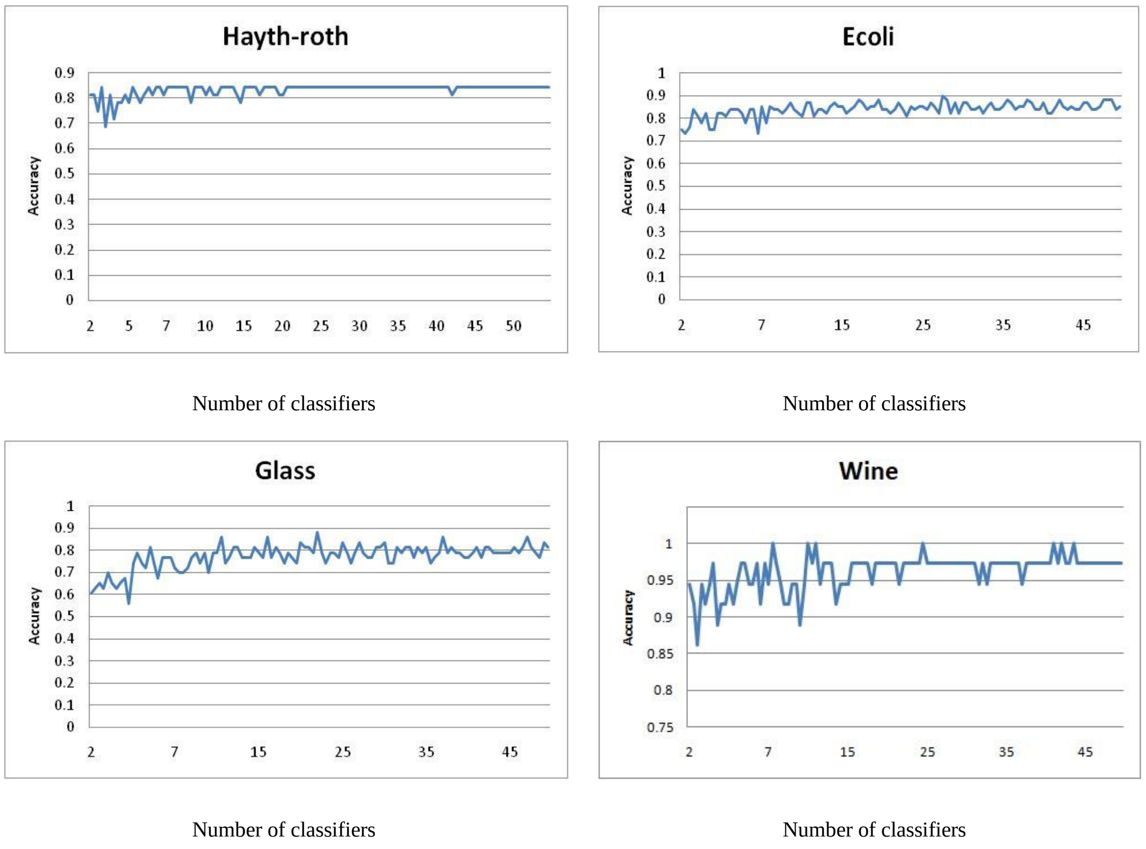


Figure 10. Effect of classifiers number on Bagged-RCL

5.4 Results

In Table 14, we show the experiments results based on the configurations stated in the previous section. For each ensemble method, we report the average accuracy of the repeated 10 * 10-folds cross-validation. The reported accuracy indicates the number of correctly classified records over the total number of validation records. Generally, we can see that the obtained results are varies between different classification models; where some models performs better in certain dataset(s) and others perform poorly in some datasets. Therefore, to decide in which datasets B-RCL outperforms other models, not just an artifact of randomness is by decomposing the average score into the score of the 10 executions, and then we draw line diagrams to observe the variance over the results during the execution. We can see that B-RCL achieves higher results in Iris, Breastcancer, Glass, Hayes-Roth, Heart and MPG; consequently, we perform comparison between B-RCL and the nearest competitor based on the behaviors over multiple executions of the algorithms in these datasets to determine the significance of the obtained results. As shown in figure 11, we can observe that B-RCL outperforms Random Forests in 9 out of 10 iterations in Breast Cancer dataset, as well as for 7 runs out of 10 against Bagging Predictors using Hayes-Roth data set. In addition, it outperforms Gradient Boosting in 9 iterations on MPG dataset. For Iris, Glass, Heart we can observe that there is overlapping over the iterations; this means that the results are very close and we cannot judge the performance for these datasets over finite iterations. Further, we can see that AdaBoost have the least performance among others where it only outperforms other methods in one scenario shared with Bagging Predictors, while Random Forests preserves the highest performance in 4 out of 5 datasets.

Table 14

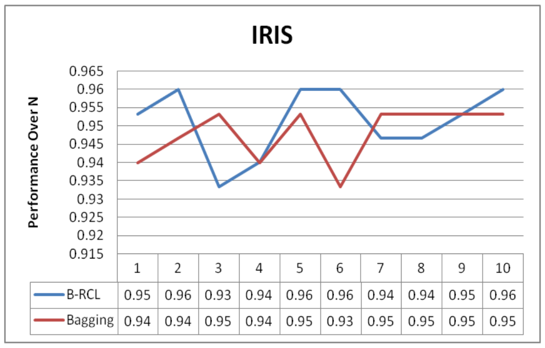
Experiments Results (Accuracy %)

Dataset	SAMME	Bagging	Gradient Boosting	Random Forests	B-RCL
Iris	94.6	94.8	94.7	94.7	**95.1**
Breast Cancer	93.6*	95.5*	95.5*	96.4	**96.9**
Sonar	70.6*	78.7	**82.1***	81.5*	78.3
Glass	65.9*	75.1	74.1*	76.6	**77.3**
Vowel	78.1*	91.0	84.7*	**95.2***	90.4
Ionosphere	88.8*	91.5	92.3	**93.1**	92.5
German credit	69.1*	74.4*	**75.1***	74.9*	72.0
Ecoli	79.3*	84.1	84.9	**86.4***	84.4
Hayes-Roth	81.8*	82.4	80.1*	81.6*	**84.3**
Car	**98.1***	**98.1***	92.6*	97.7*	94.9
Zoo	95.2	**95.7**	95.3	95.3	95.2
Liver	62.6*	70.2	**73.0***	71.9*	68.7
Wine	91.0*	96.6	93.8*	**98.0***	96.1
Heart	73.2*	80.7*	81.5	82.7	**83.0**
Balance	77.5*	80.1*	**87.0***	82.6	82.4
MPG	90.6*	93.6*	94.0*	89.8*	**95.7**
Waveform	70.5*	80.7*	82.3*	**82.7**	82.6
Twonorm	80.5*	94.5*	94.3*	95.6*	**96.0**
Letters	70.7	82.7*	79.2*	**84.7***	70.7
Sat-image	79.9*	86.4	86.0*	**87.5***	86.4
Ringnorm	83.0*	92.5*	93.3*	**93.9***	92.2
Immunotherapy	80.9	**84.7**	84.6	84.6	84.1
Cryotherapy	87.2*	90.6	89.3	**92.5**	91.2

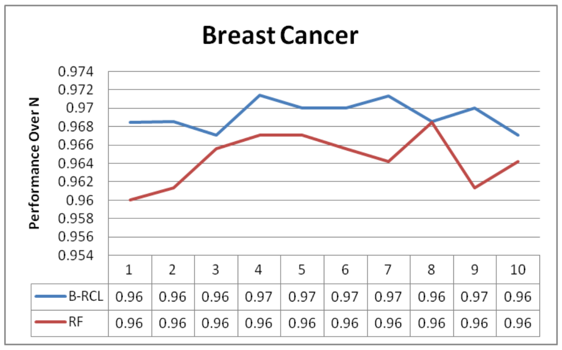
* B-RCL is significantly worse, * B-RCL is significantly better, level of significance 0.05

For the largest datasets (Sat-image, Letters, Ringnorm, Twonorm and Waveform), we used only 10% (1-fold) of the data to train each classifiers, while using the remaining 90% (9-folds) of the data in testing. From the obtained results, we can see that B-RCL is able to learn successfully from small portions of data and obtain good results. In order to analyze the obtained result and we performed two-tailed unpaired t test[55]. The reason behind using this method is to make sure that the discrepancies in accuracies are not from the same distribution and there are significant differences between them; therefore, this test is used to reject the null hypotheses of equal means. In this test we used the raw results from the experiments, 100 accuracy samples from each dataset as a result of 10 ×

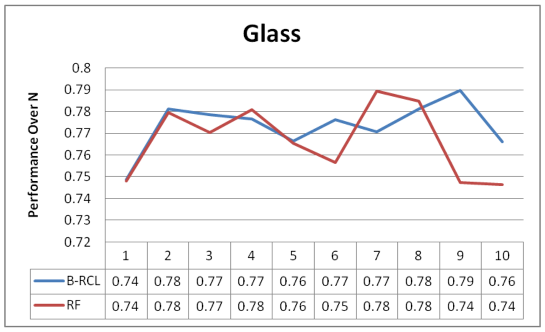
10 folds. In table 14, we indicate the significant differences with character (*), and the mean (cross-validation average) with bold font. Finally, we summarized the results from significance test in table 15. Generally, we can assume that our results fall in the same ranges compared to the other ensemble methods with these configurations.



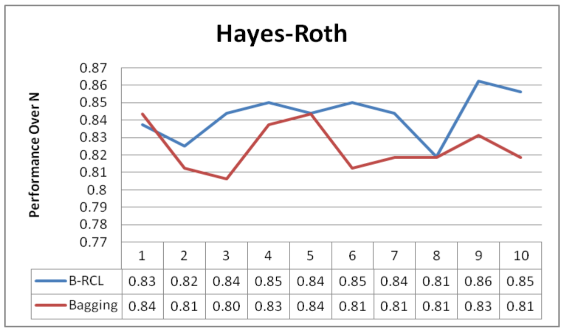
	1	2	3	4	5	6	7	8	9	10
B-RCL	0.95	0.96	0.93	0.94	0.96	0.96	0.94	0.94	0.95	0.96
Bagging	0.94	0.94	0.95	0.94	0.95	0.93	0.95	0.95	0.95	0.95



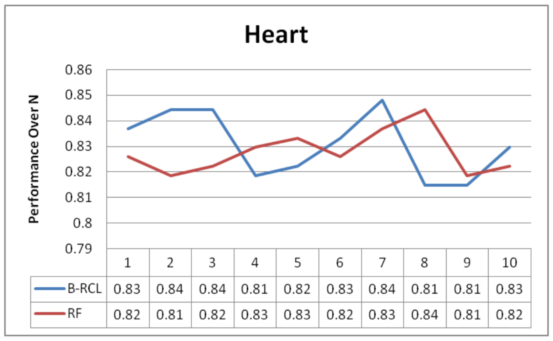
	1	2	3	4	5	6	7	8	9	10
B-RCL	0.96	0.96	0.96	0.97	0.97	0.97	0.97	0.96	0.97	0.96
RF	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96



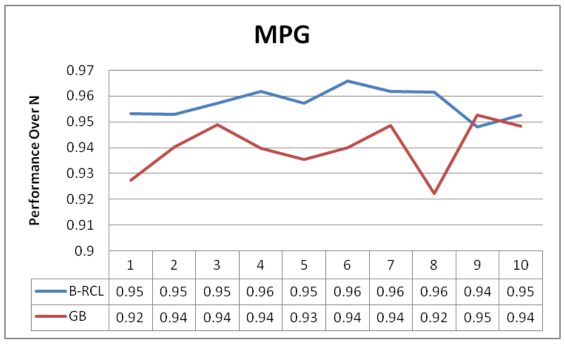
	1	2	3	4	5	6	7	8	9	10
B-RCL	0.74	0.78	0.77	0.77	0.76	0.77	0.77	0.78	0.79	0.76
RF	0.74	0.78	0.77	0.78	0.76	0.75	0.78	0.78	0.74	0.74



	1	2	3	4	5	6	7	8	9	10
B-RCL	0.83	0.82	0.84	0.85	0.84	0.85	0.84	0.81	0.86	0.85
Bagging	0.84	0.81	0.80	0.83	0.84	0.81	0.81	0.81	0.83	0.81



	1	2	3	4	5	6	7	8	9	10
B-RCL	0.83	0.84	0.84	0.81	0.82	0.83	0.84	0.81	0.81	0.83
RF	0.82	0.81	0.82	0.83	0.83	0.82	0.83	0.84	0.81	0.82



	1	2	3	4	5	6	7	8	9	10
B-RCL	0.95	0.95	0.95	0.96	0.95	0.96	0.96	0.96	0.94	0.95
GB	0.92	0.94	0.94	0.94	0.93	0.94	0.94	0.92	0.95	0.94

Figure 11. Performance of B-RCL against the nearst competitor.

Table 15

Significance test results

	RCL is Better	RCL is Worse	No significant difference
SAMME	18	1	4
Bagging Predictors	6	4	13
Gradient Boosting	10	6	7
Random Forests	3	10	10

5.5 Chapter Summary

In this chapter, we verified the usability of our new method and how it performs compared to others. We mainly conducted 3 experiments, the first experiment to test the stability of our proposed method in order to identify the most suitable ensemble method to be used alongside. The second experiment was conducted mainly to tune the ensemble hyper parameter, and to validate our selection to a certain number of classifiers in ensemble. The comparative study in section 5.4 confirms that we satisfied our main objective, by showing results that are within the range as other methods and in many cases, it outperforms other methods. Additional verification is performed through unpaired t test, and the reason is to stand on the significance of the obtained results. Moreover, the experiments performed on the largest datasets reveals the method ability to learn from small proportion of the data.

CONCLUSION AND FUTURE WORK

The main objective of this study was to design, implement and evaluate classification system based on randomized conceptual coverage, with exceptions to produce a model that able to predict with acceptable and comparable performance. Therefore, we started from the state of the art of machine learning by focusing on the ensemble learning approaches. Through this study, we discussed various ensemble-learning methods, identifying the objectives and characteristics of each method, and to what extent they contribute to the learning problem. Then, we studied several implementations of ensemble learning using Formal Concept Analysis. Consequently, we designed and implemented our method while taking into account the notion of randomness in machine learning, and then we introduced it to ensemble learning by verifying which ensemble paradigm is suitable with our approach. From the obtained results, we conclude that: our method benefits from random conceptual coverage and randomized parallel ensemble learning, in addition achieved acceptable and comparable performance, and outperforms other methods in certain scenarios.

Future work will involve scaling the algorithm implementation to run in parallel, which will allow additional experiments to be carried on in short period of time, with significantly larger datasets. Additional enhancements might be introduced to the learning algorithm, such as embedding a specialized data reduction method in order to reduce the running time.

REFERENCES

[1] T. Vyas, P. Prajapati, and S. Gadhwal, “A survey and evaluation of supervised machine learning techniques for spam e-mail filtering,” *Proc. 2015 IEEE Int. Conf. Electr. Comput. Commun. Technol. ICECCT 2015*, 2015.

[2] E. Rezk *et al.*, “Conceptual data sampling for breast cancer histology image classification,” *Comput. Biol. Med.*, vol. 89, no. C, pp. 59–67, Oct. 2017.

[3] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, “Machine learning applications in cancer prognosis and prediction,” *Comput. Struct. Biotechnol. J.*, vol. 13, pp. 8–17, 2015.

[4] J. Gama, “A survey on learning from data streams: current and future trends,” *Prog. Artif. Intell.*, vol. 1, no. 1, pp. 45–55, 2012.

[5] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science (80-.).*, vol. 349, no. 6245, pp. 255–260, 2015.

[6] L. Rokach, “Ensemble-based classifiers,” *Artif. Intell. Rev.*, vol. 33, no. 1–2, pp. 1–39, 2010.

[7] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[8] R. Sharma, A. V. Nori, and A. Aiken, “Bias-variance tradeoffs in program analysis,” *Popl*, no. May, pp. 127–137, 2014.

[9] O. Prokasheva, A. Onishchenko, and S. Gurov, “Classification Methods Based on Formal Concept Analysis,” *FDAIR Form. Concept Anal. Meets Inf. Retr.*, pp. 95–104, 2013.

[10] N. Meddouri, H. Khoufi, and M. Maddouri, “Parallel learning and classification

for rules based on formal concepts," *Procedia Comput. Sci.*, vol. 35, no. C, pp. 358–367, 2014.

[11] N. Meddouri and M. Maddouri, "Boosting Formal Concepts to Discover Classification Rules," in *Next-Generation Applied Intelligence*, 2009, pp. 501–510.

[12] Y. Kashnitsky and D. I. Ignatov, "Can FCA-based Recommender System Suggest a Proper Classi er ? 2 Multiple Classi er Systems," *FCA do Artif. Intell. 2014)*, p. 17, 2014.

[13] M. P. Sesmero, A. I. Ledezma, and A. Sanchis, "Generating ensembles of heterogeneous classifiers using Stacked Generalization," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 5, no. 1, pp. 21–34, 2015.

[14] Louisa Lam, "Ensemble Methods in Machine Learning," in *Classifier combinations: implementations and theoretical issues*, 2000, pp. 77–86.

[15] R. E. Schapire, "The Strength of Weak Learnability (Extended Abstract)," *Mach. Learn.*, vol. 227, no. October, pp. 28–33, 1989.

[16] Y. Freund and R. E. Schapire, "Experiments with a New Boosting Algorithm," *Proc. Int. Conf. Mach. Learn.*, pp. 148–156, 1996.

[17] A. Mayr, H. Binder, A. Mayr, H. Binder, and O. Gefeller, "The Evolution of Boosting Algorithms From Machine Learning to Statistical Modelling The Evolution of Boosting Algorithms From Machine Learning to Statistical Modelling ∗," no. March, pp. 1–32, 2014.

[18] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Resampling or reweighting: A comparison of boosting implementations," *Proc. - Int. Conf. Tools*

with Artif. Intell. ICTAI, vol. 1, pp. 445–451, 2008.

[19] P. Bühlmann and T. Hothorn, "Boosting Algorithms: Regularization, Prediction and Model Fitting," *Stat. Sci.*, vol. 22, no. 4, pp. 477–505, 2007.

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

[21] J. H. Friedman, "1999 Reitz Lecture," *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[22] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[23] K. M. Ting and I. H. Witten, "Stacking bagged and dagged models," *Proc. of ICML'97*, no. November 1997, pp. 367–375, 1997.

[24] E. Bauer, R. Kohavi, P. Chan, S. Stolfo, and D. Wolpert, "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants," *Mach. Learn.*, vol. 36, no. August, pp. 105–139, 1999.

[25] P. Derbeko, R. El-yaniv, and R. Meir, "Variance Optimized Bagging 2 Markowitz Mean-Variance Portfolio Optimization," *Lect. Notes Comput. Sci.*, vol. 2430, pp. 60–72, 2002.

[26] Tin Kam Ho, "Random decision forests," *Proc. 3rd Int. Conf. Doc. Anal. Recognit.*, vol. 1, pp. 278–282, 1995.

[27] J. J. Rodrguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: A New classifier ensemble method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 10, pp. 1619–1630, 2006.

[28] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–

259, 1992.

[29] E. Menahem, L. Rokach, and Y. Elovici, “Troika - An improved stacking schema for classification tasks,” *Inf. Sci. (Ny).*, vol. 179, no. 24, pp. 4097–4122, 2009.

[30] D. Fan, S. Stolfo, and P. Chan, “Using conflicts among multiple base classifiers to measure the performance of stacking,” *Proc. ICML-99 Work. Recent Adv. Meta-Learning Futur. Work*, vol. 1, pp. 10–17, 1999.

[31] L. Todorovski and S. Dzeroski, “Combining Multiple Models with Meta Decision Trees,” *Princ. Data Min. Knowl. Discov.*, pp. 69–84, 2000.

[32] A. K. Seewald, “How to Make Stacking Better and Faster While Also Taking Care of an Unknown Weakness,” *Icml*, pp. 554–561, 2002.

[33] Tin Kam Ho, “Random decision forests,” *Proc. 3rd Int. Conf. Doc. Anal. Recognit.*, vol. 1, pp. 278–282, 1995.

[34] B. G. Rudolf Wille, *No Title*. Springer Science & Business Media, 1999.

[35] W. Kahl, M. Winter, and J. N. Oliveira, “Relational and Algebraic Methods in Computer Science: 15th international conference, RAMiCS 2015 Braga, Portugal, September 28 - October 1, 2015 proceedings,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9348, pp. 312–325, 2015.

[36] M. Trabelsi, N. Meddouri, and M. Maddouri, “A New Feature Selection Method for Nominal Classifier based on Formal Concept Analysis,” *Procedia Comput. Sci.*, vol. 112, pp. 186–194, 2017.

[37] E. M. Nguifo and P. Njiwoua, “Using Lattice-Based Framework as a Tool for Feature Extraction,” in *Feature Extraction, Construction and Selection: A Data*

Mining Perspective, H. Liu and H. Motoda, Eds. Boston, MA: Springer US, 1998, pp. 205–218.

[38] J. Poelmans, S. O. Kuznetsov, D. I. Ignatov, and G. Dedene, "Formal concept analysis in knowledge processing: A survey on models and techniques," *Expert Syst. Appl.*, vol. 40, no. 16, pp. 6601–6623, 2013.

[39] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman \& Co., 1990.

[40] A. Jaoua, M. Salah, S. Ben Yahia, and J. M. AL-Ja'am, "Using Fringes for Minimal Conceptual Decomposition of Binary Contexts," *New Math. Nat. Comput.*, vol. 8, no. 3, pp. 385–394, 2012.

[41] F. Ferjani, S. Elloumi, A. Jaoua, S. Ben Yahia, S. Ismail, and S. Ravan, "Formal context coverage based on isolated labels: An efficient solution for text feature extraction," *Inf. Sci. (Ny).*, vol. 188, pp. 198–214, 2012.

[42] N. Meddouri and M. Maddouri, "Adaptive learning of nominal concepts for supervised classification," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6276 LNAI, no. PART 1, pp. 121–130, 2010.

[43] A. Mouakher and S. Ben Yahia, "Anthropocentric visualization of optimal cover of association rules," *CEUR Workshop Proc.*, vol. 672, pp. 211–222, 2010.

[44] S. O. Kuznetsov, "Fitting Pattern Structures to Knowledge Discovery in Big Data," pp. 254–266, 2013.

[45] M. Trabelsi, N. Meddouri, and M. Maddouri, "New Taxonomy of Classification Methods Based on Formal Concepts Analysis.," *Fca4Ai@Ecai*, vol. 1703, no.

August, pp. 113–120, 2016.

[46] M. Maddouri, S. Elloumi, and a Jaoua, "An incremental learning system for imprecise and uncertain knowledge discovery," *Inf. Sci. (Ny).*, vol. 109, no. 1–4, pp. 149–164, 1998.

[47] M. Maddouri and A. Jaoua, "Incremental rule production: Towards a uniform approach for knowledge organisation," in *Proceedings of the 9th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, 1996, pp. 295–304.

[48] S. García, J. Luengo, J. A. Sáez, V. López, and F. Herrera, "A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 734–750, 2013.

[49] L. E. Raileanu and K. Stoffel, "Theoretical Comparison between the Gini Index and Information Gain," *Ann. Math. Artif. Intell.*, vol. 41, no. 1, pp. 77–93, 2004.

[50] D. Dua and E. Karra Taniskidou, "UCI Machine Learning Repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml.

[51] F. Khozeimeh *et al.*, "Intralesional immunotherapy compared to cryotherapy in the treatment of warts," *Int. J. Dermatol.*, vol. 56, no. 4, pp. 474–478, Apr. 2017.

[52] F. Khozeimeh *et al.*, "An expert system for selecting wart treatment method," *Comput. Biol. Med.*, vol. 81, pp. 167–175, 2017.

[53] T. T. Nguyen, M. P. Nguyen, X. C. Pham, and A. W. C. Liew, "Heterogeneous classifier ensemble with fuzzy rule-based meta learner," *Inf. Sci. (Ny).*, vol. 422, pp. 144–160, 2018.

[54] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn.*

Res., vol. 12, pp. 2825–2830, 2012.

[55] Ross A. and Willson V.L, “Independent Samples T-Test,” in *Basic and Advanced Statistical Tests*, SensePublishers, Rotterdam, 2017, pp. 13–16.