Credit Scoring Application: An Ensemble Algorithm Implementation On Decision Tree and K-NN

Name: Raven Thevar Balasubramanium

Coventry Student ID: 8286284

Supervisor Name: Mr. Shahriman

Bachelor of Science (Hons) in Computing, 3+0 in Collaboration with Coventry University

School of Engineering and Technology, INTI International College Penang

22 February 2021

303COM Declaration of Originality

I declare that this project is all my own work and has not been copied in part or in whole from any other source except where duly acknowledged. As such, all use of previously published work (from books, journals, magazines, internet etc.) has been acknowledged by citation within the main report to an item in the References or Bibliography lists. I also agree that an electronic copy of this project may be stored and used for the purposes of plagiarism prevention and detection.

Statement of copyright

I acknowledge that the copyright of this project report, and any product developed as part of the project, belong to Coventry University. Support, including funding, is available to commercialize products and services developed by staff and students. Any revenue that is generated is split with the inventor/s of the product or service. For further information please see www.coventry.ac.uk/ipr or contact ipr@coventry.ac.uk.

Statement of ethical engagement

I declare that a proposal for this project has been submitted to the Coventry University ethics monitoring website (https://ethics.coventry.ac.uk/) and that the application number is listed below (Note: Projects without an ethical application number will be rejected for marking)

Signed: *Raven* Date: 22-February-2021

Please complete all fields.

First Name:	RAVEN
Last Name:	THEVAR BALASUBRAMANIUM
Student ID number	8286284
1st Supervisor Name	MR. SHAHRIMAN
2nd Supervisor Name	MS. VASUKY

Acknowledgement

I would like to thank Mr. Shahriman and Ms. Vasuky for their help and effort in my project.
Ms. Vasuky was very welcoming to me in the beginning and did not hesitate to supervise me.
Mr. Shahriman is very flexible in his time and outreach towards me and also agreed to supervise me once Ms. Vasuky is no longer available.

Unfortunately, I cannot proceed any further with acknowledgements as I do not like to experience overwhelming emotions, thinking I may never see either one of them ever again.

I would also like to thank God for all the near misses during submitting assignments, whereby I made it all the time.

I would also like to thank myself for completing this report. A special thank you to ab3505 too.

TABLE OF CONTENT

Title	Page Number
Declaration	2
Acknowledgement	3
Abstract	4
1.0. Introduction	5
1.1. Problem Statement	6
1.2. Research Questions	7
1.3. Research Objectives	7
1.4. Client, Audience and Motivation	8
1.5. Assumptions and Constraints	9
1.6. Scope of Research	9
1.7. Report Outline	9
2.0 Literature Review	10
2.1 Credit Scoring – An Overview	11 - 13
2.1.1 Machine Learning Models	13
2.1.1.1 Random Forest (RF)	13-14
2.1.1.2 k-Nearest Neighbour (k-NN)	14 - 15
2.1.1.3 Support Vector Machines (SVM)	15 - 16
2.1.1.4 Decision Tree (DT)	16 - 17
2.1.1.5 Logistic Regression (LR)	18 - 19
2.1.1.6 Naïve Bayes (NB)	19 - 20
2.1.1.7 Ensemble Model (Stacking)	20 - 22
2.2. Credit Scoring Applications in Finance	22 - 35
2.3. Summary of Credit Scoring Applications in Finance	36 - 65
3.0 Research Plan	66
3.1. Data Acquisition	67
3.2. Data Pre-processing	68 - 70
3.3. Model Training	70
3.4. Results Evaluation	70 - 71
3.5. Summary	71

4.0 Implementation 72
4.1. Complete R code used in this research 72 - 73
4.2. Code Discussion 74 - 79
5.0 Results and Discussion 80
5.1. Results of Confusion Matrix of Individually Developed Machine Learning Models 80 - 89
5.2. Comparison of Confusion Matrix between Developed Models vs Literature Review Models 90 - 91
5.3. Comparison of Misclassification Rate and Accuracy Metrics between Developed Models vs Literature Review Models 92 - 93
5.4. Results of Accuracy Metrix of n-fold, Repeated Cross-Validation (RepeatCV) of Developed Models with 3 repeats. 94 - 99
5.5. Results of Accuracy Metrix of 10-fold, Repeated Cross-Validation (RepeatCV) of Developed Models with n repeats. 100 - 105
5.6. Comparison of Accuracy Metrix of 10-fold, Repeated Cross-Validation (RepeatCV) of Developed Ensemble Models (Stacking) based on the Two Least Accurate Developed Individual Models vs All Developed Machine Learning Models with 6 repeats 106 - 108
5.7. Discussion 109 - 112
6.0. Reflection 113
7.0. Conclusion 114
7.1 Research Contribution 114
7.2 Future Work 114 - 115
8.0. References 116 - 121
Appendix A – Safe Assign Originality Report 121

Abstract

Credit scoring is one of the most critical aspect in a financial institutions or banks as it supports the financial decision-making processes. It basically uses applicants' historical data and statistical or machine learning techniques to assess the risk associated with an applicant. According to Soui et al. (2019), high quality risk management allows banks to build a robust decision-making system and to reduce losses. Soui et al. (2019) had also mentioned that banking risks are composed of three types of risks such as credit risk, market risk, and operational risk. Out of the three risks, credit risk represents 60 % of the overall threat for banks (Soui et al. 2019). Hence, to address this, financial institutions commonly uses a credit scoring model to examine and foresee the probability of a borrower defaulting on a current credit item (Chen et al. 2020). Armaki et al. (2017) had mentioned that it is important to notice that even a 1 % enhancement on the accuracy of the credit scoring system will significantly increase the profits of banks and other financial institutions. Therefore, this paper is being researched to address the two lowest accurate classification techniques used in a credit scoring application on a German credit data set, and to implement ensemble algorithm which is Stacking on them to indicate if Stacking can improve the overall accuracy of the two lowest credit scoring models which are k-Nearest Neighbour and Decision Tree. The proposed Stacking method had successfully improved the accuracy of the k-Nearest Neighbour and Decision Tree. The Decision Tree ensemble model (stacking) at an accuracy of 0.738 and the k-Neareast Neighbours ensemble model (stacking) at an accuracy of 0.733 has improved the accuracy of the two lowest individually developed models which are k-Nearest Neighbours at 0.71175 & Decision Tree at 0.71025 before using 10-fold, Repeated Cross-Validation technique and also k-Nearest Neighbours at 0.684 & Decision Tree at 0.731 after using 10-fold, Repeated Cross-Validation technique. Stacking is proven to be suitable for ensemble applications involving other base classifiers. This study has successfully showed that the Stacking is a good ensemble method and can be considered by financial institutions to improve their existing base classifier models.

Keywords: Decision Tree; K-NN; German credit data set; Ensemble algorithms; credit scoring model; accuracy

1.0 Introduction

According to Livieris et al. (2018), in today's competitive economy, credit scoring is still one of the most significant and successful operational research techniques used in banking and finance. According to Livieris et al. (2018), it was developed by Fair and Isaac in the early 1960s and corresponds to the procedure of estimating the risk related to credit products which is evaluated using applicants' credentials such as annual income, job status, residential status, and historical data. According to Livieris et al. (2018), credit scoring produces a score which can be used to classify customers into two separate groups which are the "credit-worthy" whom are likely to repay the credit loan, and the "non credit-worthy" whom are rejected due to their high probability of defaulting. According to Livieris et al. (2018), the global financial crisis of 2008, which resulted in the collapse of many financial institutions had caused a ripple effect throughout the economy and had demonstrated the potential large losses when a credit applicant defaults on a loan. Hence, Livieris et al. (2018) had mentioned whereby, the credit scoring systems are of great interest to banks and financial institutions, not only that they must measure credit risk, but also because any small improvement would produce great profits. According to Livieris et al. (2018), many researchers in the past have developed credit-scoring models by exploiting the knowledge acquired from individual and company records of past borrowing and repaying actions gathered by the banks and financial institutions. According to Livieris et al. (2018), in credit scoring, imbalanced datasets frequently occur as the number of non-worthy applicants and is usually much smaller than the number of worthy. Therefore, in order to address this difficulty, ensemble learning methods have been proposed as a new direction for obtaining a better composite global model with more accurate and reliable decisions than can be obtained from using a single model (Livieris et al. 2018). According to Livieris et al. (2018), the basic idea of ensemble learning is to combine two or more diverse prediction models in order to develop a prediction model with improved classification accuracy. According to Livieris et al. (2018), the widespread development of the Internet, vast collections and the adoption of electronic records have led to the occurences of large repositories of labelled and mostly unlabelled data. Most conventional credit-scoring models are based on the individual supervised classifiers (Livieris et al. 2018) and ignores the ensemble algorithms.

1.1 Problem Statement

According to the *Bank Negara Guidelines* (2010), credit risk continues to remain the largest source of risk for development financial institutions (DFIs) in Malaysia and this is due to the fact that a DFI's loan/financing portfolio is typically the largest asset and the major source of revenue. *Bank Negara Guidelines* (2010) had mentioned that prior years has shown that absence of proper management of such risk has resulted in significant losses for a number of DFIs and the consequence of such losses not only disrupts the intermediation function of the DFIs affected, but also imposes large financial burdens on the Government in recapitalising such DFIs. Therefore, effective credit risk management is therefore vital to ensure that a DFI's credit activities are conducted in a prudent manner and the risk of potential DFI failures reduced *Bank Negara Guidelines* (2010).

Bank Negara Guidelines (2010) had stated that each credit proposal should be subject to careful analysis by a credit analyst with expertise which commensurate with the size and complexity of the transaction. According to *Bank Negara Guidelines* (2010), the establishment of specialist credit groups to analyse and approve credits relating to significant product line, types of credit facilities, industry or geographic sectors are recommended. *Bank Negara Guidelines* (2010) had also stated that an effective evaluation process should establish minimum requirements for the information on which the analysis is to be based and minimum financial performance standards or benchmarks in appraising loans/financing for each product or type of industry. As a summary *Bank Negara Guidelines* (2010) had stated that a sound credit risk management process are:- well-defined credit granting criteria and risk measurement and evaluation approaches. Therefore, I would like to focus on this area by developing an ensemble model for a credit scoring application.

1.2 Research Questions

RQ1:

What are the 2 lowest accurate classification techniques used in a credit scoring application?

RQ2:

Does implementing ensemble algorithms on the two lowest accurate classification techniques in a credit scoring application will increase the accuracy of the ensemble model that uses a German credit data set ?

1.3 Research Objectives

Objective 1:

To develop 6 different machine learning which are Random Forest, k-Nearest Neighbour, Support Vector Machine, Decision Tree, Logistic Regression and, Naïve Bayes classification techniques to determine their accuracy and to select the two lowest accurate classification technique using the German credit data set.

Objective 2:

To develop an ensemble model which is Stacking based on the two lowest accurate machine learning models, in order to calculate the accuracy of the ensemble model and compare it with other developed machine learning models which are Random Forest, k-Nearest Neighbour, Support Vector Machine, Decision Tree, Logistic Regression and, Naïve Bayes using the German credit data set.

1.4 Client, Audience and Motivation

The entity that will benefit from this project of mine are financial institutions that offer loans to their client. According to Gicić and Subasi (2018), there are three different types of credit scoring models used in a credit assessment, which are corporate (large enterprises), small to medium enterprises (SMEs), and retail (consumer). As customer credit management is important to banks, and other financial organizations, they must be careful when dealing with customer loans to circumvent any inappropriate decisions that can lead to the loss of opportunity or financial losses (Gicić and Subasi 2018). In addition, an incorrect assessment of customer credibility can significantly affect the stability of the financial organization, hence the labour-intensive customer creditworthiness assessment is time and resource consuming (Gicić and Subasi 2018). To add salt to the wound, this labour-intensive assessment mostly depends on the bank employee. On the other hand, the Basel Committee on Banking Supervision has demanded that all banking and financial companies to implement rigorous credit risk assessment models when giving a loan to a company or an individual (Gicić and Subasi 2018). On the other note, Armaki et al. (2017) had mentioned that it is important to notice that even a 1 % enhancement on the accuracy of the credit scoring system will significantly increase the profits of banks and other financial institutions. Therefore, this research is hoped to deliver a computer-based automatic, and human-less credit assessment that will help to broaden the financial institutions' ability to predict good loans versus bad loans which will assist them in sanctioning loans appropriately. Finally, the plus side of using a credit scoring model helps to deter a financial crisis from getting out of control.

1.5 Assumptions and Constraints

The computer used in this project is an AMD FX-6300 processor with 16 GB of DDR3 RAM. Tests and experiments were run on this computer and the performance of developed models may be affected due to this computer.

1.6 Scope of Research

The first research is to develop six different machine learning models which are Random Forest, k-Nearest Neighbour, Support Vector Machine, Decision Tree, Logistic Regression and, Naïve Bayes to determine the lowest two out of the six.

The second research focuses on the ensemble method, which is Stacking, and to determine if Stacking could improve the overall accuracy of the developed machine learning models.

1.7 Report Outline

In this research, Chapter 1 is the headers of my report, Chapter 2 is the literature review part of this project, Chapter 3 is the research plan of this project, Chapter 4 is the code implementation of my project, Chapter 5 is where results are displayed and discussed, Chapter 6 is my reflection of the project, Chapter 7 is the conclusion of this project while Chapter 8 host references.

2.0 Literature Review

There are quite extensive research on the credit scoring application based on multiple machine learning models and datasets. Hence, the performance of each developed models varies and depends on the type of datasets and machine learning algorithm used. Therefore, a literature review was engaged to gain a better understanding on the research problems. The following Figure 2.0 summarizes the entire flow of the literature review. The first part of the literature review discusses an overview in credit scoring and the machine learning models used in credit scoring which are Random Forest, k- Nearest Neighbour, Support Vector Machines, Decision Tree, Logistic Regression, Naïve Bayes, and Ensemble Model (Stacking). The second part discusses the credit scoring applications in finance. The third part is a summary on what was discussed in the second part.

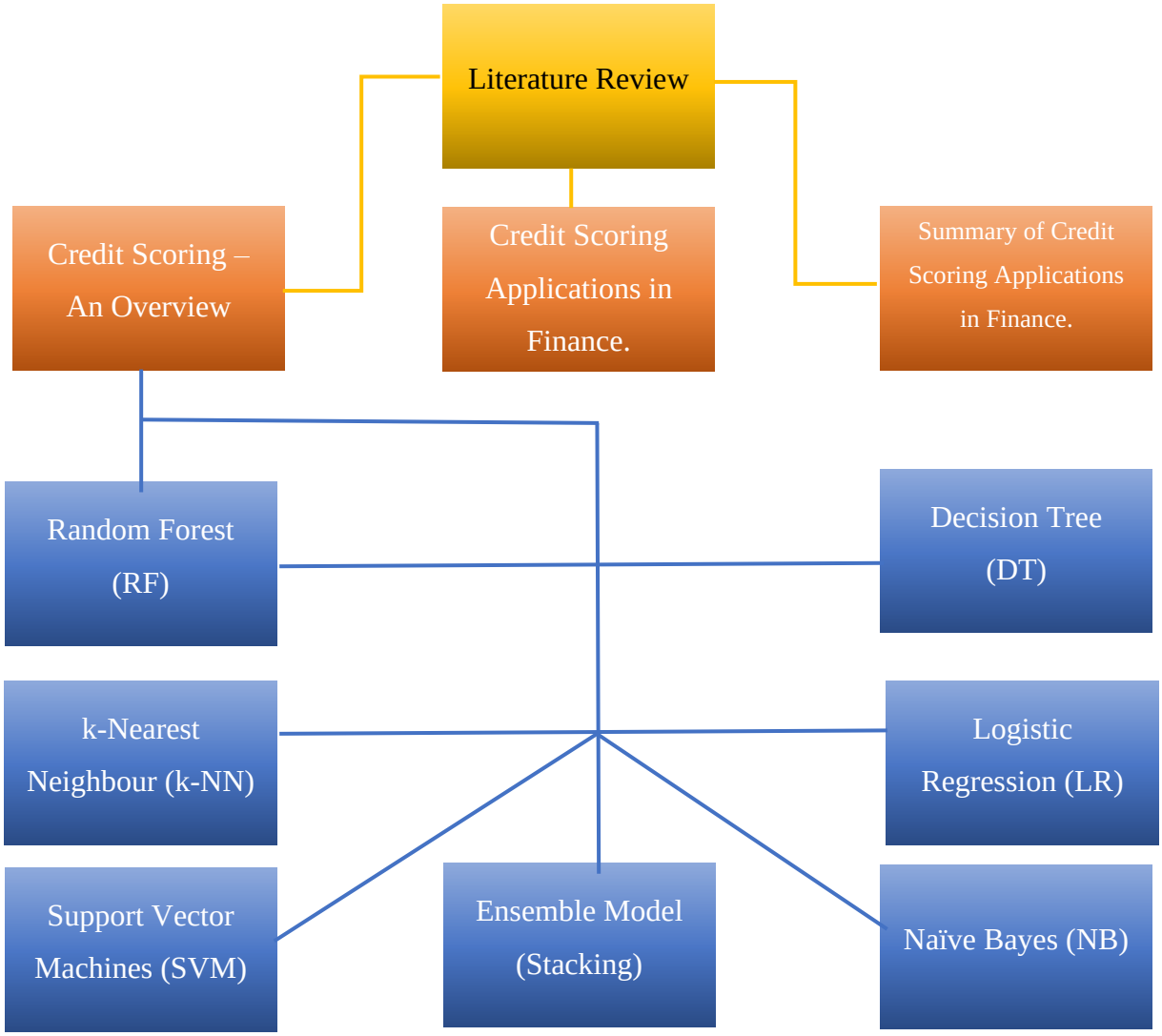


Figure 2.0 :- The Flow of Literature Review for this project

2.1 Credit Scoring – An Overview

According to Gicić and Subasi (2018), a credit scoring model is a model that calculates customers' scores using data mining techniques that requires historical data which relates to credit risk and demographic characteristics. Gicić and Subasi (2018) has also mentioned that in the past, various machine learning techniques have been used for credit scoring, such as Random Forest (RF), k-nearest neighbours (k-NN), Support Vector Machine (SVM), Decision Tree (CART), Logistic Regression (LR) and Naïve Bayes (NB). In addition, Gicić and Subasi (2018) has mentioned that ensembles of classifiers also emerged as novel classification approaches and have since been improved and combined to achieve a higher performance than the single methods. In addition, as mentioned by Gicić and Subasi (2018), ensemble modelling tries to combine a group of learners trained on the same data set and use their ideas to achieve an efficient classification result. Ensemble modelling has substantial computational and financial cost, this yields universal and better classification models for credit scoring (Gicić and Subasi 2018).

According to Masmoudi et al. (2019), many methodologies have been developed (García et al. 2014) such as the statistical method and the artificial intelligence methods (Lessmann et al. 2015; Louzada et al. 2016). The statistical methods include the Logistic Regression (ABID et al. 2017) which are popular credit scoring techniques thanks to their accuracy and easy implementation (Lessmann et al. 2015). To illustrate the artificial intelligence techniques, we can cite the support vector machines (Harris 2015; Tomczak and Zięba 2015), Decision Trees and Naïve Bayes. A Naïve Bayes (NB) model is a graphical representation of a probabilistic model that encodes a set of conditional independence relationships. It has become a popular tool for decision making systems in various fields such as biology, computer science and finance (ABID et al. 2017). Indeed, the NBs are one of the most comprehensive and consistent formalisms for the acquisition and modelling of complex systems outperforming the Logistic Regression in terms of diagnostic prediction. Based on credit worthiness, the authors of Abid et al. (2017) used a discrete NB model for personal loans prediction and classification. Conditional relationships between the factors affecting the credit risk were set up and used to the calibrated conditional probability tables to analyse the payment default causes and effects (ABID et al. 2017).

According to Armaki et al. (2017), there are several approaches employed by financial institutions over the past decades to model the credit risk which are mainly classified into two groups of statistical and artificial intelligence (AI) techniques. Generally, the statistical methods include Logistic Regression (LR) and on the other hand, AI approaches mainly comprise of machine learning techniques such as Support Vector Machines (SVM), and Decision Trees (DT) while other machine learning techniques divided into 2 categories: - classification and clustering algorithms (Armaki et al. 2017). As mentioned by Armaki et al. (2017), there are some pros and cons associated with these machine learning methods as for instance, LR is used for forecasting on a dataset with binary outcomes while Decision Trees are used for classification outcomes. Although, the normality assumption is not required by LR, but linear relationship among variables is a basic assumption for both of these models (Armaki et al. 2017). Therefore, researchers are having doubts about predictive performance of these models for credit scoring (Armaki et al. 2017). In contrast, artificial intelligence techniques had recently drawn attention from many scholars for coping with credit scoring problems (Armaki et al. 2017). These techniques are best known for their higher predictive accuracy compared to statistical models and usually do not require the above-mentioned assumptions. For example, artificial intelligence techniques simulates the human brain's mechanism on the computer environment does not need any assumptions and in the field of credit scoring, and it performs much better than its classical rivals including LR (Armaki et al. 2017). In general, it can be said that AI methods are superior to traditional ones and in recent years, many researchers have focused on the development of machine learning techniques for credit scoring applications (Armaki et al. 2017). One of the methods that they are using to improve the performance machine learning algorithms is the hybridization (ensembles) as researchers believe that the credit scoring models which are built by combing classification (supervised learning) and clustering (unsupervised learning) techniques have the ability to outperform sole machine learning methods.

According to Armaki et al. (2017), classification or supervised learning methods are capable of mapping input vectors into one of various preferred output classes through learning by examples. Armaki et al. (2017) had also mentioned that classifier can be learned by computing the rough distance between input and output instances and correctly labelling outputs out of the training set which is named as the

model generation stage. According to Armaki et al. (2017), after generating the model, the resulting classifier will be able to classify an unidentified example based on the learned classes in the training set. Therefore, the list of classification or supervised learning methods are Random Forest, Decision Tree, k-Nearest Neighbours, Logistic Regression, Naïve Bayes and Support Vector Machines.

According to Armaki et al. (2017), clustering or unsupervised learning methods can be viewed as the way towards combining similar examples into a cluster. Armaki et al. (2017) had mentioned that unlike the classification, labelled examples are not available in clustering and the main aim of clustering approach is to increase the resemblance between the group members. Moreover, the data between different clusters should have the highest dissimilarity and conversely, the highest data similarity should exist within each cluster (Armaki et al. 2017).

2.1.1 Machine Learning Models

Under this section, the machine learning models that will be discussed are Random Forest (RF), k-Nearest Neighbour (k-NN), Support Vector Machines (SVM), Decision Tree (DT), Logistic Regression (LR), Naïve Bayes (NB) and, Ensemble Model (Stacking).

2.1.1.1 Random Forest (RF)

According to Chen (2018), a random forest classifier is a specific type of bootstrap aggregating bagging. In fact, it is slightly different from bagging as they select both a random subset of the features called as “feature bagging” and a random subset of samples in the learning process (Chen 2018). According to Chen (2018), the stronger the predictors for the dependent variable, the more times these features would be selected in the feature bagging progress. According to Chen (2018). similar to bagging, the bootstrapping procedure achieves better performance because it decreases the variance of the model without increasing the bias. In addition, random forests can avoid the overfitting to the training dataset (Chen 2018). According to Chen (2018), similar to the CV error in the bagging model, the out-of-bag (OOB) error of random forest are commonly used to measure the prediction ability of random forests. Besides, to determine the optimal numbers of trees for the whole

model, the OOB error for random forest with different number of trees are calculated and the results are analysed (Chen 2018). For the same purpose of avoiding overfitting, Chen (2018), chose to generate 200 trees to establish the random forest model, which was reasonably accurate enough according to Chen (2018). According to Gicić and Subasi (2018), Random Forest is an ensemble of independent decision trees and this model is created by voting among trees. At each node, a random number of attributes is chosen for splitting (Gicić and Subasi 2018).. The number of attributes is generally much smaller than the number of attributes itself and up to log2d+ 1 attributes are chosen, where d is the number of tuples (Gicić and Subasi 2018). Random Forest is a type of Bagging algorithm with the difference being that Random Forest makes the random selection of attributes whereas Bagging makes splits based on a complete set of attributes (Gicić and Subasi 2018).

2.1.1.2 k-Nearest Neighbour (k-NN)

According to Gicić and Subasi (2018) and Wang et al. (2020), the k-nearest neighbour concept was proposed in the 1950s, but due to slow calculations, intensive use did not occur immediately, and this approach is a very simple method in which classification is based on comparing test set examples to the trained test set. As mentioned by Gicić and Subasi (2018), a test instance is defined by n attributes which searches for the closest pattern in terms of a distance metric. Gicić and Subasi (2018) also said that defining the best k attribute is crucial for good performance, and it is achieved experimentally by minimizing the error rate. Attribute k will be higher for a larger number of training instances (Gicić and Subasi 2018). According to Gicić and Subasi (2018), problems with slow execution are solved by the introduction of a partial distance method, where further computation of the distance is stopped after reaching a certain threshold, and then the process continues with the next instance. According to Wang et al. (2020), the pros of the method is that the training phase of KNN is much faster than other classifiers, and it is pretty intuitive and simple since it does not contain any assumption. KNN can also be useful in cases conducting multi-class problem and regression problem (Wang et al. 2020). The weakness of this method though is that the testing phase of K-NN classification is costlier and slower as it requires large memory for storing entire training datasets. Finally, this method is unable to deal with missing value problem

as well and is sensitive to magnitudes and outliers because it chooses the neighbours on the basis of the distance (Wang et al. 2020). On the other hand, high magnitudes features will weigh more than low magnitudes features (Wang et al. 2020). KNN is also a lazy learning method as lazy algorithm implies that all training data sets are used in the testing phase. According to Dastile et al. (2020), a k-Nearest Neighbour assigns to an input feature vector x the class of the majority of its k nearest neighbours in the training dataset. According to Dastile et al. (2020), the nearest neighbours are determined by calculating the Euclidean distance or Mahalanobis distance between the input feature vector x and the training dataset.

2.1.1.3 Support Vector Machines (SVM)

According to Gicić and Subasi (2018), SVM is a machine learning method for solving classification problems of linear and non-linear data. The main principle of the algorithm is to find an appropriate hyperplane for the class distinctions and in the case of linearly separable data, a hyperplane would be a linear decision function (Gicić and Subasi 2018). Support vectors are used to determine the maximum margin between two classes, which defines the optimal hyperplane (Gicić and Subasi 2018). According to Gicić and Subasi (2018), for linearly non separable data, input data are mapped into the higher dimensional space using non-linear mapping and the next step is to find the linear separating hyperplane in the new space that corresponds to the non-linear separating hypersurface in the original space. Not only that, as mentioned by Gicić and Subasi (2018), even with the small number of support vectors compared with the size of a training set, an effective optimal hyperplane can be produced, and it is also effective with the infinite dimensional space. According to Gicić and Subasi (2018), SVM is also used for various purposes, such as handwritten digit recognition, object recognition, speaker identification, and benchmark time-series prediction tests. According to Ala'raj and Abbod (2016a), SVM is a powerful machine-learning technique used in classification and credit-scoring problems and has been widely used in the area of credit-scoring and other fields owing to its superior results. SVM adopts the form of a linear classifier and takes a set of two classes of given inputs and predict them in order to determine which of the two classes, namely good or bad, has the output (Ala'raj and

Abbod 2016a). According to Chou et al. (2017), the advantages of SVM are it yields global optimal solution as the problem gets converted to a quadratic programming problem, Its solution is always unique and globally optimal, it works well with few samples, it employs the structural risk minimization principle (SRM), which has been shown to be superior to traditional empirical risk minimization principle (ERM) and, it is very resistant to the over-fitting problems. On the other side, the disadvantages of SVM are selection of kernel and its parameters is a tricky issue, it requires extensive memory which has high algorithmic complexity, and is abysmally slow in test phase, it shares the characteristics of transparency and transportability with neural network models and, traditional SVM does not make efficient use of both labelled training data and unlabelled testing data (Chou et al. 2017).

2.1.1.4 Decision Tree (DT)

According to Gicić and Subasi (2018) and Wang et al. (2020), C4.5 is a decision tree algorithm that was developed as an improvement of the ID3 algorithm and this algorithm is based on the principle of information entropy that solves the problem related to training data that has features with large numbers of values. Gicić and Subasi (2018) has mentioned that decision trees are models where the leaf nodes are classification, inner nodes represent attributes, and branches are conjunctions of attributes that lead to classification. If trained set S = (s1,s2,..., sn), then si represents vectors si =(x1,x2,..., xm,ci), and ci is the class where the vector and x1,x2,..., xm are predictive attributes that split the data (Gicić and Subasi 2018). According to Gicić and Subasi (2018), deciding within the tree is based on the importance of the attribute in splitting the set into subsets and all the samples found in the subset are from the same class, and the leaf node for that decision tree is created to choose that class. CTC is the algorithm created to improve the performance of C4.5, and contrary to bagging and boosting, it is formed by one tree instead of the set of trees (Gicić and Subasi 2018). According to Gicić and Subasi (2018), it is used to resolve classification problems related to highly imbalanced data without losing the explaining capacity, which is an important characteristic of single decision trees and rule sets. As mentioned by Gicić and Subasi (2018), the CTC algorithm is based on resampling the training sample that builds the tree from each subsample, but the single tree is made by consensus among the trees of what is

significantly different from Bagging. By using the base classifier, all trees are made from another subsample and propose a variable that will be used to split the current node, which is agreed upon among the trees using voting (Gicić and Subasi 2018). This process is repeated iteratively as long as the stopping condition is not fulfilled (Gicić and Subasi 2018). As mentioned by Gicić and Subasi (2018), induced from different training samples, consolidated trees are more stable and less complex than C4.5 trees. The CTC algorithm can use under sampling to change the class distribution without loss of information, building more accurate classifiers than C4.5 (Gicić and Subasi 2018). According to Chen (2018) and Wang et al. (2020), decision tree is helpful in variable screening and feature selection, and it has the capability of accurately handling high dimensional data with requiring little users' data preparation. In contrast to other models, this method requires little data cleaning since it is not affected by outliers and decision tree can also deal with both quantitative and qualitative data. The weakness of decision tree is that it does not have any stability as small variation in the dataset might lead to a completely different decision tree being generated and calculations will be complex is there are many class labels (Wang et al. 2020). According to Chen (2018), a tree is built or trained to predict the value of dependent variable with several independent variables (namely features) and each node in the tree represents an input variable, and edges to children nodes imply possible values of that independent variable. The dependent variables are divided into leaf nodes, and the path from the root to the leaf is a predictive progress (Chen 2018). A tree is learned by splitting the data set into subsets based on specific criterion and this process is repeated on each subset in a recursive manner called recursive partitioning (Chen 2018). Several decision-tree algorithms have been proposed for tree learning, including ID3, C4.5, CART, MARS and so on (Chen 2018). However, trees do not tend to be as accurate as other machine learning methods, and a single tree is very non-robust and stable (Chen 2018).

2.1.1.5 Logistic Regression (LR)

According to Harris (2015), logistic regression has been widely used in credit scoring applications in various disciplines worldwide and has become one of the most important statistical techniques used for this purpose and it is an important tool used in social science research for categorical data analysis. Harris (2015) had also mentioned that in a credit scoring application, a customer's default probability is a function of certain social-economic variables such as income, marital status and others, in relation to the behavioural characteristics of the customer as indicated by their default history. Thus, this can be view as a binary classification problem where logistic regression is a suitable method (Harris 2015). In comparison to discriminant analysis, logistic regression requires less assumptions and in fact, the only restrictions when using the logistic technique is that the output variable must be binary and that there is no multicollinearity among predictor variables (Harris 2015). According to Harris (2015), logistic regression can be seen as a special case of linear regression and the difference between it from linear regression is that with logistic regression, the outcome variable is dichotomous, while with linear regression this value can be any real number. According to Qu et al. (2019), logistic regression (LR) is essentially a linear model, using a sigmoid function f(x) = 1/(1 + $ee{-}xx$) to conduct the classification and the output is between 0 and 1, which enables the LR to have the capacity of probabilistic interpretation. According to Dastile et al. (2020), the logistic regression is very common in being used as a statistical model in credit scoring due to interpretability of its decisions. According to Dastile et al. (2020), the LR model is defined as the following:-

$$P(y = +1|\mathbf{x}) = \frac{1}{1 + \exp(\alpha_0 + \boldsymbol{\alpha}^T\mathbf{x})}$$

and

$$P(y = -1|\mathbf{x}) = 1 - P(y = +1|\mathbf{x}) = \frac{\exp(\alpha_0 + \boldsymbol{\alpha}^T\mathbf{x})}{1 + \exp(\alpha_0 + \boldsymbol{\alpha}^T\mathbf{x})}$$

where x ∈ Rn is the feature vector, P(y = +1|x) is the probability of classifying x as a good borrower, P(y = −1|x) is the probability of classifying x as a bad borrower and {α0, α} are the model parameters estimated by using, for example, maximum likelihood estimation on the training dataset (Dastile et al. 2020). Dastile et al. (2020) had also mentioned that, once the model parameters are estimated, the decision on

an input feature vector x is made in favour of ˆy = +1 if P(y = +1|x) ≥ P(y = −1|x), which is equivalent to the following decision rule:-

$$\hat{y} = \begin{cases} +1 & \text{for } 1 \geq \exp(\alpha_0 + \boldsymbol{\alpha}^T \mathbf{x}); \\ -1 & \text{otherwise.} \end{cases}$$

2.1.1.6 Naïve Bayes (NB)

According to Ala'raj and Abbod (2016a), Naïve Bayes classifiers are statistical classifiers that predict a particular class (good or bad) of a loan. Ala'raj and Abbod (2016a) had also mentioned that the Bayesian classification is based on Bayesian theory and is a valuable measure when the input feature space is high. This is considered as being a very simple method for making classification rules that are more accurate than those made by other methods; however, it has received very little attention in relation to the credit-scoring domain (Ala'raj and Abbod 2016a). According to Ala'raj and Abbod (2016a), an NB classifier is calculated using the posterior probability of a class by multiplying the prior probability of a class before seeing any data with the likelihood of the data given its class. As mentioned by Ala'raj and Abbod (2016a), for example, in the credit-scoring context, the assumption can be made that the training sample set D = {x 1 , x 2 ,….,x n } , where each x is made up of n characteristics or attributes { x 11 , x 12 ,…., x 1n } and assisted with a class label c , which is either a good or bad loan. Therefore, the task of the NB classifier is centred on analysing these training set instances and determining a mapping function f: (x 11 ,….,x 1n } - > (c) , which can decide the label of an unknown example x = (x 1 ,….., x n) (Ala'raj and Abbod 2016a). According to Dastile et al. (2020), the adjective ‘‘naive’’ comes from the assumption that the features in a dataset are mutually independent. Hence, the Naïve Bayes classifier decides for class y = +1 over y = −1 if P(y = +1|x) ≥ P(y = −1|x),

where

$$P(y = +1|\mathbf{x}) = \frac{p(\mathbf{x}|y = +1)P(y = +1)}{p(\mathbf{x})}$$

and

$$P(y = -1|\mathbf{x}) = \frac{p(\mathbf{x}|y = -1)P(y = -1)}{p(\mathbf{x})}$$

According to Dastile et al. (2020), the probability density p(x) of observing x is defined according to p(x) = p(x|y = +1)P(y = +1) + p(x|y = −1)P(y = −1) and the conditional density functions p(x|y = −1) and p(x|y = +1) are modelled according to

$$p(\mathbf{x}|y=-1) = \frac{1}{(2\pi)^{\frac{m}{2}}|\boldsymbol{\Sigma}^{(-)}|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}^{(-)})^T \mathbf{inv}\left(\boldsymbol{\Sigma}^{(-)}\right)(\mathbf{x}-\boldsymbol{\mu}^{(-)})\}$$

and

$$p(\mathbf{x}|y=+1) = \frac{1}{(2\pi)^{\frac{m}{2}}|\boldsymbol{\Sigma}^{(+)}|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}^{(+)})^T \mathbf{inv}\left(\boldsymbol{\Sigma}^{(+)}\right)(\mathbf{x}-\boldsymbol{\mu}^{(+)})\},$$

where Σ(−) and Σ(+) are covariance matrixes computed on negative and positive instances from the training dataset. For the Naïve Bayes classifier, Σ(−) and Σ(+) are diagonal matrices (Dastile et al. 2020). According to Dastile et al. (2020), p(x|y = −1) and p(x|y = +1) are uni-modal multivariate Gaussian distribution density functions which may not adequately model multi-modal data. In this case, one can employ multi-variate Gaussian Mixture Model and Expectation Maximization (EM) to learn the model parameters on the training dataset (Dastile et al. 2020).

2.1.1.7 Ensemble Model (Stacking)

According to Armaki et al. (2017), stacking is designed to enhance predictive performance through combining the predictions of several machine learning algorithms. Stacking consists of training a combiner algorithm to merge the predictions of various learning algorithms and in other words, stacking trains a set of classifiers in parallel and then the learning is done by a meta-learner (Armaki et al. 2017). Armaki et al. (2017) had also mentioned that stacking can be considered as a fully customizable hybrid machine learning system as it hosts various types of base and meta-classifiers. Stacking is also superior to other types of ensemble methods (Armaki et al. 2017; Zhang et al. 2019). According to Zhang et al. (2019), stacking can be used to form an ensemble model by combining several different classifiers in order to leverage their respective advantages, for example, in a two-layer stacking method, several different classifiers are trained in the first layer, and then the prediction results of the first layer are used as new explanatory features of stacked classifier in the second layer. Finally, the prediction results of stacked classifier are

used as final prediction results. According to Zhang et al. (2020), in a stacking-based ensemble models, the performance of the base classifiers is positively related to the performance of the entire ensemble model. According to Yan and Han (2018), stacking algorithm has a two-level structure consisting of Level 0 classifiers and Level 1 classifiers and it involves two steps. Yan and Han (2018) had mentioned that the first step is to collect the output of each model into a new set of data and for each instance in the original training set, the dataset represents every model's prediction of that instance's class and the models are base classifiers. As for the second step, based on the new data and true labels of each instance in the original training set, a learning algorithm is employed to train the second-layer model (Yan and Han 2018). According to Yan and Han (2018), the formula behind stacking is D= {(Xn,Yn) , n = 1,...,N}, where Xn the instance n and Yn is a vector representing the attribute values of is the class value. All N instances are randomly split into j equivalent parts and j-fold cross-validation is used to train the model (Yan and Han 2018). The prediction results of every time testing set gives a vector which includes n instances: {y1,y2,...,yn}. Set Mj is the difference base learning algorithm model obtained with the data set D, j = 1,...,k,and Mj is the part of Level 0 models (Yan and Han 2018). Level 0 layer consists of j base classifiers, which are employed to estimate the class probability of each instance (Yan and Han 2018). In Mj, j-fold cross-validation of training data set D gives the prediction result: {y(j) 1 ,y(j) 2 ,...,y(j) n }, j = 1,...,k. All the Mj prediction results generate input space of Level 1 model and the real value of instance is treated as the output space (Yan and Han 2018). The model is expressed as follows:

$$
\begin{array}{cc}
\text{Input} & \text{output} \\
y_1^{(1)}, y_1^{(2)}, \ldots, y_1^{(i)}, \ldots, y_1^{(k)}, & y_1 \\
y_2^{(1)}, y_2^{(2)}, \ldots, y_2^{(i)}, \ldots, y_2^{(k)}, & y_2 \\
\vdots & \vdots \\
y_n^{(1)}, y_n^{(2)}, \ldots, y_n^{(i)}, \ldots, y_n^{(k)}, & y_n.
\end{array}
$$

According to Yan and Han (2018), the above intermediate data are considered as the training data of the Level 1 layer model. The input data are treated as features and the real value of instance is treated as the output space (Yan and Han 2018). According to Yan and Han (2018), the next step is to train the data with some fused leaning algorithms and the process is called Level 1 generalizer and Level 1 model is

denoted by $\tilde{M}$,which can be regarded as the function of (y(1),y(2),...,y(i),...,y(k)). In the stacking process, Level 0 (Mj, j = 1,...,k) is combined with Level 1 $\tilde{(M)}$. Given a new instance x',models Mj produce a prediction result vector: y(1),y(2),...,y(i),...,y(k) i = 1, ...,k. Then, $\tilde{M}$ model is used to combine base classifiers and predict the final result of x` (Yan and Han 2018).

2.2 Credit Scoring Applications in Finance.

Authors Louzada et al. (2016) have conducted a research review with the title of *"Classification Methods Applied To Credit Scoring: Systematic Review And Overall Comparison"*. This research review was conducted on the Australian and German datasets, which consist of samples sizes of 690 for the Australian dataset and 1000 for the German dataset. The Australian dataset has 690 instances, with 6 continuous explanatory variables and 8 categorical explanatory variables. The German dataset has 1000 instances, with 7 continuous explanatory, 13 categorical attributes and all the explanatory variables' values and names are encrypted by symbols. The authors had reviewed on the utilization of the machine learning methods of Neural Networks, Hybrid, Combined, Support Vector Machine, Logistic Regression, Trees, Bayesian Net, Linear Regression, Genetic, Fuzzy and, Discriminant Analysis. The authors of this research had proposed their results whereby the highest used machine learning models in a credit scoring application is Logistic Regression which stands at 23.4 %, which is followed by Neural Networks at 21 %. The authors had also mentioned that, the highest objective of different research papers in a credit scoring analysis is to propose a new method for credit rating. Besides, the authors went further to explain that the hybrid machine learning technique is the most common when it comes to new proposition of new method for credit rating, which stands at 19.8 %, followed by combines at 14.6 %. The authors have also explained that feature selection is almost equal in all research papers meaning that the absence and presence of it are equal. Finally, the authors have mentioned that K-fold cross validation method is the most common in all research papers when it comes to evaluating machine learning models.

Authors ABID et al. (2017) have conducted a research with the title of *"Bayesian Network Modeling: A Case Study Of Credit Scoring Analysis Of Consumer Loans Default Payment".* This research was conducted on the Tunisian commercial bank dataset, which consist of information regarding the households' credits over the period of 2014 – 2015. This data also includes 5893 customers' records whereby 84.49 % of them are good credits while 15.51 % are bad credits. This Tunisian bank data has a demographic variable such as age, gender, profession, and credit variables such as type of credit, allocated amount, credit duration, monthly repayment burden, and outstanding credits. The authors had utilized the machine learning methods of Bayesian Network to detect and reduce the rate of bad credits. The authors of this research had proposed their results whereby bad credits are slightly higher for those under the category of granted housing loans at 58.55 % compared to those that have obtained consumer credits at 41.45 %. Besides, younger households have a higher credit worthiness increase since they have fewer commitments compared to older applicants at 76 % and 88 % respectively. Not only that, loans that are lower than 5000 dinars constitute a 52.62 % of bad credits, 88 % of bad credits exist from customers who have been granted loans ranging from 5,000 to 30,000 dinars. Finally, 52.68 % of bad debtors have monthly repayment burden. The authors of this research had mentioned that Bayesian Network model has enabled them to identify the probable stated of borrowers having bad credits and based on this, they were able to build a decision support system that aims to distinguish between good and bad borrowers.

Authors Ala'raj and Abbod (2016a) have conducted a research with the title of *"A New Hybrid Ensemble Credit Scoring Model Based On Classifiers Consensus System Approach"*. This research was conducted on the German, Australian, Japanese, Iranian, Polish, Jordanian, UCSD datasets. The authors had described that the datasets of the German, which has 1000 instances & 20 attributes, Australian which has 690 instances & 14 attributes, and Japanese which has 690 instances & 15 attributes were all used to provide extra validation. The Iranian dataset which has 1000 instances & 27 attributes, consist of corporate client data from a small private bank in Iran. The Polish dataset which has 240 instances & 30 attributes contains information on bankrupted Polish companies recorded over two years. The Jordanian dataset which has 500 instances & 12 attributes is based on a

historical loan dataset and was gathered from one public commercial bank in Jordan. The UCSD dataset which has 2435 instances & 38 attributes on the other hand matches a reduced version of a database employed in the 2007 Data Mining Contest organised by the University of California San Diego and Fair Isaac Corporation. The authors had utilized the machine learning methods of Neural Networks, Support Vector Machine, Decision Trees, Random Forests, and Naïve Bayes. The authors of this research had proposed their results of German dataset at 76.7 %, 70.5 %, 72.5 %, 74.8 %, and 76.1 % accuracy percentages for the Random Forest, Decision Tree, Naïve Bayes, Neural Networks, Support Vector Machine respectively. The Australian dataset was at 86.7 %, 82.6 %, 80.3 %, 85.9 %, and 85.2 % accuracy percentages for the Random Forest, Decision Tree, Naïve Bayes, Neural Networks, Support Vector Machine respectively. The Japanese dataset was at 86.7 %, 81.7 %, 79.7 %, 85.8 % and 85.8 % accuracy percentages for the Random Forest, Decision Tree, Naïve Bayes, Neural Networks, Support Vector Machine respectively. The Iranian dataset was at 95.1 %, 92.4 %, 92.6 %, 95 %, and 94.8 % accuracy percentages for the Random Forest, Decision Tree, Naïve Bayes, Neural Networks, Support Vector Machine respectively. The Polish dataset was at 76.3 %, 70.1 %, 69 %, 69.8 %, and 74.9 % accuracy percentages for the Random Forest, Decision Tree, Naïve Bayes, Neural Networks, Support Vector Machine respectively. The Jordanian dataset was at 85.5 %, 82.8 %, 81.1 %, 81.5 %, and 83 % accuracy percentages for the Random Forest, Decision Tree, Naïve Bayes, Neural Networks, Support Vector Machine respectively. The UCSD dataset was at 86.2 %, 82 %, 61.4 %, 83.1 %, and 83.1 % accuracy percentages for the Random Forest, Decision Tree, Naïve Bayes, Neural Networks, Support Vector Machine respectively. According to the authors Ala'raj and Abbod (2016a), the Multivariate Adaptive Regression Splines (MARS) is a non-parametric and non-linear regression feature selection technique that takes care of non-linearity, being constructed by assuming results value on the unknown points and then converts the linear model to one that is non-linear. This conversion occurs by creating knots on *t* extremes of arguments using the hinge functions and the advantage of converting the model with these functions is that different combinations of them can form a complex model, which lies in the closest position possible to the real results. As such, this has significantly boosted the accuracy of the Random Forest, Decision Tree, Naïve Bayes. Neural Networks, Support Vector Machine base models.

Authors Ala'raj and Abbod (2016b) have conducted a research with the title of *"Classifiers Consensus System Approach For Credit Scoring"*. This research was conducted based on the German, Australian, Japanese, Iranian and Polish credit scoring datasets from the UCI repository. The authors had utilized the machine learning methods of Neural Networks, Support Vector Machine, Decision Tree, Random Forest, and Naïve Bayes. The authors of this research had proposed their results of German dataset regarding the consensus approach, ACC achieves 77.72 %, AUC achieves 80.23 %, H-measure achieves 30.95 %, and Brier Score achieves 15.77 %. As for the Australian dataset, for the consensus approach, ACC achieves 87.98 %, AUC achieves 94.04 %, H-measure achieves 67.19 %, and Brier Score achieves 9.20 %. As for the Japanese dataset, for the consensus approach, ACC achieves 87.88 %, AUC achieves 93.28 %, H-measure achieves 66.53 %, and Brier Score achieves 9.46 %. As for the Iranian dataset, for the consensus approach, ACC achieves 95.69 %, AUC achieves 77.61 %, H-measure achieves 33.42 %, and Brier Score achieves 5.80 %. As for the Polish dataset, for the consensus approach, ACC achieves 76.81 %, AUC achieves 84.06 %, H-measure achieves 38.69 %, and Brier Score achieves 16.23 %. According to Ala'raj and Abbod (2016b), it is visible from the Iranian and Polish datasets whereby the consensus does better in both datasets, except for AUC and Brier score in the former. Random Forests is a strong classifier and being the best classifier across all the performance measures. The authors had also mentioned that the consensus approach has proven to be a reliable and efficient combination or fusion method when combining heterogeneous classifiers across several performance indicator measures and several dataset distributions. Random Forests is an efficient classifier in credit scoring, achieving better results than all the other base classifiers, including Neural Networks and Support Vector Machine. No traditional combination method attained better results than the best classifier. Weighted voting has emerged as being a good combination technique as it can achieve equal predictive accuracy as the best base classifier which is the Random Forests.

Authors Zhang et al. (2020) have conducted a research with the title of *"A novel multi-stage ensemble model with enhanced outlier adaptation for credit scoring"*. This research was conducted on the Australian, Japanese, German, Polish, Taiwan

and, Creator credit datasets, which consist of samples sizes of 7027 – 10,503 for the Polish, 690 for the Australian & Japanese, 1000 for the German, 30,000 for the Taiwan and, 35,690 for the Creator. The authors had utilized the machine learning methods of XGBoost, GBDT, Adaboost, Random Forest (RF), and LightGBM. The authors of this research had proposed their results whereby for the datasets of Australian, the highest accuracy yielding method is RF which stands at 0.85266, German, the highest accuracy yielding method are RF & LightGBM which both stand at 0.7533, Japanese, the highest accuracy yielding method is LightBGM which stands at 0.86087, Polish(s), the highest accuracy yielding method are XGBoost & LightGBM which stand at 0.95 – 0.99. The authors of this research had mentioned that a novel multi-stage ensemble model with enhanced outlier adaptation that combines the BLOF-based outlier adaptation method, the dimension-reduced feature transformation method, and the stacking-based ensemble learning method with self-adaptive parameter optimization is proposed to achieve good predictive power for credit scoring.

Authors Zhang et al. (2019) have conducted a research with the title of *"A novel multi-stage hybrid model with enhanced multi-population niche genetic algorithm: An application in credit scoring"*. This research was conducted on the Australian, German, Japanese, PPDai & GMSC datasets, which consist of samples sizes of 690 for the Australian, 1000 for the German, 690 for the Japanese, 55,596 for the PPDai and, 120,169 for the GMSC. The authors had utilized the machine learning methods of LDA (Linear Discriminant Analysis), LR (Logistic Regression), SVM (Support Vector Machine), DT (Decision Tree) and, MLP (Multilayer Perceptron Network). The authors of this research had proposed their results whereby for the datasets of Australian, the highest accuracy yielding method is LR which stands at 0.8616, German, the highest accuracy yielding method is SVM which stands at 0.7483, Japanese, the highest accuracy yielding method is LDA which stands at 0.8638, PPDai, the highest accuracy yielding method is LR which stands at 0.8713 and, GMSC, the highest accuracy yielding method is MLP which stands at 0.9339. The authors of this research had mentioned that a novel multi-stage hybrid model is proposed, which combines feature selection and classifier selection, to obtain a superior model in binary classification.

Authors Xiao et al. (2016) have conducted a research with the title of *"Ensemble Classification Based On Supervised Clustering For Credit Scoring"*. This research was conducted on the Australian and German dataset, which consist of sample sizes of 690 for the Australian and 1000 for the German. The authors had utilized the machine learning methods of Bagging-RS, RS-Bagging, DCE-CC and, ECSC. The authors of this research had proposed their results whereby for the Logit as base classifier, the highest average misclassification rate is RS-Bagging for the Australian dataset which stands at 12.58 % and Bagging-RS for the German dataset which stands at 23.05 %. As for the Decision Tree as base classifier, the highest average misclassification rate is DCE-CC for the Australian dataset which stands at 13.31 % and DCE-CC for the German dataset which stands at 24.74 %. As for the SVM as base classifier, the highest average misclassification rate is Bagging-RS for the Australian dataset which stands at 14.18 % and RS-Bagging for the German dataset which stands at 31.09 %. The authors of this research had mentioned that the misclassification rates of Bagging-RS, RS-Bagging, DCE-CC and, ECSC with different kinds of classifiers over 30 runs of cross validation are performed and the number of base classifiers are set at 50, 100 and, 150 in order to investigate the impact of classifiers number on the classification performance. According to the authors of this paper, the proposed ECSC method is robust since it does not need to set the number of base classifiers, which has a non-negligible impact on the overall classification performance.

Authors Tripathi et al. (2019) have conducted a research with the title of *"A Novel Hybrid Credit Scoring Model Based On Ensemble Feature Selection And Multilayer Ensemble Classification"*. This research was conducted on the Australian, Japanese and German datasets, which consist of samples sizes of 690 for the Australian, 690 for the Japanese and, 1000 for the German. The authors had utilized the machine learning methods of QDA (Quadratic Discriminant Analysis), NB (Naïve Bayes), MLFN (Multilayer Feed Forward Neural Network), DTDN, TDNN (Time Delay Neural Network), DT (Decision Tree), SVM (Support Vector Machine) and, KNN (k-Nearest Neighbour). The authors of this research had proposed their results whereby for the dataset of Australian, the highest sensitivity is TDNN which stands at 93.43 % and the highest specificity is also TDNN which stands at 91.77 %. As for the dataset of Japanese, the highest sensitivity is SVM which stands at 93.68 % and the highest

specificity is MLFN which stands at 83.42 %. As for the dataset of German-categorical, the highest sensitivity is KNN which stands at 95.28 % and the highest specificity is SVM which stands at 69.66 %. As for the dataset of German-numerical, the highest sensitivity is KNN which stands at 95.57 % and the highest specificity is SVM which stands at 72.66 %. The authors of this research had mentioned the experimental results had indicated that the features selected are more representative and had improved the performance of QDA, NB, MLFN, DTNN, TDNN, DT, and SVM classifiers in terms of classification performance.

Authors Soui et al. (2019) have conducted a research with the title of *"Rule-Based Credit Risk Assessment Model Using Multi-Objective Evolutionary Algorithms"*. This research was conducted on the Australian and German dataset, which consist of samples sizes of 690 for the Australian and 1000 for the German. The authors had utilized the machine learning methods of NSGA II, MOEA/D, SMOPSO and, SPEA2. The authors of this research had proposed their results whereby for the dataset of German, the highest accuracy is SMOPSO which stands at 92.3 %, the highest AUC is also SMOPSO which stands at 0.90, lowest Type 1 Error is also SMOPSO which stands at 17 and, lowest Type 2 Error is NSGA II which stands at 10. As for the dataset of Australian, the highest accuracy is SMOPSO which stands at 94 %, the highest AUC is also SMOPSO which stands at 0.92, lowest Type 1 Error is NSGA II which stands at 9 and, lowest Type 2 Error is SMOPSO which stands at 7. The authors of this research had mentioned that their model is presented in linguistic rules that can easily be comprehensible by both experts and applicants and the SMOPSO outperforms the other algorithms.

Authors Masmoudi et al. (2019) have conducted a research with the title of *"Credit Risk Modeling Using Bayesian Network With A Latent Variable"*. This research was conducted on the Tunisian Central bank dataset, which consist of 9 variables describing loan contracts granted by several Tunisian banks during the period of 1990-2012. The authors had utilized the machine learning methods of Discrete BN, Decision Tree, Radial SVM and, BN with latent variable. The authors of this research had proposed their results whereby the highest yielding accuracy is BN with latent variable which stand at 94.77 %, the highest yielding precision is also BN with latent variable which stand at 85.31 %, the highest recall is also BN with latent variable

which stand at 81.31 % and, the highest F1 measure is also BN with latent variable which stand at 83.26 %. The authors of this research had mentioned that the obtained BN involved about two classes which have similarities and differences in terms of probability distribution and for both classes, the gender variable seems to have little significant effect on the probability of payment default. The authors have also mentioned that the second similarity lies in the fact that the payment default is more likely to occur when the household is under 35. Finally, the BN with latent variable is proved to be useful as a decision making tool in loans attribution and clients' selection.

Authors He et al. (2018) have conducted a research with the title of *"A Novel Ensemble Method For Credit Scoring: Adaption Of Different Imbalance Ratios"*. This research was conducted on the Australian, Japanese, German, DefaultData, PPDaiData, LC2017Q1Data datasets, which consist of samples sizes of 690 for the Australian dataset, 690 for the Japanese dataset, 1000 for the German dataset, 30000 for the DefaultData dataset, 55596 for the PPDaiData and, 95633 for the LC2017Q1Data dataset. The authors had utilized the machine learning methods of KNN, LR, LDA, SVM, DT, RF, GBDT, AdaB and, XGB. The authors of this research had proposed their results whereby for the dataset of Australian, the highest AUC yielding method is XGB which stands at 0.93008, the highest Hmeasure yielding method is RF which stands at 0.68451, the highest KS yielding method is RF which stands at 0.76748, the highest Fmeasure yielding method is RF which stands at 0.85410, the highest GMean yielding method is RF which stands at 0.86693, the highest Logloss yielding method is DT which stands at 7.02047 and, the lowest AvgR yielding method is RF which stands at 2. As for the dataset of Japanese, the highest AUC yielding method is XGB which stands at 0.92535, the highest Hmeasure yielding method is XGB which stands at 0.66586, the highest KS yielding method is RF which stands at 0.75796, the highest Fmeasure yielding method is RF which stands at 0.88042, the highest GMean yielding method is RF which stands at 0.86223, the highest Logloss yielding method is DT which stands at 6.56995 and, the lowest AvgR yielding method is XGB which stands at 1.83. As for the dataset of German, the highest AUC yielding method is RF which stands at 0.79198, the highest Hmeasure yielding method is RF which stands at 0.32247, the highest KS yielding method is LDA which stands at 0.47191, the highest Fmeasure yielding

method is RF which stands at 0.84957, the highest GMean yielding method is LDA which stands at 0.66144, the highest Logloss yielding method is DT which stands at 11.36339 and, the lowest AvgR yielding method is RF which stands at 2.5. As for the dataset of DefaultData, the highest AUC yielding method is XGB which stands at 0.78562, the highest Hmeasure yielding method is XGB which stands at 0.29888, the highest KS yielding method is XGB which stands at 0.44266, the highest Fmeasure yielding method is LDA which stands at 0.48272, the highest GMean yielding method is LDA which stands at 0.59687, the highest Logloss yielding method is DT which stands at 9.61771 and, the lowest AvgR yielding method are GBDT and RF which stand at 2.33. As for the dataset of PPDaiData, the highest AUC yielding method is XGB which stands at 0.70668, the highest Hmeasure yielding method is XGB which stands at 0.14222, the highest KS yielding method is XGB which stands at 0.30301, the highest Fmeasure yielding method is DT which stands at 0.20848, the highest GMean yielding method is DT which stands at 0.43654, the highest Logloss yielding method is DT which stands at 7.48441 and, the lowest AvgR yielding method is XGB which stands at 3. As for the dataset of LC2017Q1Data, the highest AUC yielding method is XGB which stands at 0.71450, the highest Hmeasure yielding method is XGB which stands at 0.01753, the highest KS yielding method is XGB which stands at 0.33821, the highest Fmeasure yielding method is XGB which stands at 0.99353, the highest GMean yielding method is DT which stands at 0.20956, the highest Logloss yielding method is DT which stands at 1.03953 and, the lowest AvgR yielding method is XGB which stands at 1.17. This study uses BalanceCascade to construct adjustable data subsets by estimating data imbalance ratios. The three stage ensemble model with parameters optimized by PSO algorithm, is used with a stacking approach combining the RF and XGBoost as base classifiers to construct a new ensemble model.

Author Thanawala (2019) have conducted a research with the title of *"CREDIT RISK ANALYSIS USING MACHINE LEARNING AND NEURAL NETWORKS"*. This research was conducted on the Australian and German datasets, which consist of which consist of samples sizes of 690 for the Australian dataset and 1000 for the German dataset. The author had utilized the machine learning methods of KNN, Logistic Regression, Naïve Bayes, SVM (Support Vector Machine), Decision Trees, Random Forest and, ANN (Artificial Neural Network). The author of this research had

proposed their results whereby for the dataset of German, the model that yields the highest accuracy are Logistic Regression and SVM meanwhile the lowest accurate model is Decision Trees. As for the Australian dataset, the model that yields the highest accuracy is Random Forest meanwhile the lowest accurate model is Artificial Neural Network. Besides, the author of this paper has stated that SVM has the highest False Positives, Decision Tree has the highest False Negatives, Naïve Bayes has the lowest False Positives and Random Forest has the lowest False Negatives for the German dataset. As for the Australian dataset, SVM has the highest False Positives, KNN has the highest False Negatives, KNN & Naïve Bayes has the lowest False Positives and SVM has the lowest False Negatives.

Author Harris (2015) have conducted a research with the title of *"Credit Scoring Using The Clustered Support Vector Machine"*. This research was conducted on the German and Barbados datasets, which consist of samples sizes of 1000 for the German dataset and 21117 samples for the Barbados dataset. The Barbados dataset consist of 20 attributes dating from 1997 to 2012. The author had utilized the machine learning methods of K means + LR, LR, CSVM-RBF, K means + SVM-RBF, SVM-RBF, CSVM-linear, K means + SVM-linear, SVM-linear. The authors of this research had proposed their results whereby for the dataset of German, the CSVM-RBF classifier has the highest training accuracy which stands at 84.525, the SVM-linear classifier has the highest test accuracy which stands at 78.700, the LR has the highest BAC accuracy which stands at 71.955 and, the LR has the highest AUC which stands at 70.855. As for the dataset of Barbados, the K-means + SVM-linear classifier has the highest training accuracy which stands at 76.737, the K-means + SVM-linear classifier has the highest test accuracy which stands at 67.004, the SVM-RBF has the highest BAC accuracy which stands at 65.428 and, the CSVM-linear has the highest AUC which stands at 66.139. Author Harris had mentioned that CSVM represents a possible solution to the limitations of the current crop of classifiers used in practice. Prior work has noted that as datasets get large, non-linear approaches become increasingly computationally expensive. Therefore, the quest for more computationally efficient algorithms has intensified in recent year as data scientist seek to discover patterns in datasets of increased sizes and complexity.

Author Chen (2018) have conducted a research with the title of *"The Application Of Tree-Based Model To Unbalanced German Credit Data Analysis"*. The authors had utilized the machine learning methods of Decision Trees, Adaboost, Bagging and, Random Forest. The author of this research had mentioned that when predicting if a customer is trustable or otherwise, a number of problems have been met such as low efficiency and bad accuracy. However, the authors had solved this problem by using an advanced model such as bagging to decision trees. Part of the problem faced by this author is the imbalance between the predictions of good customers and bad customers as the prediction of good customers always are more precise than the prediction of bad ones. Adaboost is better than Decision Tree whereby Bagging achieves an accuracy more than 70 % which defeats Adaboost. Random Forest however, comes in second place in terms of accuracy measure to that of Bagging. The training-to-test (training:test) ratio that was used in this study is 80%:20%.

Authors Armaki et al. (2017) have conducted a research with the title of *"A Hybrid Meta-Learner Technique For Credit Scoring Of Banks' Customers"*. This research was conducted on the Australian, German, Japanese and, Iranian datasets, which consist of sample sizes of 690 for the Australian with 14 attributes, 1000 for the German with 24 attributes, 653 for the Japanese with 15 attributes and, 10000 for the Iranian with 12 attributes. The authors had utilized the machine learning methods of SVMPSO, NN, AMLP, LR, DT, SVM, KNN and, NB. The authors of this research had proposed their results whereby for the datasets of Australian, German, Japanese and Iranian, the highest accuracy yielding method is SVMPSO which stands at 97.54, 93.90, 98.48 and 98.92, respectively. As for the F-measure of datasets of Australian, German, Japanese and Iranian, the highest F-measure is SVMPSO which stands at 97.15, 88.68, 98.62 and 91.54, respectively. The authors of this research had mentioned that the SVMPSO have significant roles in enhancing predictive power of the proposed models and as a result, a best credit scoring model can assist financial institutions to determine an accurate and confident credit decisions in the future.

Authors Nalić et al. (2020) have conducted a research with the title of *"New Hybrid Data Mining Model For Credit Scoring Based On Feature Selection Algorithm And Ensemble Classifiers"*. This research was conducted on the Australian and German

dataset, which consist of 690 and 1000 samples, respectively. The authors had utilized the machine learning methods of GLM, DT, SVM, NB, GLM + DT + SVM + NB, GLM + DT + SVM, GLM + DT + NB, DT + NB + SVM, GLM + DT, GLM + SVM, GLM + NB, DT + SVM, DT + NB and, SVM + NB. The authors of this research had proposed their results whereby for the highest accuracy yielding method is GLM + DT which stands at 87.686928, the lowest Type I Error yielding method is also GLM + DT which stands at 10.740888, the lowest Type II Error yielding method is GLM + DT which stands at 1.572184, the highest sensitivity yielding method is GLM + DT which stands at 57.21624, the highest F-measure yielding method is GLM + DT which stands at 87.686928. The authors of this research had mentioned that GLM + DT has the lowest error rate, so this ensemble hybrid model is the most accurate and efficient in comparison with the other models. It is also shown that GLM + DT has the highest performance with the accuracy of 87.69 %.

Authors Wang et al. (2012) have conducted a research with the title of *"Two credit scoring models based on dual strategy ensemble trees"*. This research was conducted on the Australian and German datasets, which consist of 690 and 1000 samples, respectively. The authors had utilized the machine learning methods of LRA, LDA, MLP, RBFN, DT, Bagging DT, Random Subspace DT, Random Forest, Rotation Forest, RS-Bagging DT, Bagging-RS DT. The authors of this research had proposed their results whereby for the dataset of Australian, the highest average accuracy yielding method is RS-Bagging DT which stands at 88.17 %, the lowest Type I Error yielding method is also RS-Bagging DT which stands at 19.44 % and, the lowest Type II Error yielding method is also RS-Bagging DT which stands at 7.52 %. As for the German dataset, the highest average accuracy yielding method is Bagging-RS DT which stands at 78.52 %, the lowest Type I Error yielding method is RS-Bagging DT which stands at 5.98 % and, the lowest Type II Error yielding method is LDA which stands at 26.67 %. The authors of this research had mentioned that DT is one of the most popular classification algorithms in current use in data mining and machine learning and is not often used in credit scoring as the performance of it is inferior to other methods. Therefore, Bagging-RS DT and RS-Bagging DT is introduced to improve the accuracy of models with the use of DT and as per the results of the experiment, the Bagging-RS DT and RS-Bagging DT can be used as an alternative approach in credit scoring applications.

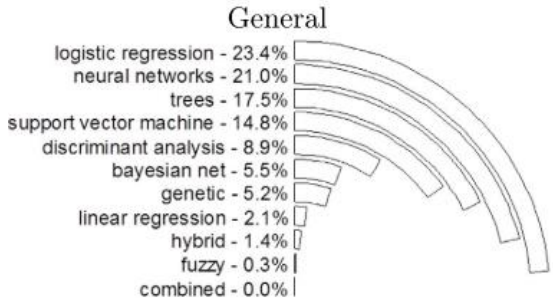
Authors Dahiya et al. (2015) have conducted a research with the title of *"Credit Scoring Using Ensemble Of Various Classifiers On Reduced Feature Set"*. This research was conducted on the German dataset, which consist of 1000 samples and contains 20 predictor variables for each credit applicant while there is one response variable with the class having two values which are good and bad. The variables are either integer or categorical in nature. The authors had utilized the machine learning methods of LR, SVM, CART, CHAID, NN, C5.1, QUEST and, Ensembled. The authors of this research had proposed their results whereby the highest accuracy in the training and test of the 60:40 ratio are SVM and Ensembled at 97.46 %, 71.88 % and 97.46 %, 82.90 %, respectively. As for the highest accuracy in the training and test of the 70:30 ratio is Ensembled at 97.97 % and 83.03 %, respectively. As for the highest accuracy in the training and test of the 50:50 ratio is Ensembled at 98.25 % and 82.56 %, respectively. The authors of this research had mentioned that the LR and SVM performed better than other base classifiers individually and the results were improved overall for almost all individual models including the ensembled model on the 70:30 ratio.

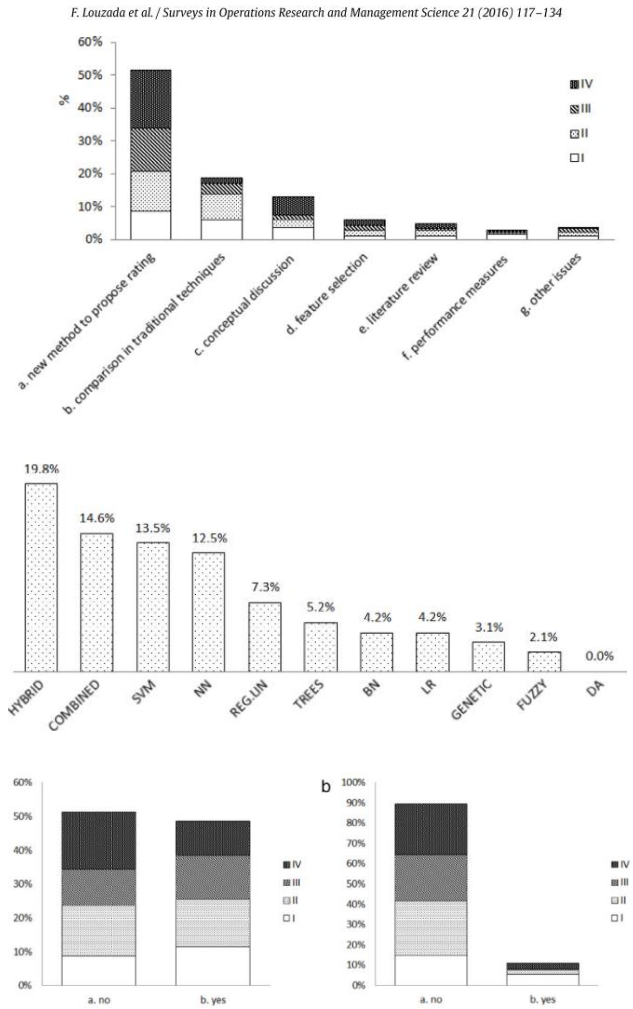
Authors Saidi et al. (2018) have conducted a research with the title of *"Comparison Of Ensemble Cost Sensitive Algorithms: Application To Credit Scoring Prediction"*. This research was conducted on the German dataset, which consist of 1000 instances of which 700 is credit worthy and 300 is not credit worthy. The authors had utilized the machine learning methods of CART, CS-CART, Bagging CART, Bagging CS-CART, CS-Bagging CART, CS-Bagging CS-CART, Multicost CART, Boosting CART, Boosting CS-CART. The authors of this research had proposed their results whereby the lowest Error yielding method are Bagging CART and Boosting CART which stands at 21.93, the highest Specificity yielding method is Bagging CART which stands at 91.87, the highest Sensibility yielding method is CS-Bagging CS-CART which stands at 89.13 and, the lowest Cost yielding method is CS-Bagging CS-CART which stands at 25.71. The authors of this research had mentioned that the ensemble approaches obtain better results than individual classifiers, the insensitive approaches had reached the best classification accuracy, the cost sensitive approaches are intended to reduce the cost at the expense of the accuracy.

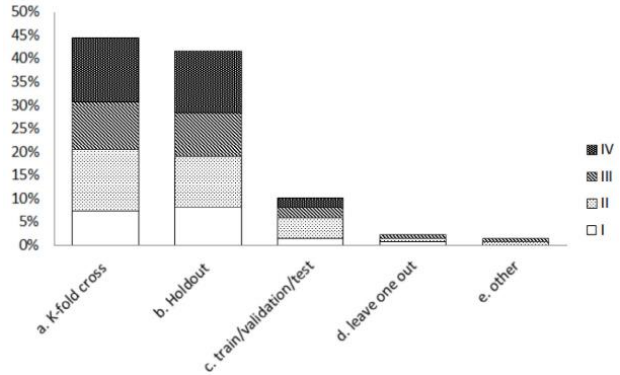
Authors Xiao et al. (2020) have conducted a research with the title of *"Cost-Sensitive Semi-Supervised Selective Ensemble Model For Customer Credit Scoring"*. This research was conducted on the Australian, German, UK-Thomas, PAKDD and, Give-Credit datasets. The authors had utilized the machine learning methods of GCSSE, Subagging, CoBag, Semi-Bagging and, Tri-training. The authors of this research had proposed their results whereby for the dataset of UK-Thomas, the GCSSE had yielded the highest accuracy which stands at 0.7150. As for the German dataset, the GCSSE had yielded the highest accuracy which stands at 0.7376. As for the Australian dataset, the GCSSE had yielded the highest accuracy which stands at 0.8689. As for the PAKDD dataset, the Semi-Bagging had yielded the highest accuracy which stands at 0.7829. As for the Give-Credit dataset, the GCSSE had yielded the highest accuracy which stands at 0.9320. The authors of this research had mentioned that GCSSE is an overall superior model compared to others used in this research and is recommended for credit scoring applications.

2.3 Summary of Credit Scoring Applications in Finance

Table 2.3 Summary of Literature Review

Authors	Datasets	Machine Learning Methods	Feature Selection	Results	Discussion
Louzada, F Ara, A Fernandes, G	Australian German	Neural Networks, Hybrid, Combined, Support Vector Machine, Logistic Regression, Trees, Bayesian Net, Linear Regression, Genetic, Fuzzy, Discriminant Analysis	N/A	 **Table 2** Distribution of reviewed papers according to the journal title in the four time periods. Journal | I | II | III | IV | Total | % Expert Systems with Applications | 9 | 16 | 14 | 13 | 52 | 27.81 Journal of the Operational Research Society | 11 | 5 | 1 | 3 | 20 | 10.70 European Journal of Operational Research | 1 | 6 | 3 | 6 | 16 | 8.56 Knowledge-Based Systems | 0 | 1 | 2 | 4 | 7 | 3.74 Applied Stochastic Models in Business and Industry | 4 | 0 | 0 | 0 | 4 | 2.14 Computational Statistics and Data Analysis | 1 | 0 | 1 | 1 | 3 | 1.60 IMA Journal Management Mathematics | 2 | 1 | 0 | 0 | 3 | 1.60 International Journal of Neural Systems | 0 | 0 | 3 | 0 | 3 | 1.60 Others[a] | 15 | 22 | 15 | 27 | 79 | 42.25 Total | 43 | 51 | 39 | 54 | 187 | 100 [a] These include papers from ACM Trans. on Knowledge Discovery from Data, Decision Support Systems, Journal of the Royal Stat. Society, Inter. Journal of Comp. Intelligence & Applications, Applied Math. & Comp., Applied Soft Computing, Comm. in Statistics, Comp. Statistics, Credit and Banking and others. **Table 3** Distribution of reviewed papers according to the author/co-author in the four time periods. Author | Affiliation, Country | I | II | III | IV | Total | % Baesens, B. | Katholieke Univ. Leuven, Belgium | 7 | 5 | 2 | 2 | 16 | 3.0 Vanthienen, J. | Katholieke Univ. Leuven, Belgium | 6 | 4 | 0 | 0 | 10 | 1.9 Hand, D.J. | Imperial College London, UK | 7 | 0 | 1 | 0 | 8 | 1.5 Thomas, L.C. | University of Southampton, UK | 1 | 2 | 2 | 3 | 8 | 1.5 Mues, C. | University of Southampton, UK | 1 | 2 | 3 | 1 | 7 | 1.3 Van Gestel, T. | Katholieke Univ. Leuven, Belgium | 3 | 4 | 0 | 0 | 7 | 1.3 Tsai, C.-F. | Nat. Chung Cheng University, Taiwan | 0 | 3 | 0 | 3 | 6 | 1.1 Bravo, C. | Universidade de Chile, Chile | 0 | 0 | 0 | 4 | 4 | 0.8 Louzada, F. | Universidade de Sao Paulo, Brazil | 0 | 0 | 3 | 1 | 4 | 0.8 Shi, Y. | University of Nebraska Omaha, US | 0 | 3 | 0 | 1 | 4 | 0.8 Others | | 95 | 107 | 101 | 148 | 451 | 85.9 Total | | 120 | 130 | 112 | 163 | 525 | 100.0	The highest used machine learning models in a credit scoring application is Logistic Regression which stands at 23.4 %, which is followed by Neural Networks at 21 %. The highest objective of different research papers in a credit scoring analysis is to propose a new method for credit rating. Besides, the authors went further to explain that the hybrid machine learning

				 Fig. 10. (a) The using of feature selection and (b) the using of missing data imputation.	technique is the most common when it comes to new proposition of new method for credit rating, which stands at 19.8 %, followed by combines at 14.6 %. The authors have also explained that feature selection is almost equal in all research papers meaning that the absence and presence of it are equal. Finally, the authors have mentioned that K-fold cross validation method is the most common in all research papers when it comes to evaluating

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td> 
</td>
<td>machine learning models.</td>
</tr>
<tr>
<td>ABID, L,

Zaghdene, S

Masmoudi, A

Ghorbel, S</td>
<td>Tunisian Bank</td>
<td>Bayesian Network</td>
<td>NO</td>
<td>
Table-3. Conditional probability distribution of X_1 given the payment default(X_9= 1)
<table>
<tr><td>$P(X_1/X_9)$</td><td>Consumer Credit</td><td>Housing Credit</td></tr>
<tr><td></td><td>0.4144956</td><td>0.5855044</td></tr>
</table>
Source: Author's calculation by using Matlab

Table-4. Conditional probability distribution of X_7 given the payment default(X_9= 1)
<table>
<tr><td>$P(X_7/ X_9)$</td><td>> 50 years</td><td>0- 50 years</td></tr>
<tr><td></td><td>0,76886955</td><td>0,23113045</td></tr>
</table>
Source: Author's calculation by using Matlab

Table-5. Conditional probability distribution of X_3 given the payment default(X_9= 1)
<table>
<tr><td>$P(X_3/X_9)$</td><td>worker</td><td>middle executive</td><td>Senior executive</td><td>retired</td><td>Liberal function</td><td>Senior executive</td><td>retired</td><td>Liberal function</td></tr>
<tr><td></td><td>0.164770</td><td>0.181609</td><td>0.122337</td><td>0.526281</td><td>0.0050</td><td>0.12233</td><td>0.526281</td><td>0.0050</td></tr>
</table>
Source: Author's calculation by using Matlab

Table-6. Conditional probability distribution of X_4 given the payment default(X_9= 1)
<table>
<tr><td>$P(X_4/X_9)$</td><td>< 5000</td><td>5000 << 30000</td><td>30000 << 50000</td><td>> 50000</td></tr>
<tr><td></td><td>0,5230303</td><td>0,44885482</td><td>0,01933639</td><td>0,0087785</td></tr>
</table>
Source: Author's calculation by using Matlab

Table-7. Conditional probability distribution of X_6 given the payment default(X_9 = 1)
<table>
<tr><td>$P(X_6/X_9)$</td><td>0-100 Dinars</td><td>100-200 Dinars</td><td>200-500 Dinars</td><td>500-1000 Dinars</td><td>> 1000Dinars</td></tr>
<tr><td></td><td>0,52686506</td><td>0,19542708</td><td>0,2682677</td><td>0,00410318</td><td>0,00533699</td></tr>
</table>
Source: Author's calculation by using Matlab
</td>
<td>Bad credits are slightly higher for those under the category of granted housing loans at 58.55 % compared to those that have obtained consumer credits at 41.45 %. Younger households have a higher credit worthiness increase since they have fewer commitments compared to older applicants at 76 % and 88 % respectively.</td>
</tr>
</table>

					Loans that are lower than 5000 dinars constitute a 52.62 % of bad credits, 88 % of bad credits exist from customers who have been granted loans ranging from 5,000 to 30,000 dinars. 52.68 % of bad debtors have monthly repayment burden. Bayesian Network model has enabled them to identify the probable stated of borrowers having bad credits and based on this, they were able to build a decision support system that aims to distinguish between good

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>and bad borrowers.</td></tr>
<tr><td>Ala'raj, M

Abbod, M</td><td>German, Australian, Japanese, Iranian, Polish, Jordanian, UCSD</td><td>Neural Networks, Support Vector Machine, Decision Trees, Random Forests, Naïve Bayes</td><td>YES</td><td>
Table 3
Classifier results for the for all individual classifiers for all the datasets for the different performance measures without GNG or MARS.
<table>
<tr><th>Dataset</th><th>Perf. measure</th><th colspan="5">Base classifiers</th></tr>
<tr><th></th><th></th><th>RF</th><th>DT</th><th>NB</th><th>NN</th><th>SVM</th></tr>
<tr><td>German</td><td>Accuracy</td><td>0.767</td><td>0.705</td><td>0.725</td><td>0.748</td><td>0.761</td></tr>
<tr><td></td><td>AUC</td><td>0.792</td><td>0.679</td><td>0.762</td><td>0.764</td><td>0.783</td></tr>
<tr><td></td><td>H-measure</td><td>0.289</td><td>0.136</td><td>0.238</td><td>0.239</td><td>0.272</td></tr>
<tr><td></td><td>Brier Score</td><td>0.162</td><td>0.252</td><td>0.199</td><td>0.172</td><td>0.166</td></tr>
<tr><td>Australian</td><td>Accuracy</td><td>0.867</td><td>0.826</td><td>0.803</td><td>0.859</td><td>0.852</td></tr>
<tr><td></td><td>AUC</td><td>0.936</td><td>0.866</td><td>0.896</td><td>0.915</td><td>0.911</td></tr>
<tr><td></td><td>H-measure</td><td>0.661</td><td>0.515</td><td>0.580</td><td>0.614</td><td>0.602</td></tr>
<tr><td></td><td>Brier Score</td><td>0.095</td><td>0.147</td><td>0.167</td><td>0.108</td><td>0.114</td></tr>
<tr><td>Japanese</td><td>Accuracy</td><td>0.867</td><td>0.817</td><td>0.797</td><td>0.858</td><td>0.858</td></tr>
<tr><td></td><td>AUC</td><td>0.931</td><td>0.856</td><td>0.889</td><td>0.912</td><td>0.907</td></tr>
<tr><td></td><td>H-measure</td><td>0.649</td><td>0.489</td><td>0.567</td><td>0.619</td><td>0.607</td></tr>
<tr><td></td><td>Brier Score</td><td>0.098</td><td>0.157</td><td>0.175</td><td>0.108</td><td>0.114</td></tr>
<tr><td>Iranian</td><td>Accuracy</td><td>0.951</td><td>0.924</td><td>0.926</td><td>0.950</td><td>0.948</td></tr>
<tr><td></td><td>AUC</td><td>0.779</td><td>0.615</td><td>0.714</td><td>0.613</td><td>0.603</td></tr>
<tr><td></td><td>H-measure</td><td>0.276</td><td>0.112</td><td>0.193</td><td>0.062</td><td>0.073</td></tr>
<tr><td></td><td>Brier Score</td><td>0.043</td><td>0.070</td><td>0.076</td><td>0.048</td><td>0.051</td></tr>
<tr><td>Polish</td><td>Accuracy</td><td>0.763</td><td>0.701</td><td>0.690</td><td>0.698</td><td>0.749</td></tr>
<tr><td></td><td>AUC</td><td>0.837</td><td>0.728</td><td>0.740</td><td>0.767</td><td>0.821</td></tr>
<tr><td></td><td>H-measure</td><td>0.384</td><td>0.210</td><td>0.235</td><td>0.250</td><td>0.366</td></tr>
<tr><td></td><td>Brier Score</td><td>0.165</td><td>0.267</td><td>0.297</td><td>0.198</td><td>0.176</td></tr>
<tr><td>Jordanian</td><td>Accuracy</td><td>0.855</td><td>0.828</td><td>0.811</td><td>0.815</td><td>0.830</td></tr>
<tr><td></td><td>AUC</td><td>0.909</td><td>0.795</td><td>0.707</td><td>0.746</td><td>0.789</td></tr>
<tr><td></td><td>H-measure</td><td>0.535</td><td>0.385</td><td>0.176</td><td>0.238</td><td>0.309</td></tr>
<tr><td></td><td>Brier Score</td><td>0.097</td><td>0.143</td><td>0.172</td><td>0.142</td><td>0.128</td></tr>
<tr><td>UCSD</td><td>Accuracy</td><td>0.862</td><td>0.820</td><td>0.614</td><td>0.831</td><td>0.831</td></tr>
<tr><td></td><td>AUC</td><td>0.903</td><td>0.788</td><td>0.574</td><td>0.859</td><td>0.843</td></tr>
<tr><td></td><td>H-measure</td><td>0.513</td><td>0.330</td><td>0.080</td><td>0.398</td><td>0.386</td></tr>
</table>
</td><td>The Multivariate Adaptive Regression Splines (MARS) is a non-parametric and non-linear regression feature selection technique that takes care of non-linearity, being constructed by assuming results value on the unknown points and then converts the linear model to one that is non-linear. This conversion occurs by creating knots on t extremes of arguments using the hinge functions and the advantage of</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>converting the model with these functions is that different combinations of them can form a complex model, which lies in the closest position possible to the real results. As such, this has significantly boosted the accuracy of the Random Forest, Decision Tree, Naïve Bayes. Neural Networks, Support Vector Machine base models.</td></tr>
<tr><td>Ala'raj, M

Abbod, M</td><td>German, Australian, Japanese, Iranian, Polish</td><td>Neural Networks, Support Vector Machine, Decision Tree, Random Forest,</td><td>NO</td><td>
Table 3
Classifier results for the German dataset over the different performance measures.
<table>
<tr><td></td><td colspan="5">Base classifiers</td><td colspan="7">Traditional combination methods</td><td>Proposed</td></tr>
<tr><td>Performance measure</td><td>NN</td><td>SVM</td><td>RF</td><td>DT</td><td>NB</td><td>Min.</td><td>Max.</td><td>Product</td><td>Mean</td><td>Majority voting</td><td>Weighted average</td><td>Weighted voting</td><td>Consensus approach</td></tr>
<tr><td>ACC</td><td>0.7555</td><td>0.7236</td><td>0.7621</td><td>0.7242</td><td>0.7263</td><td>0.7401</td><td>0.7195</td><td>0.7077</td><td>0.7591</td><td>0.7597</td><td>0.7201</td><td>0.7410</td><td>0.7772</td></tr>
<tr><td>AUC</td><td>0.7788</td><td>0.6696</td><td>0.7847</td><td>0.7290</td><td>0.7426</td><td>0.7016</td><td>0.7661</td><td>0.0760</td><td>0.7840</td><td>0.7447</td><td>0.7121</td><td>0.7121</td><td>0.8023</td></tr>
<tr><td>H-measure</td><td>0.2680</td><td>0.1506</td><td>0.2751</td><td>0.1799</td><td>0.2680</td><td>0.1853</td><td>0.2434</td><td>0.1855</td><td>0.2714</td><td>0.2542</td><td>0.1787</td><td>0.1389</td><td>0.3095</td></tr>
<tr><td>Brier score</td><td>0.2440</td><td>0.2847</td><td>0.2269</td><td>0.2276</td><td>0.2440</td><td>0.2675</td><td>0.2376</td><td>0.2918</td><td>0.1652</td><td>0.1857</td><td>0.2221</td><td>0.2464</td><td>0.1577</td></tr>
</table>
</td><td>It is visible from the Iranian and Polish datasets whereby the consensus does better in both datasets, except for AUC and Brier score in the</td></tr>
</table>

		Naïve Bayes		**Table 4** Classifier results for the Australian dataset over the different performance measures. 		Base classifiers					Traditional combination methods							Proposed
Performance measure	NN	SVM	RF	DT	NB	Min.	Max.	Product	Mean	Majority voting	Weighted average	Weighted voting	Consensus approach
ACC	0.8590	0.8133	0.8707	0.8486	0.8058	0.8219	0.8466	0.8034	0.8545	0.8641	0.8622	0.8707	**0.8798**
AUC	0.9278	0.8944	0.9351	0.9111	0.8335	0.8718	0.9132	0.8484	0.9273	0.9196	0.9130	0.9130	**0.9404**
H-measure	0.6419	0.5788	0.6597	0.5813	0.4754	0.5317	0.6079	0.5264	0.6256	0.6349	0.6158	0.6128	**0.6719**
Brier score	0.3669	0.3916	0.0935	0.3218	0.4578	0.2582	0.1571	0.3479	0.1036	0.1036	0.1204	0.1292	**0.0920** | **Table 5** Classifier results for the Japanese dataset over the different performance measures. |	Base classifiers					Traditional combination methods							Proposed
Performance measure	NN	SVM	RF	DT	NB	Min.	Max.	Product	Mean	Majority voting	Weighted average	Weighted voting	Consensus approach
ACC	0.8485	0.8194	0.8697	0.8378	0.7769	0.8396	0.7927	0.8229	0.8557	0.8669	0.8357	0.8664	**0.8788**
AUC	0.8981	0.8773	0.9294	0.9067	0.8179	0.8781	0.8837	0.8795	0.9162	0.9130	0.8705	0.8705	**0.9328**
H-measure	0.5928	0.5533	0.6418	0.5614	0.3711	0.5535	0.5598	0.5744	0.6108	0.6205	0.5311	0.5969	**0.6653**
Brier score	0.3920	0.4491	0.0979	0.3259	0.3955	0.3049	0.1624	0.4091	0.1120	0.1076	0.1378	0.2309	**0.0946** | **Table 6** Classifier results for the Iranian dataset over the different performance measures. |	Base classifiers					Traditional combination methods							Proposed
Performance measure	NN	SVM	RF	DT	NB	Min.	Max.	Product	Mean	Majority voting	Weighted average	Weighted voting	Consensus approach
ACC	0.8188	0.9449	0.9506	0.9496	0.2645	0.9498	0.2382	0.9496	0.9263	0.9490	0.9111	0.9506	**0.9569**
AUC	0.5404	0.7185	**0.7798**	0.7274	0.5427	0.5848	0.5529	0.5357	0.6772	0.6022	0.7407	0.7407	0.7761
H-measure	0.0160	0.2592	0.2833	0.2039	0.0141	0.1386	0.0240	0.1476	0.1297	0.0629	0.2572	0.0673	**0.3342**
Brier score	0.0534	0.066	**0.0437**	0.0558	0.7634	0.0489	0.7628	0.0502	0.0928	0.0747	0.0476	0.0478	0.0580 | **Table 7** Classifier results for the Polish dataset over the different performance measures. |	Base classifiers					Traditional combination methods							Proposed
Performance measure	NN	SVM	RF	DT	NB	Min.	Max.	Product	Mean	Majority voting	Weighted average	Weighted voting	Consensus approach
ACC	0.6169	0.6717	0.7596	0.7573	0.6879	0.7239	0.6852	0.6439	0.7563	0.7575	0.7406	0.7596	**0.7681**
AUC	0.6719	0.7979	0.8322	0.8273	0.7612	0.7983	0.7678	0.8055	0.8261	0.8209	0.8051	0.8051	**0.8406**
H-measure	0.1144	0.3173	0.3735	0.3607	0.2391	0.3215	0.2622	0.3472	0.3585	0.3541	0.3234	0.3091	**0.3869**
Brier score	0.3342	0.3748	0.1672	0.3205	0.3910	0.4048	0.2871	0.4576	0.1721	0.1769	0.2142	0.2401	**0.1623**		former. Random Forests is a strong classifier and being the best classifier across all the performance measures. The authors had also mentioned that the consensus approach has proven to be a reliable and efficient combination or fusion method when combining heterogeneous classifiers across several performance indicator measures and several dataset distributions. Random Forests is an efficient classifier in credit scoring, achieving better results than all

					the other base classifiers, including Neural Networks and Support Vector Machine. No traditional combination method attained better results than the best classifier. Weighted voting has emerged as being a good combination technique as it can achieve equal predictive accuracy as the best base classifier which is the Random Forests.
Zhang, W, Yang, D, Zhang, S, Ablanedo-Rosas, J,	Australian, Japanese, German, Polish, Taiwan, Creator	XGBoost, GBDT, Adaboost, Random Forest (RF), LightGBM	YES	Without BLOF	A novel multi-stage ensemble model with enhanced outlier adaptation that combines the BLOF-based outlier

<table>
<tr>
<td>Wu, X,

Lou, Y</td>
<td></td>
<td></td>
<td></td>
<td>Baseline results.
<table>
<tr><th>Dataset</th><th>Base classifiers</th><th>ACC</th><th>BA</th><th>AUC</th><th>F-score</th><th>Log loss</th><th>Brier score</th></tr>
<tr><td>Australian</td><td>XGBoost</td><td>0.84058</td><td>0.84457</td><td>0.93315</td><td>0.82732</td><td>0.40710</td><td>0.11861</td></tr>
<tr><td></td><td>GBDT</td><td>0.83575</td><td>0.84139</td><td>0.93189</td><td>0.82499</td><td>0.33877</td><td>0.11044</td></tr>
<tr><td></td><td>Adaboost</td><td>0.81159</td><td>0.81500</td><td>0.89806</td><td>0.79671</td><td>0.64093</td><td>0.22400</td></tr>
<tr><td></td><td>RF</td><td>0.85266</td><td>0.85568</td><td>0.94005</td><td>0.84068</td><td>0.32655</td><td>0.10135</td></tr>
<tr><td></td><td>LightGBM</td><td>0.83333</td><td>0.83575</td><td>0.92476</td><td>0.81564</td><td>0.42308</td><td>0.12438</td></tr>
<tr><td>German</td><td>XGBoost</td><td>0.74000</td><td>0.67038</td><td>0.76373</td><td>0.81953</td><td>0.68728</td><td>0.19911</td></tr>
<tr><td></td><td>GBDT</td><td>0.75167</td><td>0.67150</td><td>0.77514</td><td>0.83083</td><td>0.51683</td><td>0.17243</td></tr>
<tr><td></td><td>Adaboost</td><td>0.75167</td><td>0.68694</td><td>0.76533</td><td>0.82696</td><td>0.66043</td><td>0.23370</td></tr>
<tr><td></td><td>RF</td><td>0.75333</td><td>0.67855</td><td>0.77534</td><td>0.83111</td><td>0.50967</td><td>0.16961</td></tr>
<tr><td></td><td>LightGBM</td><td>0.75333</td><td>0.67459</td><td>0.75310</td><td>0.83128</td><td>0.61979</td><td>0.19050</td></tr>
<tr><td>Japanese</td><td>XGBoost</td><td>0.85507</td><td>0.85392</td><td>0.91778</td><td>0.86918</td><td>0.46731</td><td>0.11500</td></tr>
<tr><td></td><td>GBDT</td><td>0.83623</td><td>0.83799</td><td>0.92206</td><td>0.84734</td><td>0.36820</td><td>0.11425</td></tr>
<tr><td></td><td>Adaboost</td><td>0.84493</td><td>0.84320</td><td>0.91039</td><td>0.86031</td><td>0.63754</td><td>0.22232</td></tr>
<tr><td></td><td>RF</td><td>0.84783</td><td>0.84256</td><td>0.90774</td><td>0.86727</td><td>0.38863</td><td>0.11725</td></tr>
<tr><td></td><td>LightGBM</td><td>0.86087</td><td>0.85820</td><td>0.91705</td><td>0.87452</td><td>0.45090</td><td>0.11527</td></tr>
<tr><td>Polish 1</td><td>XGBoost</td><td>0.97127</td><td>0.69017</td><td>0.91788</td><td>0.51203</td><td>0.12574</td><td>0.02584</td></tr>
<tr><td></td><td>GBDT</td><td>0.97027</td><td>0.65510</td><td>0.91088</td><td>0.45043</td><td>0.10254</td><td>0.02563</td></tr>
<tr><td></td><td>Adaboost</td><td>0.96785</td><td>0.64065</td><td>0.91544</td><td>0.40936</td><td>0.60518</td><td>0.20621</td></tr>
<tr><td></td><td>RF</td><td>0.96558</td><td>0.60404</td><td>0.86009</td><td>0.32671</td><td>0.15670</td><td>0.02818</td></tr>
<tr><td></td><td>LightGBM</td><td>0.97198</td><td>0.70104</td><td>0.92550</td><td>0.53228</td><td>0.12646</td><td>0.02520</td></tr>
<tr><td>Polish 2</td><td>XGBoost</td><td>0.96698</td><td>0.62670</td><td>0.87877</td><td>0.38364</td><td>0.13863</td><td>0.02932</td></tr>
<tr><td></td><td>GBDT</td><td>0.95941</td><td>0.52422</td><td>0.80596</td><td>0.09136</td><td>0.14150</td><td>0.03511</td></tr>
<tr><td></td><td>Adaboost</td><td>0.96029</td><td>0.50714</td><td>0.80821</td><td>0.02850</td><td>0.60750</td><td>0.20735</td></tr>
<tr><td></td><td>RF</td><td>0.95735</td><td>0.50688</td><td>0.78031</td><td>0.03115</td><td>0.21795</td><td>0.03505</td></tr>
<tr><td></td><td>LightGBM</td><td>0.96688</td><td>0.62273</td><td>0.88094</td><td>0.37521</td><td>0.12472</td><td>0.02852</td></tr>
<tr><td>Polish 3</td><td>XGBoost</td><td>0.96392</td><td>0.66442</td><td>0.90803</td><td>0.45903</td><td>0.13948</td><td>0.03139</td></tr>
<tr><td></td><td>GBDT</td><td>0.95650</td><td>0.56436</td><td>0.87269</td><td>0.21858</td><td>0.13556</td><td>0.03532</td></tr>
<tr><td></td><td>Adaboost</td><td>0.95812</td><td>0.56987</td><td>0.87695</td><td>0.23689</td><td>0.60682</td><td>0.20703</td></tr>
<tr><td></td><td>RF</td><td>0.95250</td><td>0.52888</td><td>0.85029</td><td>0.10700</td><td>0.17132</td><td>0.03750</td></tr>
<tr><td></td><td>LightGBM</td><td>0.96516</td><td>0.66814</td><td>0.91438</td><td>0.47328</td><td>0.12294</td><td>0.03012</td></tr>
<tr><td>Polish 4</td><td>XGBoost</td><td>0.96580</td><td>0.72660</td><td>0.91587</td><td>0.60405</td><td>0.14480</td><td>0.03052</td></tr>
<tr><td></td><td>GBDT</td><td>0.95590</td><td>0.64939</td><td>0.88667</td><td>0.43878</td><td>0.14137</td><td>0.03642</td></tr>
<tr><td></td><td>Adaboost</td><td>0.95733</td><td>0.65819</td><td>0.89312</td><td>0.46121</td><td>0.61038</td><td>0.20879</td></tr>
<tr><td></td><td>RF</td><td>0.95140</td><td>0.60604</td><td>0.87590</td><td>0.33708</td><td>0.20679</td><td>0.03836</td></tr>
<tr><td></td><td>LightGBM</td><td>0.96498</td><td>0.72614</td><td>0.92567</td><td>0.59885</td><td>0.12806</td><td>0.03013</td></tr>
<tr><td>Polish 5</td><td>XGBoost</td><td>0.95753</td><td>0.76676</td><td>0.93151</td><td>0.64386</td><td>0.15673</td><td>0.03470</td></tr>
<tr><td></td><td>GBDT</td><td>0.95567</td><td>0.72117</td><td>0.92543</td><td>0.58692</td><td>0.13348</td><td>0.03555</td></tr>
<tr><td></td><td>Adaboost</td><td>0.95076</td><td>0.72676</td><td>0.92364</td><td>0.57135</td><td>0.60782</td><td>0.20753</td></tr>
<tr><td></td><td>RF</td><td>0.94636</td><td>0.67300</td><td>0.90531</td><td>0.48229</td><td>0.20644</td><td>0.04129</td></tr>
<tr><td></td><td>LightGBM</td><td>0.96041</td><td>0.76730</td><td>0.93693</td><td>0.65883</td><td>0.15037</td><td>0.03391</td></tr>
<tr><td>Taiwan</td><td>XGBoost</td><td>0.81810</td><td>0.65772</td><td>0.76873</td><td>0.47334</td><td>0.44380</td><td>0.13772</td></tr>
<tr><td></td><td>GBDT</td><td>0.82330</td><td>0.65557</td><td>0.78543</td><td>0.46908</td><td>0.42446</td><td>0.13274</td></tr>
<tr><td></td><td>Adaboost</td><td>0.82247</td><td>0.64613</td><td>0.77777</td><td>0.44973</td><td>0.59326</td><td>0.20067</td></tr>
<tr><td></td><td>RF</td><td>0.81770</td><td>0.65624</td><td>0.76607</td><td>0.47047</td><td>0.45198</td><td>0.13763</td></tr>
<tr><td></td><td>LightGBM</td><td>0.82370</td><td>0.66309</td><td>0.78469</td><td>0.48459</td><td>0.42465</td><td>0.13255</td></tr>
<tr><td>Creator</td><td>XGBoost</td><td>0.99765</td><td>0.99805</td><td>0.99802</td><td>0.99699</td><td>0.01941</td><td>0.00237</td></tr>
<tr><td></td><td>GBDT</td><td>0.99643</td><td>0.99648</td><td>0.99734</td><td>0.99546</td><td>0.02134</td><td>0.00341</td></tr>
<tr><td></td><td>Adaboost</td><td>0.99755</td><td>0.99797</td><td>0.99818</td><td>0.99687</td><td>0.53243</td><td>0.17096</td></tr>
<tr><td></td><td>RF</td><td>0.99768</td><td>0.99807</td><td>0.99804</td><td>0.99705</td><td>0.05168</td><td>0.00248</td></tr>
<tr><td></td><td>LightGBM</td><td>0.99768</td><td>0.99807</td><td>0.99808</td><td>0.99687</td><td>0.01786</td><td>0.00232</td></tr>
</table>
With BLOF</td>
<td>adaptation method, the dimension-reduced feature transformation method, and the stacking-based ensemble learning method with self-adaptive parameter optimization is proposed to achieve good predictive power for credit scoring.</td>
</tr>
</table>

Performance evaluation of the BLOF-based outlier adaptation method.

Dataset	Base classifiers	ACC	BA	AUC	F-score	Log loss	Brier score
Australian	XGBoost	**0.86343**	**0.86520**	**0.94035**	**0.85244**	**0.38258**	**0.10361**
	GBDT	**0.85648**	**0.85938**	**0.93889**	**0.84649**	**0.31370**	**0.09654**
	Adaboost	**0.83565**	**0.83955**	**0.92386**	**0.82609**	**0.63248**	**0.21981**
	RF	**0.87731**	**0.88056**	**0.94868**	**0.87002**	0.44921	**0.09045**
	LightGBM	**0.85185**	**0.85348**	**0.93325**	**0.83963**	**0.39483**	**0.10772**
German	XGBoost	**0.75500**	**0.71962**	**0.79194**	**0.82397**	**0.51149**	**0.16535**
	GBDT	**0.76500**	**0.70076**	**0.79868**	0.82397	**0.51660**	**0.17236**
	Adaboost	**0.76500**	**0.70065**	**0.79060**	**0.82836**	**0.64794**	**0.22754**
	RF	0.74000	0.62675	**0.80485**	**0.83495**	0.50998	**0.16922**
	LightGBM	**0.77000**	**0.73819**	**0.82999**	0.83128	**0.56718**	**0.19000**
Japanese	XGBoost	**0.93673**	**0.93674**	**0.96834**	**0.94010**	**0.23602**	**0.06300**
	GBDT	**0.90306**	**0.90400**	**0.94595**	**0.90466**	**0.25454**	**0.08091**
	Adaboost	**0.88878**	**0.88831**	**0.95324**	**0.89388**	**0.62958**	**0.20512**
	RF	**0.87755**	**0.87848**	**0.97460**	**0.95004**	**0.22715**	**0.07928**
	LightGBM	**0.91837**	**0.91836**	**0.96717**	**0.91919**	**0.23221**	**0.06003**
Polish 1	XGBoost	**0.97856**	**0.73631**	**0.91987**	**0.62222**	**0.07935**	**0.01824**
	GBDT	**0.97680**	**0.73167**	**0.92840**	**0.61196**	**0.08526**	**0.02017**
	Adaboost	**0.97554**	**0.72644**	**0.93462**	**0.59363**	0.60639	0.20680
	RF	**0.97516**	**0.72473**	**0.90024**	**0.58899**	**0.12391**	**0.02190**
	LightGBM	**0.98096**	**0.79634**	**0.94884**	**0.71085**	**0.08663**	**0.01769**
Polish 2	XGBoost	**0.96983**	**0.68027**	0.87304	**0.50000**	**0.11336**	**0.02743**
	GBDT	**0.96250**	**0.55449**	**0.83895**	**0.19329**	**0.12911**	**0.03155**
	Adaboost	0.95922	0.50624	**0.83920**	0.02506	**0.60681**	**0.20701**
	RF	**0.96707**	**0.63015**	**0.83591**	**0.39483**	**0.17999**	**0.02808**
	LightGBM	**0.96940**	**0.70496**	0.87414	**0.52980**	**0.11023**	**0.02531**
Polish 3	XGBoost	**0.96996**	**0.73130**	**0.91584**	**0.59758**	**0.11742**	**0.02617**
	GBDT	**0.95831**	**0.58786**	**0.87305**	**0.29025**	**0.12706**	**0.03281**
	Adaboost	0.95624	0.56860	**0.90559**	0.23419	**0.60632**	**0.20678**
	RF	**0.96021**	**0.62168**	**0.88504**	**0.37207**	**0.15101**	**0.03133**
	LightGBM	**0.97013**	**0.73308**	0.90702	**0.59819**	**0.10552**	**0.02528**
Polish 4	XGBoost	**0.97002**	**0.77212**	**0.93016**	**0.65990**	**0.10301**	**0.02591**
	GBDT	**0.96367**	**0.69465**	**0.92023**	**0.54192**	**0.11506**	**0.02874**
	Adaboost	**0.96251**	**0.68938**	**0.93027**	**0.52812**	0.61126	0.20922
	RF	**0.96421**	**0.67702**	**0.89327**	**0.51220**	**0.13928**	**0.03302**
	LightGBM	**0.97136**	**0.80483**	**0.92141**	**0.69524**	**0.10209**	**0.02365**
Polish 5	XGBoost	**0.96003**	**0.78853**	**0.94566**	**0.68263**	**0.12167**	**0.03286**
	GBDT	**0.96229**	**0.81823**	0.92388	**0.71591**	0.16625	**0.03227**
	Adaboost	**0.95852**	**0.77822**	**0.94477**	**0.66667**	**0.60701**	**0.20566**
	RF	**0.94947**	0.65939	**0.93554**	0.48062	**0.17048**	**0.03780**
	LightGBM	0.95777	**0.77781**	**0.94899**	**0.66265**	**0.12015**	**0.03180**
Taiwan	XGBoost	**0.82997**	**0.66755**	**0.79989**	**0.49265**	**0.42727**	**0.13028**
	GBDT	**0.82572**	**0.66117**	**0.78596**	**0.48087**	**0.42176**	**0.13177**
	Adaboost	0.82175	**0.65707**	**0.78260**	**0.47295**	0.59538	0.20168
	RF	**0.82856**	0.65159	**0.82946**	**0.47905**	**0.42244**	**0.13103**
	LightGBM	**0.82742**	0.65977	**0.80139**	0.47747	**0.41118**	**0.12848**
Creator	XGBoost	**0.99768**	**0.99807**	**0.99819**	**0.99706**	0.01989	**0.00236**
	GBDT	**0.99658**	**0.99669**	**0.99773**	**0.99563**	**0.02103**	**0.00330**
	Adaboost	**0.99759**	**0.99799**	0.99798	**0.99694**	**0.53032**	**0.16997**
	RF	**0.99769**	**0.99809**	**0.99823**	**0.99706**	**0.04188**	**0.00246**
	LightGBM	0.99755	0.99799	**0.99829**	**0.99706**	**0.01756**	0.00240

Table 3
Performance of baseline classifiers in different datasets.

Dataset	Model	Accuracy	Acc_std	AUC	AUC_std	*H* measure	Hm_std	Brier score	Bs_std
Australian	LDA	0.8580	0.0264	0.9228	0.0254	**0.6772**	0.0657	0.1083	0.0238
	LR	**0.8616**	0.0266	**0.9255**	0.0249	0.6696	0.0658	**0.1023**	0.0187
	SVM	0.8507	0.0319	0.9183	0.0228	0.6518	0.0606	0.1082	0.0177
	DT	0.8145	0.0308	0.8124	0.0324	0.4454	0.0826	0.1855	0.0308
	MLP	0.8355	0.0248	0.9006	0.0204	0.6107	0.0596	0.1344	0.0215
German	LDA	0.7300	0.0292	0.7700	0.0310	0.2682	0.0556	0.1795	0.0160
	LR	0.7317	0.0306	0.7684	0.0325	0.2800	0.0556	0.1793	0.0158
	SVM	**0.7483**	0.0247	**0.7753**	0.0273	0.2874	0.0462	**0.1729**	0.0107
	DT	0.6633	0.0245	0.6060	0.0287	0.0544	0.0330	0.3367	0.0245
	MLP	0.7383	0.0184	0.7709	0.0281	**0.3015**	0.0427	0.2220	0.0155
Japanese	LDA	**0.8638**	0.0373	0.9258	0.0284	**0.6758**	0.0853	0.1100	0.0311
	LR	0.8507	0.0320	**0.9263**	0.0285	0.6657	0.0838	**0.1056**	0.0217
	SVM	0.8493	0.0322	0.9190	0.0264	0.6419	0.0731	0.1102	0.0215
	DT	0.8116	0.0284	0.8110	0.0277	0.4308	0.0725	0.1884	0.0284
	MLP	0.8275	0.0300	0.9027	0.0264	0.5894	0.0639	0.1415	0.0245
PPDai	LDA	0.8660	0.0029	0.6494	0.0029	0.0909	0.0048	0.1091	0.0048
	LR	**0.8713**	0.0033	**0.6506**	0.0033	**0.0923**	0.0055	**0.1075**	0.0055
	SVM	0.8709	0.0034	0.5793	0.0037	0.0422	0.1185	0.1113	0.0020
	DT	0.7816	0.0046	0.5424	0.0057	0.0145	0.0043	0.2184	0.0046
	MLP	0.8514	0.0033	0.6155	0.0090	0.0557	0.0083	0.1223	0.0020
GMSC	LDA	0.9312	0.0015	0.6858	0.0015	0.2129	0.0130	0.0597	0.0013
	LR	0.9313	0.0014	0.6892	0.0014	0.1916	0.0089	0.0598	0.0011
	SVM	0.9315	0.0015	0.6288	0.0044	0.2731	0.0112	0.0619	0.0009
	DT	0.8915	0.0019	0.6064	0.0042	0.1133	0.0071	0.1085	0.0019
	MLP	**0.9339**	0.0014	**0.8218**	0.0043	**0.3673**	0.0101	**0.0529**	0.0009

Zhang, W, He, H. Zhang, S	Australian, German, Japanese, PPDai, GMSC	LDA (Linear Discriminant Analysis), LR (Logistic Regression), SVM (Support Vector Machine), DT (Decision Tree), MLP (Multilayer Perceptron Network)	YES		For the datasets of Australian, the highest accuracy yielding method is LR which stands at 0.8616, German, the highest accuracy yielding method is SVM which stands at 0.7483, Japanese, the highest accuracy yielding method

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>is LDA which stands at 0.8638, PPDai, the highest accuracy yielding method is LR which stands at 0.8713 and, GMSC, the highest accuracy yielding method is MLP which stands at 0.9339. A novel multi-stage hybrid model is proposed, which combines feature selection and classifier selection, to obtain a superior model in binary classification.</td>
</tr>
<tr>
<td>Xiao, H,

Xiao, Z,

Wang, Y</td>
<td>Australian, German</td>
<td>Bagging-RS, RS-Bagging, DCE-CC, ECSC</td>
<td>YES</td>
<td>
Table 2
Misclassification rates of Bagging-RS, RS-Bagging, DCE-CC and ECSC (Logit as classifier).
<table>
<tr><th rowspan="2">Method</th><th rowspan="2">Number of base classifiers[a]</th><th colspan="2">Average misclassification rate (%)</th></tr>
<tr><th>Australian</th><th>German</th></tr>
<tr><td>Bagging-RS</td><td>100/100</td><td>12.73 ± 1.76**</td><td>23.05 ± 2.53</td></tr>
<tr><td>RS-Bagging</td><td>100/150</td><td>12.58 ± 1.43**</td><td>23.00 ± 2.04**</td></tr>
<tr><td>DCE-CC</td><td>100/100</td><td>11.73 ± 1.91</td><td>22.63 ± 2.55</td></tr>
<tr><td>ECSC</td><td>50.27/29.63</td><td>11.95 ± 1.89</td><td>22.24 ± 2.35</td></tr>
</table>
[a] The number of base classifiers in Australian/German data sets.
** The paired-t test result (vs. ECSC) is significant at the 0.05 level.
</td>
<td>The misclassification rates of Bagging-RS, RS-Bagging, DCE-CC and, ECSC with different kinds of classifiers over</td>
</tr>
</table>

				Table 3 Misclassification rates of Bagging-RS, RS-Bagging, DCE-CC and ECSC (DT as classifier). Method: Bagging-RS; Number of base classifiers[a]: 100/100; Average misclassification rate (%): Australian 13.07 ± 2.42, German 24.12 ± 2.07 Method: RS-Bagging; Number of base classifiers[a]: 100/150; Average misclassification rate (%): Australian 13.24 ± 2.22, German 24.17 ± 2.09 Method: DCE-CC; Number of base classifiers[a]: 100/100; Average misclassification rate (%): Australian 13.31 ± 2.22**, German 24.74 ± 2.23** Method: ECSC; Number of base classifiers[a]: 50.27/29.63; Average misclassification rate (%): Australian 12.38 ± 2.06, German 23.37 ± 1.85 [a] The number of base classifiers in Australian/German data sets. ** The paired-*t* test result (vs. ECSC) is significant at the 0.05 level. **Table 4** Misclassification rates of Bagging-RS, RS-Bagging, DCE-CC and ECSC (SVM as classifier). Method: Bagging-RS; Number of base classifiers[a]: 100/100; Average misclassification rate (%): Australian 14.18 ± 2.19**, German 30.97 ± 2.52** Method: RS-Bagging; Number of base classifiers[a]: 100/150; Average misclassification rate (%): Australian 13.99 ± 2.01, German 31.09 ± 2.64** Method: DCE-CC; Number of base classifiers[a]: 100/100; Average misclassification rate (%): Australian 13.75 ± 0.17**, German 30.01 ± 2.38 Method: ECSC; Number of base classifiers[a]: 50.27/29.63; Average misclassification rate (%): Australian 13.14 ± 1.85, German 29.40 ± 2.64 [a] The number of base classifiers in Australian/German data sets. ** The paired-*t* test result (vs. ECSC) is significant at the 0.05 level.	30 runs of cross validation are performed and the number of base classifiers are set at 50, 100 and, 150 in order to investigate the impact of classifiers number on the classification performance. The proposed ECSC method is robust since it does not need to set the number of base classifiers, which has a non-negligible impact on the overall classification performance.

<table>
<tr>
<td>Tripathi, D,

Edla, D,

Cheruku, R,

Kuppili, V</td>
<td>Australian, Japanese, German</td>
<td>QDA (Quadratic Discriminant Analysis), NB (Naïve Bayes), MLFN (Multilayer Feed Forward Neural Network), DTDN, TDNN (Time Delay Neural Network), DT (Decision Tree), SVM (Support Vector Machine), KNN (k-Nearest Neighbour)</td>
<td>YES</td>
<td>
TABLE 8 Classifiers ranking using CIV on four datasets
<table>
<tr><th></th><th colspan="4">Australian dataset</th><th colspan="4">Japanese dataset</th></tr>
<tr><th>CL</th><th>Sen</th><th>Spe</th><th>CIV</th><th>R</th><th>Sen</th><th>Spe</th><th>CIV</th><th>R</th></tr>
<tr><td>QDA</td><td>82.01</td><td>90.85</td><td>89.08</td><td>5</td><td>90.26</td><td>80.97</td><td>88.40</td><td>2</td></tr>
<tr><td>NB</td><td>86.24</td><td>85.74</td><td>86.14</td><td>6</td><td>88.04</td><td>82.38</td><td>86.90</td><td>4</td></tr>
<tr><td>MLFN</td><td>82.47</td><td>86.13</td><td>85.39</td><td>7</td><td>85.22</td><td>83.42</td><td>84.86</td><td>5</td></tr>
<tr><td>DTDN</td><td>92.22</td><td>89.86</td><td>91.74</td><td>2</td><td>81.58</td><td>79.98</td><td>81.26</td><td>7</td></tr>
<tr><td>TDNN</td><td>93.43</td><td>85.13</td><td>91.77</td><td>1</td><td>89.47</td><td>78.96</td><td>87.37</td><td>3</td></tr>
<tr><td>DT</td><td>84.64</td><td>78.09</td><td>83.33</td><td>8</td><td>83.35</td><td>81.47</td><td>82.97</td><td>6</td></tr>
<tr><td>SVM</td><td>79.94</td><td>92.49</td><td>89.98</td><td>3</td><td>93.68</td><td>80.97</td><td>91.14</td><td>1</td></tr>
<tr><td>KNN</td><td>89.47</td><td>51.19</td><td>89.81</td><td>4</td><td>83.13</td><td>61.73</td><td>78.85</td><td>8</td></tr>
<tr><th></th><th colspan="4">German-categorical dataset</th><th colspan="4">German-numerical dataset</th></tr>
<tr><td>QDA</td><td>88.00</td><td>47.33</td><td>79.87</td><td>3</td><td>88.42</td><td>49.66</td><td>80.67</td><td>4</td></tr>
<tr><td>NB</td><td>81.84</td><td>55.30</td><td>76.53</td><td>5</td><td>82.05</td><td>51.90</td><td>76.02</td><td>5</td></tr>
<tr><td>MLFN</td><td>75.74</td><td>60.60</td><td>72.71</td><td>7</td><td>75.69</td><td>66.56</td><td>73.86</td><td>7</td></tr>
<tr><td>DTDN</td><td>87.50</td><td>60.77</td><td>82.15</td><td>2</td><td>88.53</td><td>63.84</td><td>83.59</td><td>2</td></tr>
<tr><td>TDNN</td><td>89.28</td><td>61.42</td><td>83.70</td><td>1</td><td>90.82</td><td>67.77</td><td>86.21</td><td>1</td></tr>
<tr><td>DT</td><td>83.42</td><td>49.66</td><td>76.67</td><td>6</td><td>82.57</td><td>47.33</td><td>75.52</td><td>6</td></tr>
<tr><td>SVM</td><td>73.14</td><td>69.66</td><td>72.44</td><td>8</td><td>72.14</td><td>72.66</td><td>72.56</td><td>8</td></tr>
<tr><td>KNN</td><td>95.28</td><td>16.00</td><td>79.42</td><td>4</td><td>95.57</td><td>23.66</td><td>81.18</td><td>3</td></tr>
</table>
CL: Classifier, Sen: Sensitivity, Spe: Specificity, R: Rank. Abbreviations: CIV, Choquet integral value; DT, decision tree; KNN, K-nearest neighborhood; MLFN, multilayer feed forward neural network; NB, Naïve Bayes; QDA, quadratic discriminant analysis; SVM, support vector machine; TDNN, time delay neural network.
</td>
<td>For the dataset of Australian, the highest sensitivity is TDNN which stands at 93.43 % and the highest specificity is also TDNN which stands at 91.77 %. As for the dataset of Japanese, the highest sensitivity is SVM which stands at 93.68 % and the highest specificity is MLFN which stands at 83.42 %. As for the dataset of German-categorical, the highest sensitivity is KNN which stands at 95.28 % and the</td>
</tr>
</table>

					highest specificity is SVM which stands at 69.66 %. As for the dataset of German-numerical, the highest sensitivity is KNN which stands at 95.57 % and the highest specificity is SVM which stands at 72.66 %. The experimental results had indicated that the features selected are more representative and had improved the performance of QDA, NB, MLFN, DTNN, TDNN, DT, and SVM classifiers in terms of

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>classification performance.</td></tr>
<tr><td>Soui, M,

Gasmi, I,

Smiti, S,

Ghédira, K</td><td>Australian, German</td><td>NSGA II, MOEA/D, SMOPSO, SPEA2</td><td>YES</td><td>
Table 5
Performances of credit risk assessment model.
<table>
<tr><td></td><td></td><td>Accuracy (%)</td><td>AUC</td><td>Type Error I (%)</td><td>Type Error II (%)</td></tr>
<tr><td>German dataset</td><td>NSGA II</td><td>85.1</td><td>0.89</td><td>18</td><td>10</td></tr>
<tr><td></td><td>MOEA/D</td><td>76.9</td><td>0.84</td><td>25</td><td>19</td></tr>
<tr><td></td><td>SMOPSO</td><td>92.3</td><td>0.90</td><td>17</td><td>11</td></tr>
<tr><td></td><td>SPEA2</td><td>84.8</td><td>0.87</td><td>25</td><td>15</td></tr>
<tr><td>Australian dataset</td><td>NSGA II</td><td>93.15</td><td>0.89</td><td>9</td><td>9</td></tr>
<tr><td></td><td>MOEA/D</td><td>87.30</td><td>0.87</td><td>15</td><td>10</td></tr>
<tr><td></td><td>SMOPSO</td><td>94</td><td>0.92</td><td>10</td><td>7</td></tr>
<tr><td></td><td>SPEA2</td><td>90</td><td>0.91</td><td>17</td><td>23</td></tr>
</table>
</td><td>For the dataset of German, the highest accuracy is SMOPSO which stands at 92.3 %, the highest AUC is also SMOPSO which stands at 0.90, lowest Type 1 Error is also SMOPSO which stands at 17 and, lowest Type 2 Error is NSGA II which stands at 10. As for the dataset of Australian, the highest accuracy is SMOPSO which stands at 94 %, the highest AUC is also SMOPSO which stands at 0.92, lowest Type 1 Error is NSGA II which stands at 9 and,</td></tr>
</table>

					lowest Type 2 Error is SMOPSO which stands at 7. The model is presented in linguistic rules that can easily be comprehensible by both experts and applicants and the SMOPSO outperforms the other algorithms.
Masmoudi, K, Abid, L, Masmoudi, A	Tunisian Central bank	Discrete BN, Decision Tree, Radial SVM and, BN with latent variable	NO	**Table 3** Performance comparison of the four techniques. Method | Accuracy | Precision | Recall | F1 Discrete BN | 83.23% | 83.05% | 74.87% | 78.73% Decision tree | 78.96% | 81.51% | 62.15% | 63.50% Radial SVM | 83.16% | 83.52% | 72.18% | 76.10% BN with latent variable | 94.77% | 82.31% | 81.13% | 83.50%	The highest yielding accuracy is BN with latent variable which stand at 94.77 %, the highest yielding precision is also BN with latent variable which stand at 85.31 %, the highest recall is also BN with latent variable which

					stand at 81.31 % and, the highest F1 measure is also BN with latent variable which stand at 83.26 %. The obtained BN involved about two classes which have similarities and differences in terms of probability distribution and for both classes, the gender variable seems to have little significant effect on the probability of payment default. The second similarity lies in the fact that the payment default is more likely to occur when the household is under 35.

					Finally, the BN with latent variable is proved to be useful as a decision making tool in loans attribution and clients' selection.

He, H., Zhang, W. and Zhang, S.	Australian, Japanese, German, DefaultData, PPDaiData, LC2017Q1Data	KNN, LR, LDA, SVM, DT, RF, GBDT, AdaB, XGB	NO	The six performance of classifiers in different datasets. Note: Dset is the abbreviation of Dataset, Aus is the Australian dataset, Jap is the Japanese dataset, Ger is the German dataset, Def is the DefaultData dataset, PPD is the PPDaiData dataset and LCD is the LC2017Q1Data dataset and the same abbreviation rules are applied to the following tables.	This study uses BalanceCascade to construct adjustable data subsets by estimating data imbalance ratios. The three stage ensemble model with parameters optimized by PSO algorithm, is used with a stacking approach combining the RF and XGBoost as base classifiers to construct a new ensemble model.

Dset	Clf	AUC	Hmeasure	KS	Fmeasure	GMean	Logloss	AvgR
Aus	KNN	0.84811	0.43886	0.57372	0.73734	0.76613	1.55927	8.5
	LR	0.91635	0.64234	0.73706	0.83548	0.84846	0.37480	4.83
	LDA	0.91676	0.65816	0.75043	0.84971	0.85682	0.44347	3.67
	SVM	0.91425	0.63598	0.73361	0.83123	0.84481	0.36941	5.5
	DT	0.79679	0.40569	0.59357	0.77807	0.79627	**7.02047**	8.5
	RF	0.92788	**0.68451**	**0.76748**	**0.85410**	**0.86693**	0.38119	**2**
	GBDT	0.92944	0.67094	0.75103	0.84111	0.85445	0.34293	2.83
	AdaB	0.90661	0.62091	0.71663	0.82539	0.83986	0.65370	7
	XGB	**0.93008**	0.67920	0.76051	0.84115	0.85412	0.34088	2.17
Jap	KNN	0.83541	0.39393	0.54116	0.80367	0.76149	1.76792	8.67
	LR	0.91492	0.64064	0.73697	0.85505	0.84352	0.38516	4.83
	LDA	0.91491	0.64979	0.75367	0.85589	0.85588	0.46111	4
	SVM	0.91420	0.62484	0.72920	0.85686	0.84348	0.37341	5.17
	DT	0.80931	0.42679	0.61862	0.82566	0.80852	**6.56995**	8.33
	RF	0.92357	0.66451	**0.75796**	**0.88042**	**0.86223**	0.39076	2
	GBDT	0.92280	0.66094	0.74501	0.86295	0.84859	0.36208	3.17
	AdaB	0.90155	0.60512	0.71299	0.84754	0.83056	0.65363	7
	XGB	**0.92535**	**0.66586**	0.74717	0.86368	0.84916	0.35130	**1.83**
Ger	KNN	0.67455	0.11684	0.28448	0.80443	0.52427	2.01788	8.17
	LR	0.78077	0.30036	0.47040	0.82940	0.66005	0.51577	4.17
	LDA	0.78275	0.30329	**0.47191**	0.83040	**0.66144**	0.51616	3.5
	SVM	0.78741	0.30999	0.46388	0.83829	0.61935	0.49446	2.83
	DT	0.61268	0.06498	0.22536	0.76266	0.59326	**11.36339**	8.83
	RF	**0.79198**	**0.32247**	0.46433	**0.84957**	0.60746	0.49669	**2.5**
	GBDT	0.77863	0.28924	0.45231	0.83164	0.63053	0.50950	5.17
	AdaB	0.76387	0.26711	0.44677	0.82491	0.64444	0.67900	6.33
	XGB	0.78634	0.30929	0.46273	0.83803	0.63170	0.49815	3.5
Def	KNN	0.70797	0.18525	0.32743	0.43930	0.57705	2.13499	7.67
	LR	0.77063	0.28279	0.41809	0.47084	0.58433	0.43758	3.83
	LDA	0.76779	0.28144	0.41787	**0.48272**	**0.59687**	0.45000	3.67
	SVM	0.72372	0.24476	0.36637	0.45127	0.56478	0.45476	7.17
	DT	0.60845	0.06960	0.21703	0.39138	0.57381	**9.61771**	8.67
	RF	0.76656	0.27536	0.41293	0.47204	0.59027	0.43843	4.33
	GBDT	0.78474	0.29835	0.44200	0.47141	0.58523	0.42551	**2.33**
	AdaB	0.77890	0.28958	0.42570	0.45633	0.57122	0.68043	5
	XGB	**0.78562**	**0.29888**	**0.44266**	0.47040	0.58318	0.42496	**2.33**
PPD	KNN	0.55500	0.01777	0.08762	0.08769	0.22670	2.33358	6.33
	LR	0.64734	0.08875	0.21650	0.03514	0.13397	0.36833	5.83
	LDA	0.64647	0.08848	0.21360	0.13546	0.28226	0.37176	4.5
	SVM	0.58902	0.04815	0.14851	0.02120	0.10353	0.38163	7.5
	DT	0.54368	0.01546	0.08748	**0.20848**	**0.43654**	**7.48441**	6.33
	RF	0.69103	0.12685	0.27418	0.05992	0.17721	0.35997	3.67
	GBDT	0.70519	0.14101	0.29720	0.04253	0.14789	0.35473	3.33
	AdaB	0.69090	0.12226	0.27145	0.06773	0.18958	0.67632	4.5
	XGB	**0.70668**	**0.14222**	**0.30301**	0.03670	0.13689	0.35425	**3**
LCD	KNN	0.51163	0.00050	0.02309	0.99352	0.00000	0.42512	6.67
	LR	0.69440	0.01589	0.31139	0.99350	0.00000	0.06623	3.5
	LDA	0.70260	0.01620	0.32330	0.99344	0.00000	0.06770	3.17
	SVM	0.58736	0.00562	0.15444	**0.99353**	0.00000	0.06825	4.67
	DT	0.51339	0.00073	0.02678	0.98471	**0.20956**	**1.03953**	7.17
	RF	0.66098	0.00996	0.24678	**0.99353**	0.00000	0.10904	4.5
	GBDT	0.71207	0.01578	0.32132	0.99302	0.00000	0.06916	4
	AdaB	0.68881	0.01201	0.30564	0.99352	0.00000	0.64468	5
	XGB	**0.71450**	**0.01753**	**0.33821**	**0.99353**	0.00000	0.06503	**1.17**

<table>
<tr>
<td>Thanawala, D</td>
<td>Australian, German</td>
<td>KNN, Logistic Regression, Naïve Bayes, SVM (Support Vector Machine), Decision Trees, Random Forest, ANN (Artificial Neural Network)</td>
<td>NO</td>
<td>
Table 8.1
Comparison Table. Comparison of accuracy between our models and the models used in the paper
<table>
<tr><th rowspan="2">Models</th><th colspan="2">German data (%)</th><th colspan="2">Australian data (%)</th></tr>
<tr><th>Our Results</th><th>From paper</th><th>Our Results</th><th>From paper</th></tr>
<tr><td>KNN</td><td>75</td><td>72</td><td>85</td><td>89</td></tr>
<tr><td>Logistic Regression</td><td>77</td><td>-</td><td>87</td><td>-</td></tr>
<tr><td>Naive Bayes</td><td>75</td><td>77</td><td>87</td><td>78</td></tr>
<tr><td>SVM</td><td>77</td><td>78</td><td>87</td><td>85</td></tr>
<tr><td>Decision Trees</td><td>69</td><td>85</td><td>85</td><td>90</td></tr>
<tr><td>Random Forest</td><td>75</td><td>-</td><td>88</td><td>-</td></tr>
<tr><td>ANN</td><td>75</td><td>77</td><td>77</td><td>82</td></tr>
</table>
Table 8.2
Comparison Table of False Positives and False Negatives
<table>
<tr><th rowspan="2">Models</th><th colspan="2">German dataset</th><th colspan="2">Australian dataset</th></tr>
<tr><th>False Positives</th><th>False Negatives</th><th>False Positives</th><th>False Negatives</th></tr>
<tr><td>KNN</td><td>28</td><td>17</td><td>8</td><td>12</td></tr>
<tr><td>Logistic Regression</td><td>30</td><td>25</td><td>14</td><td>8</td></tr>
<tr><td>Naive Bayes</td><td>27</td><td>23</td><td>8</td><td>10</td></tr>
<tr><td>SVM</td><td>33</td><td>22</td><td>19</td><td>4</td></tr>
<tr><td>Decision Tree</td><td>28</td><td>32</td><td>18</td><td>7</td></tr>
<tr><td>Random Forest</td><td>30</td><td>10</td><td>11</td><td>7</td></tr>
<tr><td>ANN</td><td>32</td><td>18</td><td>10</td><td>11</td></tr>
</table>
</td>
<td>SVM has the highest False Positives, Decision Tree has the highest False Negatives, Naïve Bayes has the lowest False Positives and Random Forest has the lowest False Negatives for the German dataset. As for the Australian dataset, SVM has the highest False Positives, KNN has the highest False Negatives, KNN & Naïve Bayes has the lowest False Positives and SVM has the lowest False Negatives. The training-to-test (training:test) ratio that was used in this study is</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>80%:20%. K-fold cross-validation was used with k = 10 in this study.</td></tr>
<tr><td>Harris, T</td><td>German, Barbados</td><td>K means + LR, LR, CSVM-RBF, K means + SVM-RBF, SVM-RBF, CSVM-linear, K means + SVM-linear, SVM-linear</td><td>NO</td><td>

Table 1
Showing comparative classifier performances on German data.

Classifier	Training accuracy	Test accuracy	BAC	AUC	Training time (s)
(1) K means + LR					
Mean	77.625	74.700	69.526	68.868	0.103
S.D	0.995	1.716	3.021	2.758	0.005
(2) LR					
Mean	70.675	68.900	71.955	70.855	0.035
S.D	0.337	1.798	2.699	2.875	0.004
(3) CSVM-RBF					
Mean	84.525	77.100	69.834	69.234	0.071
S.D	2.897	2.114	3.775	3.172	0.038
(4) K means + SVM-RBF					
Mean	83.250	76.500	69.000	68.614	0.141
S.D	2.494	1.604	3.871	3.119	0.184
(5) SVM-RBF					
Mean	83.400	78.000	70.654	69.526	0.122
S.D	2.236	1.843	3.269	2.915	0.021
(6) CSVM-linear					
Mean	79.300	76.300	71.387	70.219	0.029
S.D	0.565	2.477	2.551	2.830	0.004
(7) K means + SVM-linear					
Mean	82.233	76.381	69.238	68.752	0.107
S.D	2.514	2.105	3.790	3.089	0.046
(8) SVM-linear					
Mean	78.950	78.700	69.779	69.133	0.017
S.D	0.404	1.045	2.449	2.942	0.042

Table 2
Showing comparative classifier performances on Barbados data.

Classifier	Training accuracy	Test accuracy	BAC	AUC	Training time (s)
(1) K means + LR					
Mean	76.275	66.438	65.017	65.669	0.474
S.D	1.452	0.738	.583	.656	0.012
(2) LR					
Mean	75.903	66.122	65.055	65.896	0.131
S.D	2.054	1.683	.553	1.311	0.014
(3) CSVM-RBF					
Mean	76.452	66.442	65.022	66.053	11.328
S.D	2.086	2.009	.685	.909	0.046
(4) K means + SVM-RBF					
Mean	76.629	66.442	65.022	65.840	14.650
S.D	1.777	2.018	.673	.849	0.075
(5) SVM-RBF					
Mean	76.643	66.701	65.428	66.070	15.566
S.D	1.750	2.299	1.658	.0775	0.074
(6) CSVM-linear					
Mean	76.684	66.961	65.125	66.139	0.133
S.D	1.924	1.796	1.717	.921	0.008
(7) K means + SVM-linear					
Mean	76.737	67.004	65.068	66.054	0.492
S.D	2.026	1.775	1.737	1.034	0.037
(8) SVM-linear					
Mean	76.447	66.966	65.117	65.983	0.078
S.D	1.890	1.775	1.747	1.07	0.052

</td><td>CSVM represents a possible solution to the limitations of the current crop of classifiers used in practice. Prior work has noted that as datasets get large, non-linear approaches become increasingly computationally expensive. Therefore, the quest for more computationally efficient algorithms has intensified in recent year as data scientist seek to discover patterns in</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>datasets of increased sizes and complexity.</td></tr>
<tr><td>Chen, Z</td><td>German</td><td>Decision Trees, Adaboost, Bagging, Random Forest</td><td>NO</td><td>

Table 3: The accuracy of weighted tree model on the training data and testing set.

<table>
<tr><td colspan="3">Training data</td><td colspan="3">Testing data</td></tr>
<tr><td></td><td>Bad</td><td>Good</td><td></td><td>Bad</td><td>Good</td></tr>
<tr><td>Bad</td><td>0.85854</td><td>0.141463</td><td>Bad</td><td>0.63158</td><td>0.36842</td></tr>
<tr><td>Good</td><td>0.31313</td><td>0.686869</td><td>Good</td><td>0.34146</td><td>0.65854</td></tr>
</table>

Table 4: The accuracy of adaboost model on the training data and testing set.

<table>
<tr><td colspan="3">Training data</td><td colspan="3">Testing data</td></tr>
<tr><td></td><td>Bad</td><td>Good</td><td></td><td>Bad</td><td>Good</td></tr>
<tr><td>Bad</td><td>0.85854</td><td>0.141463</td><td>Bad</td><td>0.60318</td><td>0.39683</td></tr>
<tr><td>Good</td><td>0.31313</td><td>0.686869</td><td>Good</td><td>0.24051</td><td>0.75949</td></tr>
</table>

Table 6: The accuracy of weighted bagging model on the training data and testing set.

<table>
<tr><td colspan="3">Training data</td><td colspan="3">Testing data</td></tr>
<tr><td></td><td>Bad</td><td>Good</td><td></td><td>Bad</td><td>Good</td></tr>
<tr><td>Bad</td><td>0.96032</td><td>0.039683</td><td>Bad</td><td>0.70732</td><td>0.29268</td></tr>
<tr><td>Good</td><td>0.14634</td><td>0.853659</td><td>Good</td><td>0.25483</td><td>0.74517</td></tr>
</table>

Table 8: The accuracy of weighted random forest on the training data and testing set.

<table>
<tr><td colspan="3">Training data</td><td colspan="3">Testing data</td></tr>
<tr><td></td><td>Bad</td><td>Good</td><td></td><td>Bad</td><td>Good</td></tr>
<tr><td>Bad</td><td>0.97561</td><td>0.02439</td><td>Bad</td><td>0.63158</td><td>0.36842</td></tr>
<tr><td>Good</td><td>0.09091</td><td>0.909091</td><td>Good</td><td>0.22439</td><td>0.77561</td></tr>
</table>

</td><td>Part of the problem faced by this author is the imbalance between the predictions of good customers and bad customers as the prediction of good customers always are more precise than the prediction of bad ones. Adaboost is better than Decision Tree whereby Bagging achieves an accuracy more than 70 % which defeats Adaboost. Random Forest however, comes in second place in terms of accuracy</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>measure to that of Bagging. The training-to-test (training:test) ratio that was used in this study is 80%:20%.</td></tr>
<tr><td>Armaki, A,

Fallah, M,

Alborzi, M,

Mohammadzadeh, A</td><td>Australian, German, Japanese, Iranian</td><td>SVMPSO, NN, AMLP, LR, DT, SVM, KNN, NB</td><td>YES</td><td>TABLE III. ACCURACY, F-MEASURE AND AVERAGE RANK OF THE SINGLE BASELINE MODELS
<table>
<tr><th rowspan="2">Model</th><th colspan="4">Accuracy</th><th colspan="4">F-measure</th><th colspan="2">Average Rank</th></tr>
<tr><th>AUS</th><th>GER</th><th>JPN</th><th>IRA</th><th>AUS</th><th>GER</th><th>JPN</th><th>IRA</th><th>Accuracy</th><th>F-measure</th></tr>
<tr><td>SVMPSO</td><td>97.54</td><td>93.90</td><td>98.47</td><td>98.92</td><td>97.15</td><td>88.68</td><td>98.62</td><td>91.54</td><td>1.00</td><td>1.00</td></tr>
<tr><td>NN</td><td>87.54</td><td>81.20</td><td>87.29</td><td>98.66</td><td>86.65</td><td>66.19</td><td>87.70</td><td>89.25</td><td>3.00</td><td>3.00</td></tr>
<tr><td>AMLP</td><td>87.10</td><td>77.40</td><td>87.90</td><td>98.57</td><td>85.53</td><td>65.65</td><td>88.47</td><td>88.62</td><td>3.75</td><td>3.75</td></tr>
<tr><td>LR</td><td>87.68</td><td>73.40</td><td>88.82</td><td>96.21</td><td>86.82</td><td>64.53</td><td>89.62</td><td>76.65</td><td>4.00</td><td>3.25</td></tr>
<tr><td>DT</td><td>86.81</td><td>68.00</td><td>86.68</td><td>98.89</td><td>86.32</td><td>46.67</td><td>86.80</td><td>90.65</td><td>5.00</td><td>4.50</td></tr>
<tr><td>SVM</td><td>85.51</td><td>77.90</td><td>86.52</td><td>94.75</td><td>85.03</td><td>55.71</td><td>86.67</td><td>48.58</td><td>5.50</td><td>6.50</td></tr>
<tr><td>KNN</td><td>73.33</td><td>74.80</td><td>74.89</td><td>95.48</td><td>63.35</td><td>37.31</td><td>77.66</td><td>54.34</td><td>6.75</td><td>7.50</td></tr>
<tr><td>NB</td><td>80.72</td><td>74.40</td><td>80.40</td><td>93.29</td><td>75.77</td><td>60.25</td><td>83.55</td><td>52.11</td><td>7.00</td><td>6.50</td></tr>
</table>
Note: All the numbers are in percentage form. The analyses are carried out in RapidMiner 7.2 program.</td><td>For the datasets of Australian, German, Japanese and Iranian, the highest accuracy yielding method is SVMPSO which stands at 97.54, 93.90, 98.48 and 98.92, respectively. As for the F-measure of datasets of Australian, German, Japanese and Iranian, the highest F-measure is SVMPSO which stands at 97.15, 88.68, 98.62 and</td></tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>91.54, respectively. SVMPSO have significant roles in enhancing predictive power of the proposed models and as a result, a best credit scoring model can assist financial institutions to determine an accurate and confident credit decisions in the future.</td>
</tr>
<tr>
<td>Nalić, J,

Martinović, G,

Žagar, D</td>
<td>Australian, German</td>
<td>GLM, DT, SVM, NB, GLM + DT + SVM + NB, GLM + DT + SVM, GLM + DT + NB, DT + NB + SVM, GLM + DT, GLM + SVM, GLM + NB, DT + SVM,</td>
<td>YES</td>
<td>
Table 8
Measure of performances of model application.
<table>
<tr><th>Single classifier/Ensemble</th><th>Accuracy (%)</th><th>Type I error (%)</th><th>Type II error (%)</th><th>Sensitivity (%)</th><th>F measure (%)</th></tr>
<tr><td>GLM</td><td>79.739296</td><td>17.547052</td><td>2.7136521</td><td>41.178655</td><td>79.739296</td></tr>
<tr><td>DT</td><td>85.061409</td><td>11.099107</td><td>3.8394837</td><td>44.40565</td><td>85.061409</td></tr>
<tr><td>SVM</td><td>79.342696</td><td>17.549895</td><td>3.1074089</td><td>39.338788</td><td>79.342696</td></tr>
<tr><td>NB</td><td>76.431455</td><td>20.212373</td><td>3.3561722</td><td>35.370702</td><td>76.431455</td></tr>
<tr><td>GLM + DT + SVM + NB</td><td>78.758458</td><td>18.375789</td><td>2.8657531</td><td>39.519165</td><td>78.758458</td></tr>
<tr><td>GLM + DT + SVM</td><td>75.25225</td><td>17.62305</td><td>7.124702</td><td>46.70394</td><td>55.5171</td></tr>
<tr><td>GLM + DT + NB</td><td>78.022119</td><td>19.27418</td><td>2.7037016</td><td>39.256669</td><td>78.022119</td></tr>
<tr><td>DT + NB + SVM</td><td>77.989424</td><td>19.107419</td><td>2.8131574</td><td>38.851592</td><td>77.989424</td></tr>
<tr><td>GLM + DT</td><td>87.686928</td><td>10.740888</td><td>1.572184</td><td>57.21624</td><td>87.686928</td></tr>
<tr><td>GLM + SVM</td><td>76.111617</td><td>20.887587</td><td>3.000796</td><td>36.294022</td><td>76.111617</td></tr>
<tr><td>GLM + NB</td><td>74.016319</td><td>23.443453</td><td>2.5402286</td><td>35.865408</td><td>74.016319</td></tr>
<tr><td>DT + SVM</td><td>83.44658</td><td>13.549781</td><td>3.0036391</td><td>45.109592</td><td>83.44658</td></tr>
<tr><td>DT + NB</td><td>83.081253</td><td>13.244158</td><td>3.6745892</td><td>42.020655</td><td>83.081253</td></tr>
<tr><td>SVM + NB</td><td>75.344004</td><td>21.720589</td><td>2.9354068</td><td>35.785421</td><td>75.344004</td></tr>
</table>
</td>
<td>The highest accuracy yielding method is GLM + DT which stands at 87.686928, the lowest Type I Error yielding method is also GLM + DT which stands at 10.740888, the lowest Type II Error yielding</td>
</tr>
</table>

		DT + NB, SVM + NB			method is GLM + DT which stands at 1.572184, the highest sensitivity yielding method is GLM + DT which stands at 57.21624, the highest F-measure yielding method is GLM + DT which stands at 87.686928. GLM + DT has the lowest error rate, so this ensemble hybrid model is the most accurate and efficient in comparison with the other models. It is also shown that GLM + DT has the highest performance with the

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>accuracy of 87.69 %.</td></tr>
<tr><td>Wang, G,

Ma, J,

Huang, L,

Xu, K</td><td>Australian, German</td><td>LRA, LDA, MLP, RBFN, DT, Bagging DT, Random Subspace DT, Random Forest, Rotation Forest, RS-Bagging DT, Bagging-RS DT</td><td>YES</td><td>
Table 3
Results of different methods.
<table>
<tr><th rowspan="2">Methods</th><th colspan="3">Australian credit dataset</th><th colspan="3">German credit dataset</th></tr>
<tr><th>Average accuracy (%)</th><th>Type I error (%)</th><th>Type II error (%)</th><th>Average accuracy (%)</th><th>Type I error (%)</th><th>Type II error (%)</th></tr>
<tr><td>LRA (West, 2000)</td><td>87.25</td><td>11.07</td><td>14.09</td><td>76.30</td><td>11.86</td><td>51.33</td></tr>
<tr><td>LDA (West, 2000)</td><td>85.96</td><td>7.82</td><td>19.06</td><td>72.60</td><td>27.71</td><td>26.67</td></tr>
<tr><td>MLP (West, 2000)</td><td>85.84</td><td>15.40</td><td>13.26</td><td>73.28</td><td>13.52</td><td>57.53</td></tr>
<tr><td>RBFN (West, 2000)</td><td>87.14</td><td>13.15</td><td>12.74</td><td>74.6</td><td>13.47</td><td>52.99</td></tr>
<tr><td>DT</td><td>84.39</td><td>18.00</td><td>13.70</td><td>72.10</td><td>17.06</td><td>53.20</td></tr>
<tr><td>Bagging DT</td><td>86.38</td><td>14.15</td><td>13.19</td><td>76.45</td><td>11.85</td><td>50.83</td></tr>
<tr><td>Random Subspace DT</td><td>86.93</td><td>18.18</td><td>8.97</td><td>76.12</td><td>6.44</td><td>64.60</td></tr>
<tr><td>Random Forest</td><td>86.89</td><td>13.75</td><td>12.60</td><td>77.05</td><td>9.52</td><td>54.28</td></tr>
<tr><td>Rotation Forest</td><td>86.55</td><td>13.17</td><td>13.69</td><td>77.00</td><td>9.39</td><td>54.78</td></tr>
<tr><td>RS-Bagging DT</td><td>88.17</td><td>19.44</td><td>7.52</td><td>78.36</td><td>5.98</td><td>58.56</td></tr>
<tr><td>Bagging-RS DT</td><td>88.01</td><td>15.6</td><td>9.00</td><td>78.52</td><td>7.19</td><td>55.44</td></tr>
</table>
</td><td>DT is one of the most popular classification algorithms in current use in data mining and machine learning and is not often used in credit scoring as the performance of it is inferior to other methods. Therefore, Bagging-RS DT and RS-Bagging DT is introduced to improve the accuracy of models with the use of DT and as per the results of the experiment, the Bagging-RS DT and RS-Bagging DT can be used as an alternative approach in</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>credit scoring applications.</td></tr>
<tr><td>Dahiya, S,

Handa, S,

Singh, N</td><td>German</td><td>LR,
SVM,
CART,
CHAID,
NN,
C5.1,
QUEST,
Ensembled</td><td>YES</td><td>

Table 2. Comparative Accuracy of classifiers using Feature Selection

<table>
<tr><th rowspan="3">S.N</th><th rowspan="3">Classifier</th><th colspan="6">Accuracy on Training and Test sets with cross classification</th></tr>
<tr><th colspan="2">60:40</th><th colspan="2">70:30</th><th colspan="2">50:50</th></tr>
<tr><th>Training</th><th>Test</th><th>Training</th><th>Test</th><th>Training</th><th>Test</th></tr>
<tr><td>1.</td><td>LR</td><td>79.86</td><td>74.82</td><td>78.43</td><td>75.33</td><td>79.06</td><td>73.29</td></tr>
<tr><td>2.</td><td>SVM</td><td>97.46</td><td>71.88</td><td>97.57</td><td>72.67</td><td>97.54</td><td>74.66</td></tr>
<tr><td>3.</td><td>CART</td><td>76.48</td><td>71.64</td><td>78.29</td><td>71.33</td><td>80.7</td><td>68.42</td></tr>
<tr><td>4.</td><td>CHAID</td><td>79.86</td><td>71.15</td><td>75.00</td><td>68.67</td><td>79.67</td><td>70.57</td></tr>
<tr><td>5.</td><td>NN</td><td>73.77</td><td>70.90</td><td>74.43</td><td>70.67</td><td>72.69</td><td>67.45</td></tr>
<tr><td>6.</td><td>C5.1</td><td>83.08</td><td>69.68</td><td>85.29</td><td>69.33</td><td>83.78</td><td>70.76</td></tr>
<tr><td>7.</td><td>QUEST</td><td>78.51</td><td>66.75</td><td>78.86</td><td>69.67</td><td>74.54</td><td>71.73</td></tr>
<tr><td>8.</td><td>Ensembled</td><td>97.46</td><td>82.90</td><td>97.97</td><td>83.03</td><td>98.25</td><td>82.56</td></tr>
</table>

</td><td>The authors of this research had mentioned that the LR and SVM performed better than other base classifiers individually and the results were improved overall for almost all individual models including the ensembled model on the 70:30 ratio.</td></tr>
<tr><td>Saidi, M,

Settouti, N,
El Habib Daho, M,

El Amine Bechar, M</td><td>German</td><td>CART, CS-CART, Bagging CART, Bagging CS-CART, CS-Bagging CART, CS-Bagging CS-CART, Multicost CART, Boosting CART,</td><td>NO</td><td>

Table 3: Classification Results

<table>
<tr><th></th><th>Error</th><th>Specificity</th><th>Sensibility</th><th>Cost</th><th>Friedman test</th></tr>
<tr><td>CART</td><td>27.57</td><td>88.52</td><td>35.87</td><td>47.68</td><td>7</td></tr>
<tr><td>CS-CART</td><td>39.34</td><td>49.76</td><td>83.70</td><td>26.91</td><td>3.5</td></tr>
<tr><td>BAGGING CART</td><td>21.93</td><td>91.87</td><td>46.74</td><td>39.16</td><td>6.62</td></tr>
<tr><td>BAGGING CS-CART</td><td>36.88</td><td>55.50</td><td>80.43</td><td>27.35</td><td>4.76</td></tr>
<tr><td>CS-BAGGING CART</td><td>38.87</td><td>51.67</td><td>82.61</td><td>27.06</td><td>4</td></tr>
<tr><td>CS-BAGGING CS-CART</td><td>43.85</td><td>41.63</td><td>89.13</td><td>25.71</td><td>3</td></tr>
<tr><td>MULTICOST CART</td><td>40.53</td><td>49.76</td><td>81.52</td><td>28.40</td><td>5</td></tr>
<tr><td>BOOSTING CART</td><td>21.93</td><td>87.08</td><td>57.61</td><td>33.18</td><td>5.87</td></tr>
<tr><td>BOOSTING CS-CART</td><td>37.21</td><td>55.50</td><td>79.35</td><td>28.10</td><td>5.25</td></tr>
</table>

</td><td>The ensemble approaches obtain better results than individual classifiers, the insensitive approaches had reached the best classification accuracy, the cost sensitive approaches are intended to</td></tr>
</table>

<table>
<tr><td></td><td></td><td>Boosting CS-CART</td><td></td><td></td><td>reduce the cost at the expense of the accuracy.</td></tr>
<tr><td>Xiao, J,

Zhou, X,

Zhong, Y,

Xie, L,

Gu, X,

Liu, D</td><td>Australian, German, UK-Thomas, PAKDD, Give-Credit</td><td>GCSSE, Subagging, CoBag, Semi-Bagging, Tri-training</td><td>YES</td><td>
Table 4
Comparison with other models on five public datasets.
<table>
<tr><th></th><th>Dataset</th><th>GCSSE</th><th>Subagging</th><th>CoBag</th><th>Semi-Bagging</th><th>Tri-training</th></tr>
<tr><td rowspan="5">Acc</td><td>“uk-thomas”</td><td>0.7150$^{(1)}$</td><td>0.7137$^{(2)}$</td><td>0.7055$^{(4)}$</td><td>0.7057$^{(3)}$</td><td>0.6694$^{(5)}$</td></tr>
<tr><td>“german”</td><td>0.7376$^{(1)}$</td><td>0.7158$^{(3)}$</td><td>0.7114$^{(4)}$</td><td>0.7275$^{(2)}$</td><td>0.6967$^{(5)}$</td></tr>
<tr><td>“australian”</td><td>0.8689$^{(1)}$</td><td>0.8593$^{(3)}$</td><td>0.8663$^{(2)}$</td><td>0.8540$^{(4)}$</td><td>0.7573$^{(5)}$</td></tr>
<tr><td>“PAKDD”</td><td>0.7667$^{(2)}$</td><td>0.7651$^{(4)}$</td><td>0.7656$^{(3)}$</td><td><b>0.7829</b>$^{(1)}$</td><td>0.7550$^{(5)}$</td></tr>
<tr><td>“give-credit”</td><td>0.9320$^{(1)}$</td><td>0.9270$^{(5)}$</td><td>0.9286$^{(4)}$</td><td>0.9313$^{(2)}$</td><td>0.9294$^{(3)}$</td></tr>
<tr><td rowspan="5">AUC</td><td>“uk-thomas”</td><td>0.6177$^{(1)}$</td><td>0.6003$^{(4)}$</td><td>0.6167$^{(2)}$</td><td>0.6021$^{(3)}$</td><td>0.5510$^{(5)}$</td></tr>
<tr><td>“german”</td><td>0.7561$^{(1)}$</td><td>0.7195$^{(3)}$</td><td>0.7215$^{(4)}$</td><td>0.7225$^{(2)}$</td><td>0.7635$^{(5)}$</td></tr>
<tr><td>“australian”</td><td>0.9128$^{(3)}$</td><td><b>0.9173</b>$^{(1)}$</td><td>0.9149$^{(2)}$</td><td>0.8967$^{(4)}$</td><td>0.7708$^{(5)}$</td></tr>
<tr><td>“PAKDD”</td><td>0.5914$^{(2)}$</td><td>0.5876$^{(3)}$</td><td><b>0.6036</b>$^{(1)}$</td><td>0.5670$^{(4)}$</td><td>0.5315$^{(5)}$</td></tr>
<tr><td>“give-credit”</td><td>0.7705$^{(1)}$</td><td>0.7525$^{(3)}$</td><td>0.7696$^{(2)}$</td><td>0.6551$^{(4)}$</td><td>0.5167$^{(5)}$</td></tr>
<tr><td rowspan="5">MinAcc</td><td>“uk-thomas”</td><td>0.3270$^{(1)}$</td><td>0.2604$^{(4)}$</td><td>0.2861$^{(3)}$</td><td>0.1757$^{(5)}$</td><td>0.3021$^{(2)}$</td></tr>
<tr><td>“german”</td><td>0.5244$^{(1)}$</td><td>0.3805$^{(4)}$</td><td>0.4629$^{(3)}$</td><td>0.3471$^{(5)}$</td><td>0.4806$^{(2)}$</td></tr>
<tr><td>“australian”</td><td>0.8913$^{(1)}$</td><td>0.8679$^{(4)}$</td><td>0.8755$^{(3)}$</td><td>0.8413$^{(5)}$</td><td>0.8874$^{(2)}$</td></tr>
<tr><td>“PAKDD”</td><td>0.1738$^{(1)}$</td><td>0.1427$^{(4)}$</td><td>0.1507$^{(3)}$</td><td>0.0774$^{(5)}$</td><td>0.1628$^{(2)}$</td></tr>
<tr><td>“give-credit”</td><td>0.1809$^{(1)}$</td><td>0.1165$^{(3)}$</td><td>0.1693$^{(2)}$</td><td>0.0768$^{(4)}$</td><td>0.0374$^{(5)}$</td></tr>
<tr><td rowspan="5">MajAcc</td><td>“uk-thomas”</td><td>0.9157$^{(1)}$</td><td>0.9078$^{(3)}$</td><td>0.9002$^{(4)}$</td><td>0.9000$^{(5)}$</td><td>0.9130$^{(2)}$</td></tr>
<tr><td>“german”</td><td>0.8719$^{(2)}$</td><td>0.8546$^{(4)}$</td><td>0.8593$^{(3)}$</td><td><b>0.8903</b>$^{(1)}$</td><td>0.7893$^{(5)}$</td></tr>
<tr><td>“australian”</td><td>0.8667$^{(1)}$</td><td>0.8614$^{(2)}$</td><td>0.8538$^{(5)}$</td><td>0.8609$^{(3)}$</td><td>0.8543$^{(4)}$</td></tr>
<tr><td>“PAKDD”</td><td>0.9481$^{(2)}$</td><td>0.9169$^{(4)}$</td><td>0.9182$^{(3)}$</td><td><b>0.9564</b>$^{(1)}$</td><td>0.9003$^{(5)}$</td></tr>
<tr><td>“give-credit”</td><td>0.9892$^{(3)}$</td><td>0.9853$^{(5)}$</td><td>0.9876$^{(4)}$</td><td><b>0.9942</b>$^{(1)}$</td><td>0.9930$^{(2)}$</td></tr>
<tr><td rowspan="5">COST</td><td>“uk-thomas”</td><td>368$^{(2)}$</td><td>418$^{(4)}$</td><td>395$^{(3)}$</td><td>421$^{(5)}$</td><td><b>337</b>$^{(1)}$</td></tr>
<tr><td>“german”</td><td>251$^{(1)}$</td><td>308$^{(4)}$</td><td>288$^{(3)}$</td><td>316$^{(5)}$</td><td>268$^{(2)}$</td></tr>
<tr><td>“australian”</td><td>104$^{(4)}$</td><td>78$^{(2)}$</td><td><b>73</b>$^{(1)}$</td><td>89$^{(3)}$</td><td>209$^{(5)}$</td></tr>
<tr><td>“PAKDD”</td><td>1285$^{(1)}$</td><td>1365$^{(3)}$</td><td>1355$^{(2)}$</td><td>1416$^{(4)}$</td><td>1805$^{(5)}$</td></tr>
<tr><td>“give-credit”</td><td>1006$^{(1)}$</td><td>1151$^{(3)}$</td><td>1092$^{(2)}$</td><td>1204$^{(4)}$</td><td>1625$^{(5)}$</td></tr>
<tr><td colspan="2">Average Rank</td><td>1.48</td><td>3.4</td><td>2.88</td><td>3.4</td><td>3.88</td></tr>
</table>
Note: The boldface in this table marks the best classification performance in each dataset. The numbers in parentheses are the ranks of the five models in each dataset.
</td><td>For the dataset of UK-Thomas, the GCSSE had yielded the highest accuracy which stands at 0.7150. As for the German dataset, the GCSSE had yielded the highest accuracy which stands at 0.7376. As for the Australian dataset, the GCSSE had yielded the highest accuracy which stands at 0.8689. As for the PAKDD dataset, the Semi-Bagging had yielded the highest accuracy which stands at 0.7829. As for the Give-Credit</td></tr>
</table>

					dataset, the GCSSE had yielded the highest accuracy which stands at 0.9320. The authors of this research had mentioned that GCSSE is an overall superior model compared to others used in this research and is recommended for credit scoring applications.

Table 2.3 above indicates the summary of 20 research papers that was reviewed. From it, it was clear that out of the 20 papers, only 2 did not use a German credit dataset, which makes the German credit dataset consist of 90 % of all reviewed papers. From the summary of this literature review, it was also discovered that the highest objective of researches in a credit scoring analysis is done to propose a new method for credit rating. Not only that, the hybrid machine learning technique is the most common when it comes to new proposition of new method for credit rating. When it comes to feature selection, more than half of the 20 papers had used feature selection making it a popular choice, however, it is also important to choose the appropriate feature selection technique as an inappropriate feature selection technique is not guaranteed to increase performance of the models, which is why on a broader perspective, feature selection is almost equally used with the absence and presence of it. As mentioned in this review, K-fold cross validation method is the most common in all research papers when it comes to evaluating machine learning models. Apart from this, the training-to-test (training:test) ratio that is commonly used in research papers is 80%:20%. Finally, Random Forest (RF), k-Nearest Neighbour (k-NN), Support Vector Machines (SVM), Decision Tree (DT), Logistic Regression (LR), Naïve Bayes (NB) and Ensemble Model (Stacking) are machine learning methods that have been used by other authors. Authors have also agreed that Stacking based ensemble methods are also superior to other types of ensemble methods.

To position my standpoint against other authors' model, I would like to develop an ensemble model using Stacking method on the two lowest accurate developed machine learning models on a German credit dataset to determine if the accuracy of the two lowest developed machine learning models can be improved via Stacking.

3.0 Research Plan

When it comes to this project, there would be a research plan consisting of four main phases, which are data acquisition, data pre-processing, model training, and results and model evaluation. This is in accordance with the Knowledge Discovery in Databases (KDD) as per Guarascio et al. (2019). According to Guarascio et al. (2019), the KDD process is pointed at classifiers, which are estimator models that are able to divide data into predefined categories, often called classes such as, Naïve Bayes, Logistic Regression, Decision Trees, k-NN, Random Forest and, Support Vector Machines. As for the first part in KDD, it is to retrieve relevant datasets from the database (Guarascio et al. 2019). The second part would be to pre-process the data, to make sure the datasets would fit in the machine learning models logically (Guarascio et al. 2019). The third part would be to apply machine learning models on the pre-processed dataset to derive informational patterns (Guarascio et al. 2019). Finally, the fourth part would be to evaluate the obtained informational patterns (Guarascio et al. 2019). The Figure 3.0 below will summarise the four phases in the proposed research plan.

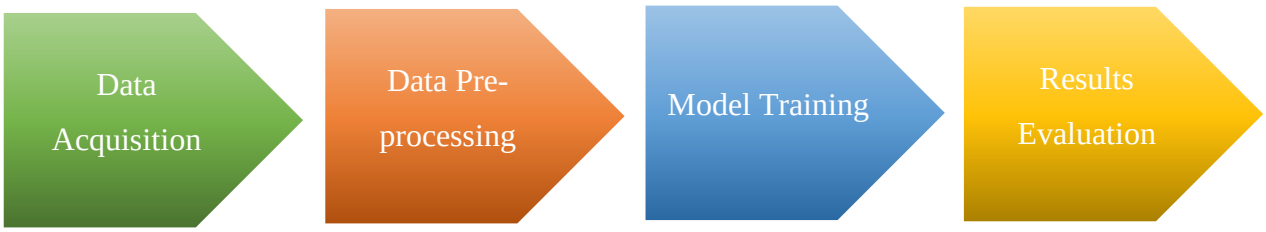


Figure 3.0 :- Proposed Research Plan based on KDD

3.1 Data Acquisition

Referring to Figure 3.0 above, data acquisition happens to be the first phase of the proposed research plan. Hence, the German credit dataset is selected as the primary dataset in this project as most of the datasets used in the literature review happens to be it. According to Ali (2015), the German Credit data is publicly available at the UCI machine-learning repository. This data consist of 1000 samples with 21 available variables related to the credit process (Ali 2015). The default status for each individual is given as a binary variable indicating whether that particular sample is a good loan or a bad loan, and the default status falls under attribute 21 (Ali 2015). There is a total of 700 (70 %) good loans and 300 (30 %) bad loans in the German credit dataset (Chen 2018). According to Tripathi et al. (2019), in the case of German dataset, attributes 1, 3, 4, 6, 7, 9, 10, 12, 14, 15, 17, 19, and 20 are categorical whereas the rest of the attributes are numerical attributes.

No.	Variable	Possible values	Summary
1	Status of existing checking account	A11: ... < 0 DM A12: 0 <= ... < 200 DM A13: ... >= 200 DM /salary assignments for at least 1 year A14: no checking account	Outcome Percentage A11 27.4 A12 26.9 A13 6.3 A14 39.4
2	Duration of loan in months	Numerical (4 to 72)	Min. Median Mean Max. 4.0 18.0 20.9 72.0
3	Credit history	A30: no credits taken/all credits paid back duly A31: all credits at this bank paid back duly A32: existing credits paid back duly till now A33: delay in paying off in the past A34: critical account/ other credits existing (not at this bank)	Outcome Percentage A30 4.0 A31 4.9 A32 53.0 A33 8.8 A34 29.3
4	Purpose	A40: car (new) A41: car (used) A42: furniture/equipment A43: radio/television A44: domestic appliances A45: repairs A46: education A47: (vacation - does not exist?) A48: retraining A49: business A410: others	Outcome Percentage A40 23.4 A41 10.3 A42 18.1 A43 28.0 A44 1.2 A45 2.2 A46 5.0 A48 0.9 A49 9.7 A410 1.2
5	Credit amount	Numerical	Min. Median Mean Max. 250 2320 3271 18420
6	Saving account amount	A61: ... < 100 DM A62: 100 <= ... < 500 DM A63: 500 <= ... < 1000 DM A64: >= 1000 DM A65: unknown/ no savings account	Outcome Percentage A61 60.3 A62 10.3 A63 6.3 A64 4.8 A65 18.3
7	Present employed since	A71: unemployed A72: ... < 1 year A73: 1 <= ... < 4 years A74: 4 <= ... < 7 years A75: >= 7 years	Outcome Percentage A71 6.2 A72 17.2 A73 33.9 A74 17.4 A75 25.3
8	Instalment rate in percentage of disposable income	Numerical	Outcome Percentage 1 13.6 2 23.1 3 15.7 4 47.6
9	Personal status and sex	A91: male: divorced/separated A92: female: divorced/separated/married A93: male: single A94: male: married/widowed A95: female: single	Outcome Percentage A91 5.0 A92 31.0 A93 54.8 A94 9.2
10	Other debtors / guarantors	A101: none A102: co-applicant A103: guarantor	Outcome Percentage A101 90.7 A102 4.1 A103 5.2
11	Present residence since	Numerical	Outcome Percentage 1 13.0 2 30.8 3 14.9 4 41.3
12	Property	A121: real estate A122: if not A121: building society savings agreement/life insurance A123: if not A121/A122: car or other, not in attribute 6 A124: unknown / no property	Outcome Percentage A121 28.2 A122 23.2 A123 33.2 A124 15.4
13	Age	Numerical	Min. Median Mean Max. 19.00 33.00 35.55 75.00
14	Other instalment plans	A141: bank A142: stores A143: none	Outcome Percentage A141 13.9 A142 4.7 A143 81.4
15	Housing	A151: rent A152: own A153: for free	Outcome Percentage A151 17.9 A152 71.3 A153 10.8
16	Number of existing credits at this bank	Numerical	Outcome Percentage 1 63.3 2 33.3 3 2.8 4 0.6
17	Job	A171: unemployed/ unskilled - non-resident A172: unskilled - resident A173: skilled employee / official A174: management/ self-employed/highly qualified employee/ officer	Outcome Percentage A171 2.2 A172 20.0 A173 63.0 A174 14.8
18	Number of dependents	Numerical (form 1 to 2)	Outcome Percentage 1 84.5 2 15.5
19	Telephone	A191: no A192: yes	Outcome Percentage A191 59.6 A192 40.4
20	Foreign worker	A201: yes A202: no	Outcome Percentage A201 96.3 A202 3.7
21	Default status	0: Did not default 1: Did default	Outcome Percentage 0 70 1 30

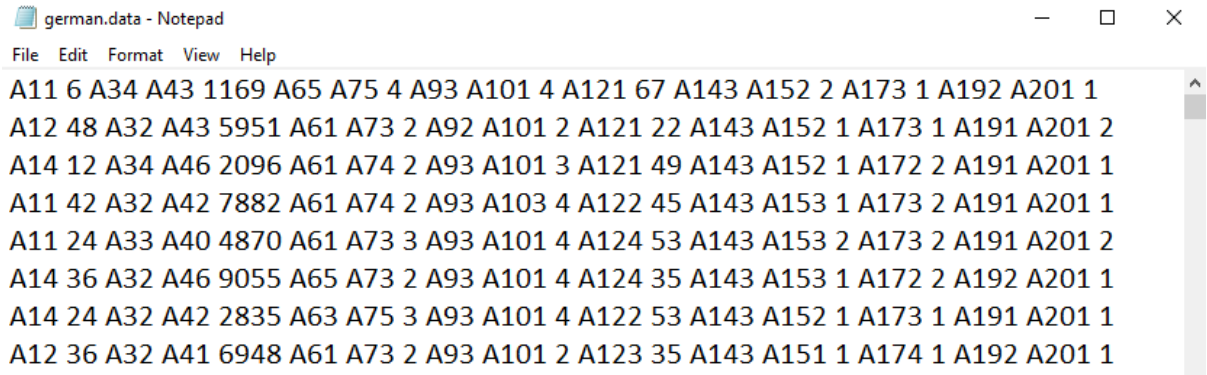
Figure 3.1 :- The summary table for all the variables available in German Credit dataset (Ali 2015)

3.2 Data Pre-processing

Referring to Figure 3.0 above, data pre-processing happens to be the second phase of the proposed research plan. The German credit dataset does not have any missing values, or any obviously miss-specified variables (Ali 2015). Hence, there was no need to perform any quality control in the dataset before performing the downstream analysis, and all the samples in the dataset were utilized (Ali 2015).

According to Gicić and Subasi (2018), credit scoring is the term used to describe the statistical methods that decide if an applicant belongs to a good or bad class. Typically, there is third, intermediate class of “refused”, in addition to the use of “good,” “poor,” and “bad”. In this study, we use the terms “good,” “poor,” and “bad.” The credit applicant is classified into one of the classes based on several quantifiable parameters (Gicić and Subasi 2018). A member of the “bad” risk class has a high possibility of falling into default, which means not repaying the loan as promised (Gicić and Subasi 2018). This means that for the predictor, which is Default Status as attribute 21, the binary variables must be converted to classes such as “NO” and “YES”. “NO” hereby means non-default status meanwhile “YES” hereby means default status. The reason as to why instances in attribute 21 has to be converted to classes is to avoid runtime error during classification and fitting of data via classification models such as Random Forest, k-Nearest Neighbour, Support Vector Machine, Decision Tree, Logistic Regression and, Naïve Bayes. Hence, “NO” will be the positive class. The coding part for this step will be further discussed in Chapter 4, sub-section, 4.2.

In addition to the above, all qualitative attributes such as, for example, A11, A34 has been converted to numerical values. This is to ensure better adaptability in terms of data fitting, in models such as Random Forest, k-Nearest Neighbour, Support Vector Machine, Decision Tree, Logistic Regression and, Naïve Bayes. Figure 3.2.1 & Figure 3.2.2 clearly indicates this step. Besides, the description of columns which corresponds to their columns were added to ensure easy parsing of the possible predictors in R IDE by the Random Forest, k-Nearest Neighbour, Support Vector Machine, Decision Tree, Logistic Regression and, Naïve Bayes models, and this is indicated as in Table 3.2.1.



```
A11 6 A34 A43 1169 A65 A75 4 A93 A101 4 A121 67 A143 A152 2 A173 1 A192 A201 1
A12 48 A32 A43 5951 A61 A73 2 A92 A101 2 A121 22 A143 A152 1 A173 1 A191 A201 2
A14 12 A34 A46 2096 A61 A74 2 A93 A101 3 A121 49 A143 A152 1 A172 2 A191 A201 1
A11 42 A32 A42 7882 A61 A74 2 A93 A103 4 A122 45 A143 A153 1 A173 2 A191 A201 1
A11 24 A33 A40 4870 A61 A73 3 A93 A101 4 A124 53 A143 A153 2 A173 2 A191 A201 2
A14 36 A32 A46 9055 A65 A73 2 A93 A101 4 A124 35 A143 A153 1 A172 2 A192 A201 1
A14 24 A32 A42 2835 A63 A75 3 A93 A101 4 A122 53 A143 A152 1 A173 1 A191 A201 1
A12 36 A32 A41 6948 A61 A73 2 A93 A101 2 A123 35 A143 A151 1 A174 1 A192 A201 1
```

Figure 3.2.1 :- German credit data **before** qualitative to numerical attributes conversion.

A	B	C	D	E	F	G	H	I	J
Status of E	Duration c	Credit His	Purpose	Credit Am	Saving Acc	Present Er	Installmer	Personal s	Other del
1	6	4	3	1169	5	5	4	3	1
2	48	2	3	5951	1	3	2	2	1
4	12	4	6	2096	1	4	2	3	1
1	42	2	2	7882	1	4	2	3	3
1	24	3	0	4870	1	3	3	3	1
4	36	2	6	9055	5	3	2	3	1
4	24	2	2	2835	3	5	3	3	1
2	36	2	1	6948	1	3	2	3	1

Figure 3.2.2 :- German credit data **after** qualitative to numerical attributes conversion.

Table 3.2.1:- Details of Column Descriptions that were added, as per Figure 3.2.2.

Column #	**Column Description Added**
1	Status of Existing Checking Account
2	Duration of loan in months
3	Credit History
4	Purpose
5	Credit Amount
6	Saving Account Amount
7	Present Employed Since
8	Instalment rate in percentage of disposable income
9	Personal status and gender
10	Other debtors or guarantors
11	Present residence since
12	Property
13	Age
14	Other Instalment Plans
15	Housing
16	Number of existing credits at this bank
17	Job
18	Number of Dependents
19	Telephone

20	Foreign Worker
21	Default Status

3.3 Model Training

Referring to Figure 3.0 above, model training happens to be the third phase of the proposed research plan. The German credit dataset will be divided into two separate subsets, whereby one of it concentrates at 80 % or 800 observations to serve as training data while the remaining 20 % or 200 observations to serve as testing dataset. The same training and test datasets will be used for Random Forest, k-Nearest Neighbour, Support Vector Machine, Decision Tree, Logistic Regression and, Naïve Bayes models.

3.4 Results Evaluation

Referring to Figure 3.0 above, results evaluation happens to be the fourth and last phase of the proposed research plan. In order to perform results evaluation, the confusion matrix method will be used to derive accuracy and misclassification rate. A confusion matrix is able to describe the performance of a classification model such as Random Forest, k-Nearest Neighbour, Support Vector Machine, Decision Tree, Logistic Regression and, Naïve Bayes models. The derivation of True Positives, True Negatives, False Positives and, False Negatives can be obtained from a confusion matrix. True Positive and True Negatives are cases whereby the actual and predicted results are the same. A False Positive is a case whereby a positive is predicted, but the actual result reflects a negative and this is commonly known as Type I Error. A False Negative is a case whereby a negative is predicted, but the actual result reflects a positive and this is commonly known as Type II Error. In order to calculate accuracy from the confusion matrix, the summation of True Positives and True Negatives must divide the summation of total results of a sample size. In order to calculate misclassification rate from the confusion matrix, the formula of 1 – Accuracy is used. Therefore, 20 iterations will be done on each developed models to obtain the average True Positives, True Negatives, False Positives and, False Negatives in order to calculate the misclassification rate and accuracy.

Apart from confusion matrix, k-fold, repeated cross-validation will also be used to derive accuracy from Random Forest, k-Nearest Neighbour, Support Vector Machine, Decision Tree, Logistic Regression and, Naïve Bayes models based on a complete German credit dataset with 1000 samples. During the model training in Phase 3, there are chances for overfitting to occur due to the fact that 200 instances for test sample is quite low for the developed models' 800 instances of training sample to be based on. Therefore, the goal of k-fold, repeated cross-validation is performed is to test the developed models' ability to predict higher samples of data accurately. It is mentioned by Brownlee (2020) that k-fold, repeated cross-validation has the benefit of improving the estimate of mean model performance. The way how it works is by multiplying the k-fold value with the number of repeats to obtain the resamples size. According to Brownlee (2018), the general procedure for it would be shuffle the dataset randomly, splitting the dataset into k groups, and for each unique group: group as a hold out or test dataset is taken, remaining groups as training dataset is taken, a model on the training set is fitted and evaluated on the test set, the evaluation score is retained, and the model is discarded. Finally, the skill of the model using the sample of model evaluation scored is summarized.

A detailed discussion will be conducted in Chapter 5 on Results Evaluation.

3.5 Summary

Figure 3.5 indicates the detailed research methodology for this project.

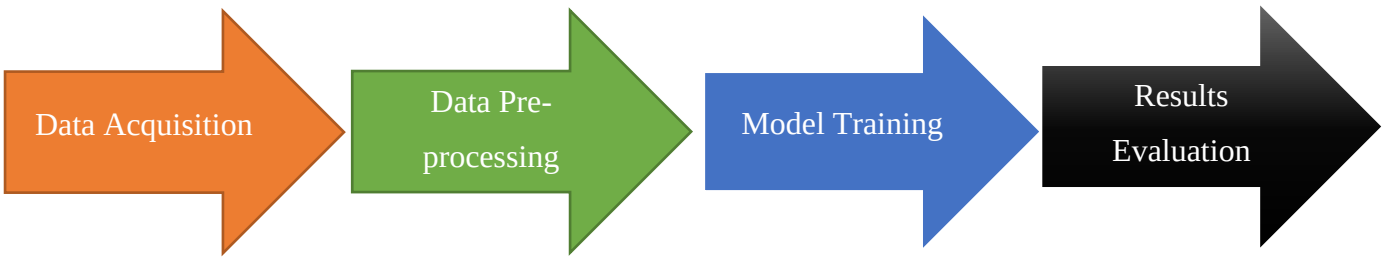


Figure 3.5 :- Summary of Detailed Research Plan

4.0 Implementation

The developed models which are Random Forest (RF), k-Nearest Neighbour (k-NN), Support Vector Machines (SVM), Decision Tree (DT), Logistic Regression (LR), Naïve Bayes (NB) and Ensemble Model (Stacking) are all done via R programming Language. R language is very useful in machine learning applications and simplifies efforts for the end user to program machine learning applications. R was designed for statisticians who do not necessarily have the programming skills. Therefore, R programming is a fierce rival to Python.

4.1 Complete R code used in this research

```
# Setting up Working Directory in R
getwd()
setwd("C:/Users/Oscar's Oasis/Desktop/R Data")

# Initializing R Packages via Library declarations
library(caret)
library(AppliedPredictiveModeling)
library(mlbench)
library(caretEnsemble)

# Loading of the German credit data into R
german.credit <- read.csv(file="German_Data_Modified_2.csv")
german.credit$Default.Status = as.factor(german.credit$Default.Status)
levels(german.credit$Default.Status) <- c("NO", "YES")

# Splitting of German credit data randomly into training (80 %) and testing (20
%)
index <- sample(nrow(german.credit), nrow(german.credit)*0.80)
german.train = german.credit[index,]
german.test = german.credit[-index,]

# Fitting, training, testing & confusion matrix derivation for Random Forest
modelFitRandomForests <- train(Default.Status~., data = german.train, method =
"rf")
predRF <- predict(modelFitRandomForests, newdata = german.test)
confusionMatrix(predRF, german.test$Default.Status)

# Fitting, training, testing & confusion matrix derivation for k-Nearest
Neighbour
modelFitkNearestNeighbour <- train(Default.Status~., data = german.train,
method = "knn")
predKNN <- predict(modelFitkNearestNeighbour, newdata = german.test)
confusionMatrix(predKNN, german.test$Default.Status)

# Fitting, training, testing & confusion matrix derivation for Support Vector
Machine
modelFitSupportVectorMachine <- train(Default.Status~., data = german.train,
method = "svmRadial")
predSVM <- predict(modelFitSupportVectorMachine, newdata = german.test)
confusionMatrix(predSVM, german.test$Default.Status)
```

```r
# Fitting, training, testing & confusion matrix derivation for Decision Tree
modelFitDecisionTree <- train(Default.Status~., data = german.train, method =
"rpart")
predDT <- predict(modelFitDecisionTree, newdata = german.test)
confusionMatrix(predDT, german.test$Default.Status)

# Fitting, training, testing & confusion matrix derivation for Logistic
Regression
modelFitLogisticRegression <- train(Default.Status~., data = german.train,
method = "glm")
predLR <- predict(modelFitLogisticRegression, newdata = german.test)
confusionMatrix(predLR, german.test$Default.Status)

# Fitting, training, testing & confusion matrix derivation for Naive Bayes
modelFitNaiveBayes <- train(Default.Status~., data = german.train, method =
"nb")
predNB <- predict(modelFitNaiveBayes, newdata = german.test)
confusionMatrix(predNB, german.test$Default.Status)

# Evaluation of 5 machine learning methods
controlMethod <- trainControl(method="repeatedcv", number=10, repeats=6,
savePredictions='final', classProbs=TRUE)
baseModelsList <- c('rf', 'knn', 'svmRadial', 'rpart', 'glm', 'nb')
baseModels <- caretList(Default.Status~., data=german.credit, trControl
=controlMethod, methodList=baseModelsList)
resultsA <- resamples(baseModels)
summary(resultsA)

# Pre-Stack for KNN and DT
controlMethod2 <- trainControl(method="repeatedcv", number=10, repeats=6,
savePredictions='final', classProbs=TRUE)
baseModelsList2 <- c('knn', 'rpart')
baseModels2 <- caretList(Default.Status~., data=german.credit, trControl
=controlMethod2, methodList=baseModelsList2)
resultsB <- resamples(baseModels2)
summary(resultsB)
```

```r
# k-Nearest Neighbours Ensemble Model (Stacking)
stackControl <- trainControl(method="repeatedcv", number=10, repeats=6,
savePredictions='final', classProbs=TRUE)
knn_ensembleStack =  caretStack(baseModels2, method = "knn", metric =
"Accuracy", trControl=stackControl)
summary(knn_ensembleStack)
print(knn_ensembleStack)

# CART (Decision Tree) Ensemble Model (Stacking)
stackControl <- trainControl(method="repeatedcv", number=10, repeats=6,
savePredictions='final', classProbs=TRUE)
DT_ensembleStack =  caretStack(baseModels2, method = "rpart", metric =
"Accuracy", trControl=stackControl)
summary(DT_ensembleStack)
print(DT_ensembleStack)
```

4.2. Code Discussion

```
# Setting up Working Directory in R
getwd()
setwd("C:/Users/Oscar's Oasis/Desktop/R Data")
```

Figure 4.2.1 :- Setting up working directory.

Figure 4.2.1 indicates the code used for setting up the working directory, in order to load the German Credit dataset.

```
# Initializing R Packages via Library declarations
library(caret)
library(AppliedPredictiveModeling)
library(mlbench)
library(caretEnsemble)
```

Figure 4.2.2 :- Loading caret, AppliedPredictiveModeling, mlbench, & caretEnsemble libraries

Figure 4.2.2 indicates the caret library which is a short form for Classification and Regression Training. The caret library contain functions to streamline the model training processed for complex regression and classification problems. The AppliedPredictiveModeling library contains a few functions and several data set for the Springer book “Applied Predictive Modeling”. The mlbench library has a collection of artificial and real-world machine learning benchmark problems, including several data sets from the UCI repository. The caretEnsemble library has functions for creating ensembles of caret models.

```
# Loading of the German credit data into R
german.credit <- read.csv(file="German_Data_Modified_2.csv")
german.credit$Default.Status = as.factor(german.credit$Default.Status)
levels(german.credit$Default.Status) <- c("NO", "YES")
```

Figure 4.2.3 :- Loading of the German Credit dataset.

Figure 4.2.3 indicates the loading of German Credit dataset into the R IDE. The predictor column of the dataset which is Default.Status is also converted to as factor

to avoid data frame error while building the models. The instances of the predictor column, Default.Status is also converted to “NO” and “YES”. This means for every instance of “0”, it will be converted to “NO” while every instance of “1” will be converted to “YES”. This is done to avoid any runtime error returned by the IDE during data fitting of the models.

```
# Splitting of German credit data randomly into training (80 %) and testing (20 %)
index <- sample(nrow(german.credit), nrow(german.credit)*0.80)
german.train = german.credit[index,]
german.test = german.credit[-index,]
```

Figure 4.2.4 :- Splitting of the German Credit dataset into 80 % training and 20 % testing.

Figure 4.2.4 indicates the steps that are done to shuffle the German Credit dataset equally during the splitting of training samples into 800 instances and testing samples into 200 instances via “sample” function. The “german.train” is coded to be clear about the object that will assume the 800 instances while the “german.test” is coded to assume the 200 instances from this process.

```
# Fitting, training, testing & confusion matrix derivation for Random Forest
modelFitRandomForests <- train(Default.Status~., data = german.train, method = "rf")
predRF <- predict(modelFitRandomForests, newdata = german.test)
confusionMatrix(predRF, german.test$Default.Status)
```

Figure 4.2.5 :- Fitting of train & test data into Random Forest to calculate Confusion Matrix

Figure 4.2.5 indicates the fitting of the Random Forest model with the train and test data that were obtained earlier. The derivation of confusion matrix is done via the “confusionMatrix” function inputs the “predRF” and “german.test$Default.Status” arguments to calculate the confusion matrix for Random Forest. The “$Default.Status” in the “german.test$Default.Status” happens to be the predictor.

```
# Fitting, training, testing & confusion matrix derivation for k-Nearest Neighbour
modelFitkNearestNeighbour <- train(Default.Status~., data = german.train, method = "knn")
predKNN <- predict(modelFitkNearestNeighbour, newdata = german.test)
confusionMatrix(predKNN, german.test$Default.Status)
```

Figure 4.2.6 :- Fitting of train & test data into k-Nearest Neighbour to calculate Confusion Matrix

Figure 4.2.6 indicates the fitting of the k-Nearest Neighbour model with the train and test data that were obtained earlier. The derivation of confusion matrix is done via the “confusionMatrix” function inputs the “predKNN” and “german.test$Default.Status” arguments to calculate the confusion matrix for k-Nearest Neighbour. The “$Default.Status” in the “german.test$Default.Status” happens to be the predictor.

```
# Fitting, training, testing & confusion matrix derivation for Support Vector Machine
modelFitSupportVectorMachine <- train(Default.Status~., data = german.train, method = "svmRadial")
predSVM <- predict(modelFitSupportVectorMachine, newdata = german.test)
confusionMatrix(predSVM, german.test$Default.Status)
```

Figure 4.2.7 :- Fitting of train & test data into Support Vector Machine to calculate Confusion Matrix

Figure 4.2.7 indicates the fitting of the Support Vector Machine model with the train and test data that were obtained earlier. The derivation of confusion matrix is done via the “confusionMatrix” function inputs the “predSVM” and “german.test$Default.Status” arguments to calculate the confusion matrix for Support Vector Machine. The “$Default.Status” in the “german.test$Default.Status” happens to be the predictor.

```
# Fitting, training, testing & confusion matrix derivation for Decision Tree
modelFitDecisionTree <- train(Default.Status~., data = german.train, method = "rpart")
predDT <- predict(modelFitDecisionTree, newdata = german.test)
confusionMatrix(predDT, german.test$Default.Status)
```

Figure 4.2.8 :- Fitting of train & test data into Decision Tree to calculate Confusion Matrix

Figure 4.2.8 indicates the fitting of the Decision Tree model with the train and test data that were obtained earlier. The derivation of confusion matrix is done via the “confusionMatrix” function inputs the “predDT” and “german.test$Default.Status” arguments to calculate the confusion matrix for Decision Tree. The “$Default.Status” in the “german.test$Default.Status” happens to be the predictor. The “rpart” is an abbreviation used in R to represent the Decision Tree model.

```
# Fitting, training, testing & confusion matrix derivation for Logistic Regression
modelFitLogisticRegression <- train(Default.Status~., data = german.train, method = "glm")
predLR <- predict(modelFitLogisticRegression, newdata = german.test)
confusionMatrix(predLR, german.test$Default.Status)
```

Figure 4.2.9 :- Fitting of train & test data into Logistic Regression to calculate Confusion Matrix

Figure 4.2.9 indicates the fitting of the Logistic Regression model with the train and test data that were obtained earlier. The derivation of confusion matrix is done via the “confusionMatrix” function inputs the “predLR” and “german.test$Default.Status” arguments to calculate the confusion matrix for Logistic Regression. The “$Default.Status” in the “german.test$Default.Status” happens to be the predictor. The “glm” is an abbreviation used in R to represent the Logistic Regression model.

```
# Fitting, training, testing & confusion matrix derivation for Naive Bayes
modelFitNaiveBayes <- train(Default.Status~., data = german.train, method = "nb")
predNB <- predict(modelFitNaiveBayes, newdata = german.test)
confusionMatrix(predNB, german.test$Default.Status)
```

Figure 4.2.10 :- Fitting of train & test data into Naïve Bayes to calculate Confusion Matrix

Figure 4.2.10 indicates the fitting of the Naïve Bayes model with the train and test data that were obtained earlier. The derivation of confusion matrix is done via the “confusionMatrix” function inputs the “predNB” and “german.test$Default.Status” arguments to calculate the confusion matrix for Naïve Bayes. The “$Default.Status” in the “german.test$Default.Status” happens to be the predictor.

```
# Evaluation of 6 machine learning methods
controlMethod <- trainControl(method="repeatedcv", number=10, repeats=6, savePredictions='final', classProbs
=TRUE)
baseModelsList <- c('rf', 'knn', 'svmRadial', 'rpart', 'glm', 'nb')
baseModels <- caretList(Default.Status~., data=german.credit, trControl=controlMethod, methodList
=baseModelsList)
resultsA <- resamples(baseModels)
summary(resultsA)
```

Figure 4.2.11 :- Evaluation of the 6 machine learning models using Repeated Cross-Validation

Figure 4.2.11 indicates the “trainControl” function that inputs “repeatedcv”, “number=10”, “repeats=6”, “savePredictions=”final”, and “classProbs=TRUE” to initialize the 10-fold repeated cross-validation with six repeats. The “baseModelsList”

concatenates all six different machine learning models to a list. The “caretList” is a function that inputs “Default.Status” as the predictor, “data=german.credit” as the applicable dataset, “trControl=controlMethod” as the repeated cross-validation initialization and “methodList=baseModelsList” as the list of machine learning models. The “baseModels” then assumes the data from “caretList” to be used in the “resamples()” function and “summary()” function to display the concatenated accuracy data for all six different machine learning models.

```
# Pre-Stack for KNN and DT
controlMethod2 <- trainControl(method="repeatedcv", number=10, repeats=6, savePredictions='final', classProbs
=TRUE)
baseModelsList2 <- c('knn', 'rpart')
baseModels2 <- caretList(Default.Status~., data=german.credit, trControl=controlMethod2, methodList
=baseModelsList2)
resultsB <- resamples(baseModels2)
summary(resultsB)
```

Figure 4.2.12 :- Evaluation of k-Nearest Neighbour and Decision Tree machine learning models using Repeated Cross-Validation for Stacking

Figure 4.2.12 indicates the “trainControl” function that inputs “repeatedcv”, “number=10”, “repeats=6”, “savePredictions=”final”, and “classProbs=TRUE” to initialize the 10-fold repeated cross-validation with six repeats. The “baseModelsList” concatenates two different models which are k-Nearest Neighbour and Decision Tree machine learning models to a list. The “caretList” is a function that inputs “Default.Status” as the predictor, “data=german.credit” as the applicable dataset, “trControl=controlMethod” as the repeated cross-validation initialization and “methodList=baseModelsList” as the list of machine learning models. The “baseModels” then assumes the data from “caretList” to be used in the “resamples()” function and “summary()” function to display the concatenated accuracy data for k-Nearest Neighbour and Decision Tree.

```
# k-Nearest Neighbours Ensemble Model (Stacking)
stackControl <- trainControl(method="repeatedcv", number=10, repeats=6, savePredictions='final', classProbs
=TRUE)
knn_ensembleStack =  caretStack(baseModels2, method = "knn", metric = "Accuracy", trControl=stackControl)
summary(knn_ensembleStack)
print(knn_ensembleStack)
```

Figure 4.2.13 :- A k-Nearest Neighbour Stacking of two base models which are k-Nearest Neighbour and Decision Tree.

Figure 4.2.13 indicates the “trainControl” function that inputs “repeatedcv”, “number=10”, “repeats=6”, “savePredictions=”final”, and “classProbs=TRUE” to initialize the 10-fold repeated cross-validation with six repeats. The “caretStack” function inputs “baseModels2”, “method=’knn’”, “metric=”Accuracy””, and “trControl=stackControl” to stack the k-Nearest Neighbour model with two other base models which are k-Nearest Neighbour and Decision Tree. The “summary()” function and “print()” function displays the accuracy results for this stack model.

```
# CART (Decision Tree) Ensemble Model (Stacking)
stackControl <- trainControl(method="repeatedcv", number=10, repeats=6, savePredictions='final', classProbs
=TRUE)
DT_ensembleStack =  caretStack(baseModels2, method = "rpart", metric = "Accuracy", trControl=stackControl)
summary(DT_ensembleStack)
print(DT_ensembleStack)
```

Figure 4.2.14 :- A Decision Tree Stacking of two base models which are k-Nearest Neighbour and Decision Tree.

Figure 4.2.14 indicates the “trainControl” function that inputs “repeatedcv”, “number=10”, “repeats=6”, “savePredictions=”final”, and “classProbs=TRUE” to initialize the 10-fold repeated cross-validation with six repeats. The “caretStack” function inputs “baseModels2”, “method=’rpart’”, “metric=”Accuracy””, and “trControl=stackControl” to stack the Decision Tree model with two other base models which are k-Nearest Neighbour and Decision Tree. The “summary()” function and “print()” function displays the accuracy results for this stack model.

5.0 Results and Discussion

This section can be further divided into 7 sub-sections. They are:-

1. Results of Confusion Matrix of Individually Developed Machine Learning Models
2. Comparison of Confusion Matrix between Developed Models vs Literature Review Models
3. Comparison of Misclassification Rate and Accuracy Metrics between Developed Models vs Literature Review Models
4. Results of Accuracy Metrix of n-fold, Repeated Cross-Validation (RepeatCV) of Developed Models with 3 repeats.
5. Results of Accuracy Metrix of 10-fold, Repeated Cross-Validation (RepeatCV) of Developed Models with n repeats.
6. Comparison of Accuracy Metrix of 10-fold, Repeated Cross-Validation (RepeatCV) of Developed Ensemble Models (Stacking) based on the Two Least Accurate Developed Individual Models vs All Developed Machine Learning Models with 6 repeats
7. Discussion

5.1. Results of Confusion Matrix of Individually Developed Machine Learning Models

Table 5.1.1 indicates that the RF average for True Positive, False Positive, False Negative and True Negative are at 133.65, 32.7, 12.25, and 21.4, respectively.

Table 5.1.1 :- 20 Iterations Results for Developed Random Forest Model

Random Forest (RF)				
# Iteration	**True Positive**	**False Positive**	**False Negative**	**True Negative**
1	137	37	5	21
2	133	41	8	18
3	128	31	18	23
4	134	29	16	21
5	131	33	17	19
6	125	22	6	47
7	134	28	15	23
8	136	35	13	16
9	133	38	11	18
10	138	40	5	17

11	141	21	16	22
12	126	34	13	27
13	136	24	18	22
14	135	32	7	26
15	132	46	5	17
16	139	26	12	23
17	138	42	9	11
18	137	35	11	17
19	133	37	18	12
20	127	23	22	28
Average	**133.65**	**32.7**	**12.25**	**21.4**

```
> # Fitting, training, testing & confusion matrix derivation for Random Forest
> modelFitRandomForests <- train(Default.Status~., data = german.train, method = "rf")
> predRF <- predict(modelFitRandomForests, newdata = german.test)
> confusionMatrix(predRF, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  137  37
       YES   5  21
```

Figure 5.1.1 :- Snippet of results of Confusion Matrix for Random Forest, Iteration # 1

```
> # Fitting, training, testing & confusion matrix derivation for Random Forest
> modelFitRandomForests <- train(Default.Status~., data = german.train, method = "rf")
> predRF <- predict(modelFitRandomForests, newdata = german.test)
> confusionMatrix(predRF, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  134  28
       YES  15  23
```

Figure 5.1.2 :- Snippet of results of Confusion Matrix for Random Forest, Iteration # 7

```
> # Fitting, training, testing & confusion matrix derivation for Random Forest
> modelFitRandomForests <- train(Default.Status~., data = german.train, method = "rf")
> predRF <- predict(modelFitRandomForests, newdata = german.test)
> confusionMatrix(predRF, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  132  46
       YES   5  17
```

Figure 5.1.3 :- Snippet of results of Confusion Matrix for Random Forest, Iteration #15

```
> # Fitting, training, testing & confusion matrix derivation for Random Forest
> modelFitRandomForests <- train(Default.Status~., data = german.train, method = "rf")
> predRF <- predict(modelFitRandomForests, newdata = german.test)
> confusionMatrix(predRF, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  127  23
       YES  22  28
```

Figure 5.1.4 :- Snippet of results of Confusion Matrix for Random Forest, Iteration #20

Table 5.1.2 indicates that the knn average for True Positive, False Positive, False Negative and True Negative are at 126.05, 41.4, 16.25, and 16.3, respectively.

Table 5.1.2 :- 20 Iterations Results for Developed k-Nearest Neighbour Model

k-Nearest Neighbours (knn)				
# Iteration	**True Positive**	**False Positive**	**False Negative**	**True Negative**
1	129	39	17	15
2	127	50	11	12
3	118	52	20	10
4	124	52	12	12
5	133	41	15	11
6	125	43	16	16
7	120	56	14	10
8	132	44	11	13
9	128	31	24	17
10	134	33	13	20
11	117	52	16	15
12	131	36	13	20
13	116	39	22	23
14	123	42	14	21
15	113	46	30	11
16	124	34	12	30
17	132	36	16	16
18	137	33	13	17
19	127	23	26	24
20	131	46	10	13
Average	**126.05**	**41.4**	**16.25**	**16.3**

```
> # Fitting, training, testing & confusion matrix derivation for k-Nearest Neighbour
> modelFitkNearestNeighbour <- train(Default.Status~., data = german.train, method = "knn")
> predKNN <- predict(modelFitkNearestNeighbour, newdata = german.test)
> confusionMatrix(predKNN, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  129  39
       YES  17  15
```

Figure 5.1.5 :- Snippet of results of Confusion Matrix for k-Nearest Neighbour, Iteration # 1

```
> # Fitting, training, testing & confusion matrix derivation for k-Nearest Neighbour
> modelFitkNearestNeighbour <- train(Default.Status~., data = german.train, method = "knn")
> predKNN <- predict(modelFitkNearestNeighbour, newdata = german.test)
> confusionMatrix(predKNN, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  120  56
       YES  14  10
```

Figure 5.1.6 :- Snippet of results of Confusion Matrix for k-Nearest Neighbour, Iteration # 7

```
> # Fitting, training, testing & confusion matrix derivation for k-Nearest Neighbour
> modelFitkNearestNeighbour <- train(Default.Status~., data = german.train, method = "knn")
> predKNN <- predict(modelFitkNearestNeighbour, newdata = german.test)
> confusionMatrix(predKNN, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  113  46
       YES  30  11
```

Figure 5.1.7 :- Snippet of results of Confusion Matrix for k-Nearest Neighbour, Iteration #15

```
> # Fitting, training, testing & confusion matrix derivation for k-Nearest Neighbour
> modelFitkNearestNeighbour <- train(Default.Status~., data = german.train, method = "knn")
> predKNN <- predict(modelFitkNearestNeighbour, newdata = german.test)
> confusionMatrix(predKNN, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  131  46
       YES  10  13
```

Figure 5.1.8 :- Snippet of results of Confusion Matrix for k-Nearest Neighbour, Iteration #20

Table 5.1.3 indicates that the SVM average for True Positive, False Positive, False Negative and True Negative are at 125, 36.45, 12.4, and 26.15, respectively.

Table 5.1.3 :- 20 Iterations Results for Developed Support Vector Machine Model

Support Vector Machine (SVM)				
# Iteration	**True Positive**	**False Positive**	**False Negative**	**True Negative**
1	124	41	9	26
2	122	39	11	28
3	121	35	13	31
4	126	36	15	23

5	123	34	12	31
6	125	42	10	23
7	128	41	7	24
8	126	33	15	26
9	127	33	8	32
10	121	36	15	28
11	129	34	14	23
12	124	35	16	25
13	128	32	11	29
14	125	37	8	30
15	125	34	17	24
16	123	33	13	31
17	126	36	10	28
18	127	35	18	20
19	129	38	12	21
20	121	45	14	20
Average	**125**	**36.45**	**12.4**	**26.15**

```
> # Fitting, training, testing & confusion matrix derivation for Support Vector Machine
> modelFitSupportVectorMachine <- train(Default.Status~., data = german.train, method = "svmRadial")
> predSVM <- predict(modelFitSupportVectorMachine, newdata = german.test)
> confusionMatrix(predSVM, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  124  41
       YES   9  26
```

Figure 5.1.9 :- Snippet of results of Confusion Matrix for Support Vector Machine, Iteration # 1

```
> # Fitting, training, testing & confusion matrix derivation for Support Vector Machine
> modelFitSupportVectorMachine <- train(Default.Status~., data = german.train, method = "svmRadial")
> predSVM <- predict(modelFitSupportVectorMachine, newdata = german.test)
> confusionMatrix(predSVM, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  128  41
       YES   7  24
```

Figure 5.1.10 :- Snippet of results of Confusion Matrix for Support Vector Machine, Iteration # 7

```
> # Fitting, training, testing & confusion matrix derivation for Support Vector Machine
> modelFitSupportVectorMachine <- train(Default.Status~., data = german.train, method = "svmRadial")
> predSVM <- predict(modelFitSupportVectorMachine, newdata = german.test)
> confusionMatrix(predSVM, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  125  34
       YES  17  24
```

Figure 5.1.11 :- Snippet of results of Confusion Matrix for Support Vector Machine, Iteration # 15

```
> # Fitting, training, testing & confusion matrix derivation for Support Vector Machine
> modelFitSupportVectorMachine <- train(Default.Status~., data = german.train, method = "svmRadial")
> predSVM <- predict(modelFitSupportVectorMachine, newdata = german.test)
> confusionMatrix(predSVM, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  121  45
       YES  14  20
```

Figure 5.1.12 :- Snippet of results of Confusion Matrix for Support Vector Machine, Iteration # 20

Table 5.1.4 indicates that the DT average for True Positive, False Positive, False Negative and True Negative are at 124.85, 41.6, 16.35, and 17.2, respectively.

Table 5.1.4 :- 20 Iterations Results for Developed Decision Tree Model

Decision Tree (DT)				
# Iteration	**True Positive**	**False Positive**	**False Negative**	**True Negative**
1	126	41	18	15
2	117	51	17	15
3	119	38	18	25
4	126	42	17	15
5	117	34	27	22
6	125	41	16	18
7	138	62	0	0
8	122	34	21	23
9	130	60	8	2
10	130	34	19	17
11	127	29	17	27
12	124	32	18	26
13	120	39	21	20
14	130	39	17	14
15	134	38	15	13
16	121	50	13	16
17	112	34	29	25
18	136	51	6	7
19	128	35	16	21
20	115	48	14	23
Average	**124.85**	**41.6**	**16.35**	**17.2**

```
> # Fitting, training, testing & confusion matrix derivation for Decision Tree
> modelFitDecisionTree <- train(Default.Status~., data = german.train, method = "rpart")
> predDT <- predict(modelFitDecisionTree, newdata = german.test)
> confusionMatrix(predDT, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  126  41
       YES  15  18
```

Figure 5.1.13 :- Snippet of results of Confusion Matrix for Decision Tree, Iteration # 1

```
> # Fitting, training, testing & confusion matrix derivation for Decision Tree
> modelFitDecisionTree <- train(Default.Status~., data = german.train, method = "rpart")
> predDT <- predict(modelFitDecisionTree, newdata = german.test)
> confusionMatrix(predDT, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  138  62
       YES   0   0
```

Figure 5.1.14 :- Snippet of results of Confusion Matrix for Decision Tree, Iteration # 7

```
> # Fitting, training, testing & confusion matrix derivation for Decision Tree
> modelFitDecisionTree <- train(Default.Status~., data = german.train, method = "rpart")
> predDT <- predict(modelFitDecisionTree, newdata = german.test)
> confusionMatrix(predDT, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  134  38
       YES  12  16
```

Figure 5.1.15 :- Snippet of results of Confusion Matrix for Decision Tree, Iteration # 15

```
> # Fitting, training, testing & confusion matrix derivation for Decision Tree
> modelFitDecisionTree <- train(Default.Status~., data = german.train, method = "rpart")
> predDT <- predict(modelFitDecisionTree, newdata = german.test)
> confusionMatrix(predDT, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  115  48
       YES  11  26
```

Figure 5.1.16 :- Snippet of results of Confusion Matrix for Decision Tree, Iteration # 20

Table 5.1.5 indicates that the LR average for True Positive, False Positive, False Negative and True Negative are at 123.55, 31.85, 15.9, and 28.7, respectively.

Table 5.1.5 :- 20 Iterations Results for Developed Logistic Regression Model

Logistic Regression (LR)				
# Iteration	**True Positive**	**False Positive**	**False Negative**	**True Negative**
1	121	28	24	27
2	126	26	17	31
3	124	38	14	24
4	121	37	11	31
5	132	27	10	31
6	131	33	14	22
7	107	34	18	41
8	133	23	14	30
9	127	28	22	23
10	126	34	12	28

11	124	34	13	29
12	140	26	10	24
13	119	34	23	24
14	122	29	23	26
15	118	35	8	39
16	127	36	13	24
17	118	37	18	27
18	120	34	13	33
19	119	32	23	26
20	116	32	18	34
Average	**123.55**	**31.85**	**15.9**	**28.7**

```
> # Fitting, training, testing & confusion matrix derivation for Logistic Regression
> modelFitLogisticRegression <- train(Default.Status~., data = german.train, method = "glm")
> predLR <- predict(modelFitLogisticRegression, newdata = german.test)
> confusionMatrix(predLR, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  121  28
       YES  24  27
```

Figure 5.1.17 :- Snippet of results of Confusion Matrix for Logistic Regression, Iteration # 1

```
> # Fitting, training, testing & confusion matrix derivation for Logistic Regression
> modelFitLogisticRegression <- train(Default.Status~., data = german.train, method = "glm")
> predLR <- predict(modelFitLogisticRegression, newdata = german.test)
> confusionMatrix(predLR, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  107  34
       YES  18  41
```

Figure 5.1.18 :- Snippet of results of Confusion Matrix for Logistic Regression, Iteration # 7

```
> # Fitting, training, testing & confusion matrix derivation for Logistic Regression
> modelFitLogisticRegression <- train(Default.Status~., data = german.train, method = "glm")
> predLR <- predict(modelFitLogisticRegression, newdata = german.test)
> confusionMatrix(predLR, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  118  35
       YES   8  39
```

Figure 5.1.19 :- Snippet of results of Confusion Matrix for Logistic Regression, Iteration # 15

```
> # Fitting, training, testing & confusion matrix derivation for Logistic Regression
> modelFitLogisticRegression <- train(Default.Status~., data = german.train, method = "glm")
> predLR <- predict(modelFitLogisticRegression, newdata = german.test)
> confusionMatrix(predLR, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  116  32
       YES  18  34
```

Figure 5.1.20 :- Snippet of results of Confusion Matrix for Logistic Regression, Iteration # 20

Table 5.1.6 indicates that the NB average for True Positive, False Positive, False Negative and True Negative are at 124.8, 38.95, 15.65, and 20.6, respectively.

Table 5.1.6 :- 20 Iterations Results for Developed Naïve Bayes Model

Naïve Bayes (NB)				
# Iteration	**True Positive**	**False Positive**	**False Negative**	**True Negative**
1	123	38	15	24
2	124	35	13	28
3	127	33	17	23
4	121	36	14	29
5	129	34	19	18
6	128	40	16	16
7	129	51	11	9
8	122	49	15	14
9	130	32	16	22
10	125	33	18	24
11	126	36	14	24
12	131	40	16	13
13	126	35	13	26
14	129	29	17	25
15	125	44	12	19
16	99	25	43	33
17	123	59	10	8
18	126	59	4	11
19	124	35	18	23
20	129	36	12	23
Average	**124.8**	**38.95**	**15.65**	**20.6**

```
> confusionMatrix(predNB, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  123  38
       YES  15  24
```

Figure 5.1.21 :- Snippet of results of Confusion Matrix for Naïve Bayes, Iteration # 1

```
> confusionMatrix(predNB, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  129  51
       YES  11   9
```

Figure 5.1.22 :- Snippet of results of Confusion Matrix for Naïve Bayes, Iteration # 7

```
> confusionMatrix(predNB, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  125  44
       YES  12  19
```

Figure 5.1.23 :- Snippet of results of Confusion Matrix for Naïve Bayes, Iteration # 15

```
> confusionMatrix(predNB, german.test$Default.Status)
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  129  36
       YES  12  23
```

Figure 5.1.24 :- Snippet of results of Confusion Matrix for Naïve Bayes, Iteration # 20

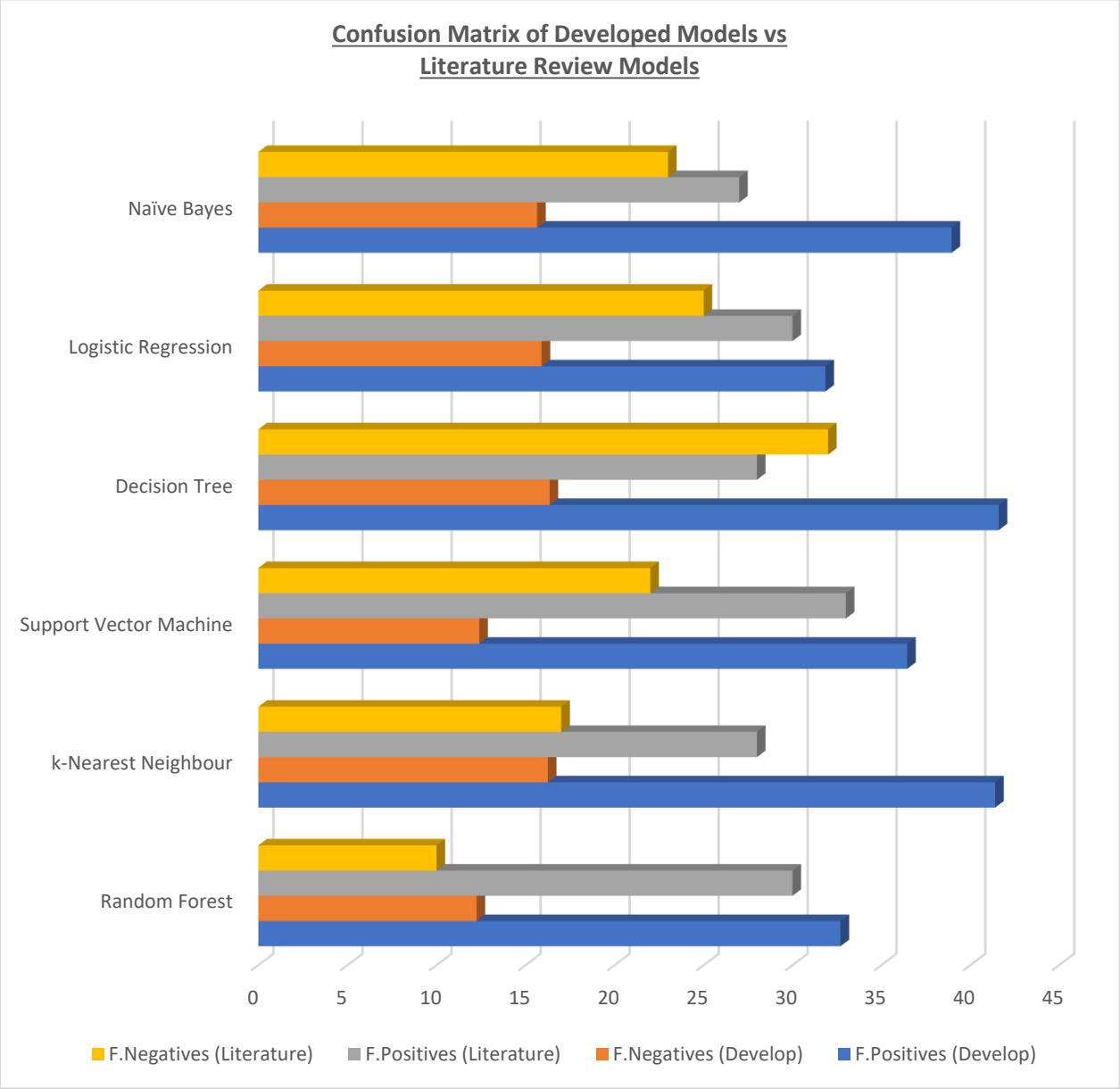
5.2. Comparison of Confusion Matrix between Developed Models vs Literature Review Models

For the results in Table 5.2 below, the reason behind the selection of comparison between False Positives and False Negatives instead of True Negatives and True Positives is because False Positives and False Negatives are sufficient enough to calculate the misclassification rate and accuracy of machine learning models. The misclassification rate can be determined using the Equation 1 in Section 5.3 (Ali 2015; Armaki et al. 2017; He et al. 2018; Livieris et al. 2018; Louzada et al. 2016; Soui et al. 2019; Tripathi et al. 2019; Zhang et al. 2020). Then, the accuracy can be determined using Equation in Section 5.3 (Ali 2015; Armaki et al. 2017; He et al. 2018; Livieris et al. 2018; Louzada et al. 2016; Soui et al. 2019; Tripathi et al. 2019; Zhang et al. 2020). Therefore, the information regarding True Negatives and True Positives are redundant and unnecessary to be displayed.

Table 5.2:- Confusion Matrix of Developed Models vs Literature Review Models

Models	**Results from Developed Models**		**Results from Literature Review Models**	
	Average False Positives	**Average False Negatives**	**False Positives**	**False Negatives**
Random Forest	32.7	12.25	30	10
k-Nearest Neighbour	41.4	16.25	28	17
Support Vector Machine	36.45	12.4	33	22
Decision Tree	41.6	16.35	28	32
Logistic Regression	31.85	15.9	30	25
Naïve Bayes	38.95	15.65	27	23

Graph 5.2 indicates that Decision Tree and k-Nearest Neighbours developed models shows the highest False Positives topping above the 40th mark.



Graph 5.2 :- Confusion Matrix of Developed Models vs Literature Review Models

5.3. Comparison of Misclassification Rate and Accuracy Metrics between Developed Models vs Literature Review Models

For the results in Table 5.3 below, the derivation was done based on Table 5.2, using Equation 1 and Equation 2 to obtain the calculations as presented in it. The average accuracy in the literature models have obtained the highest accuracy compared to the average accuracy of developed models. The average misclassification rate in the literature models is lower compared to that of the developed models.

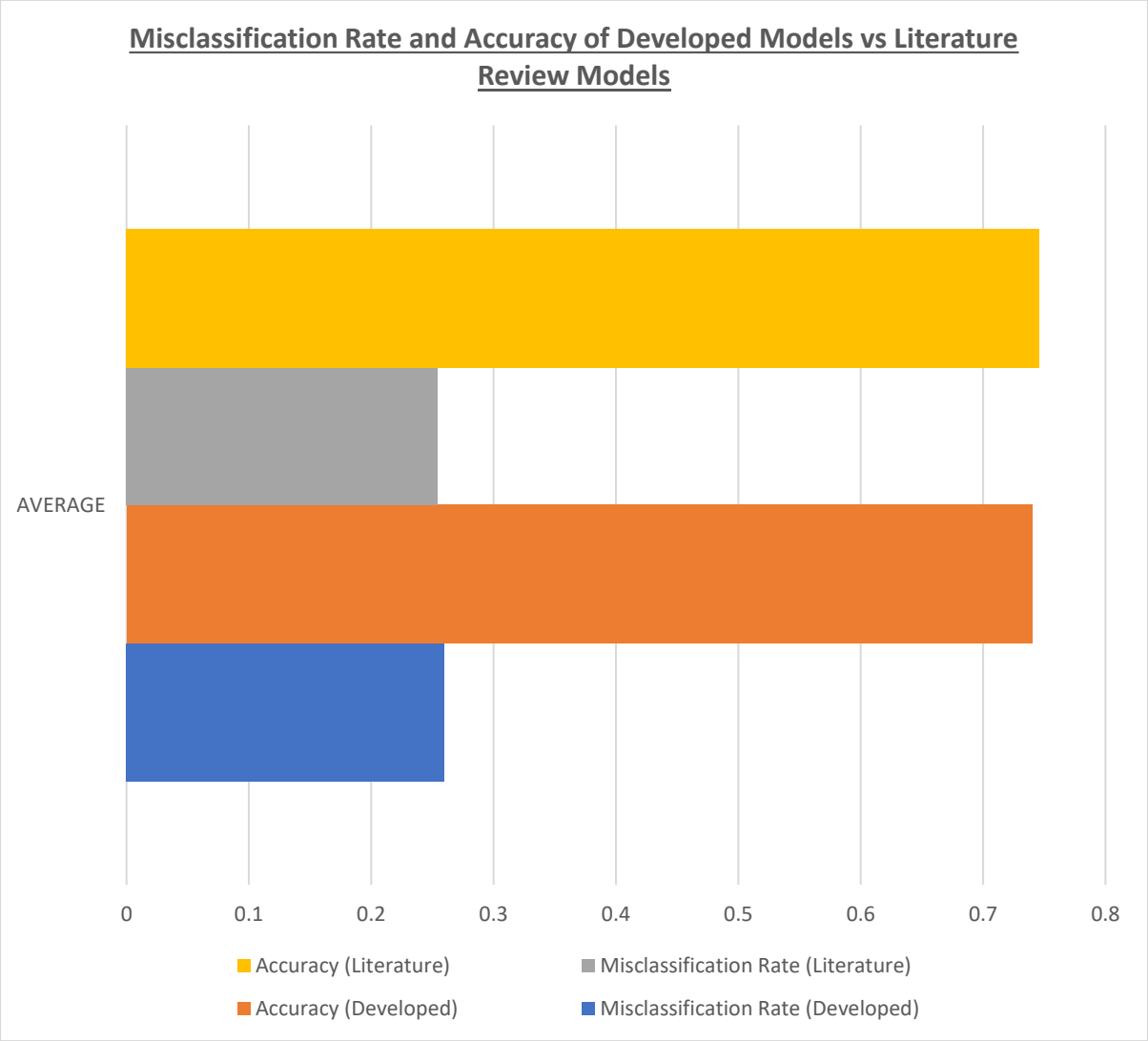
[Equation 1] $$\text{Misclassification Rate} = \frac{\text{False Positives + False Negatives}}{\text{Total summation of instances in test dataset}}$$

[Equation 2] $$\text{Accuracy} = 1 - \text{Misclassification Rate}$$

Table 5.3:- Misclassification Rate and Accuracy of Developed Models vs Literature Review Models

Models	**Results from Developed Models**		**Results from Literature Review Models**	
	Misclassification Rate	**Accuracy**	**Misclassification Rate**	**Accuracy**
Random Forest	0.22475	0.77525	0.2	0.8
k-Nearest Neighbour	0.28825	0.71175	0.225	0.775
Support Vector Machines	0.24425	0.75575	0.275	0.725
Decision Tree	0.28975	0.71025	0.3	0.7
Logistic Regression	0.23875	0.76125	0.275	0.725
Naïve Bayes	0.273	0.727	0.25	0.75
Average	**0.259791667**	**0.740208333**	**0.254166667**	**0.745833333**

Graph 5.3 indicates that there is a slight increment in the average accuracy for the literature models compared to that of the developed models meanwhile a slight decrease in misclassification rate for the literature models compared to that of the developed models.



Graph 5.3 :- Misclassification Rate and Accuracy of Developed Models vs Literature Review Models

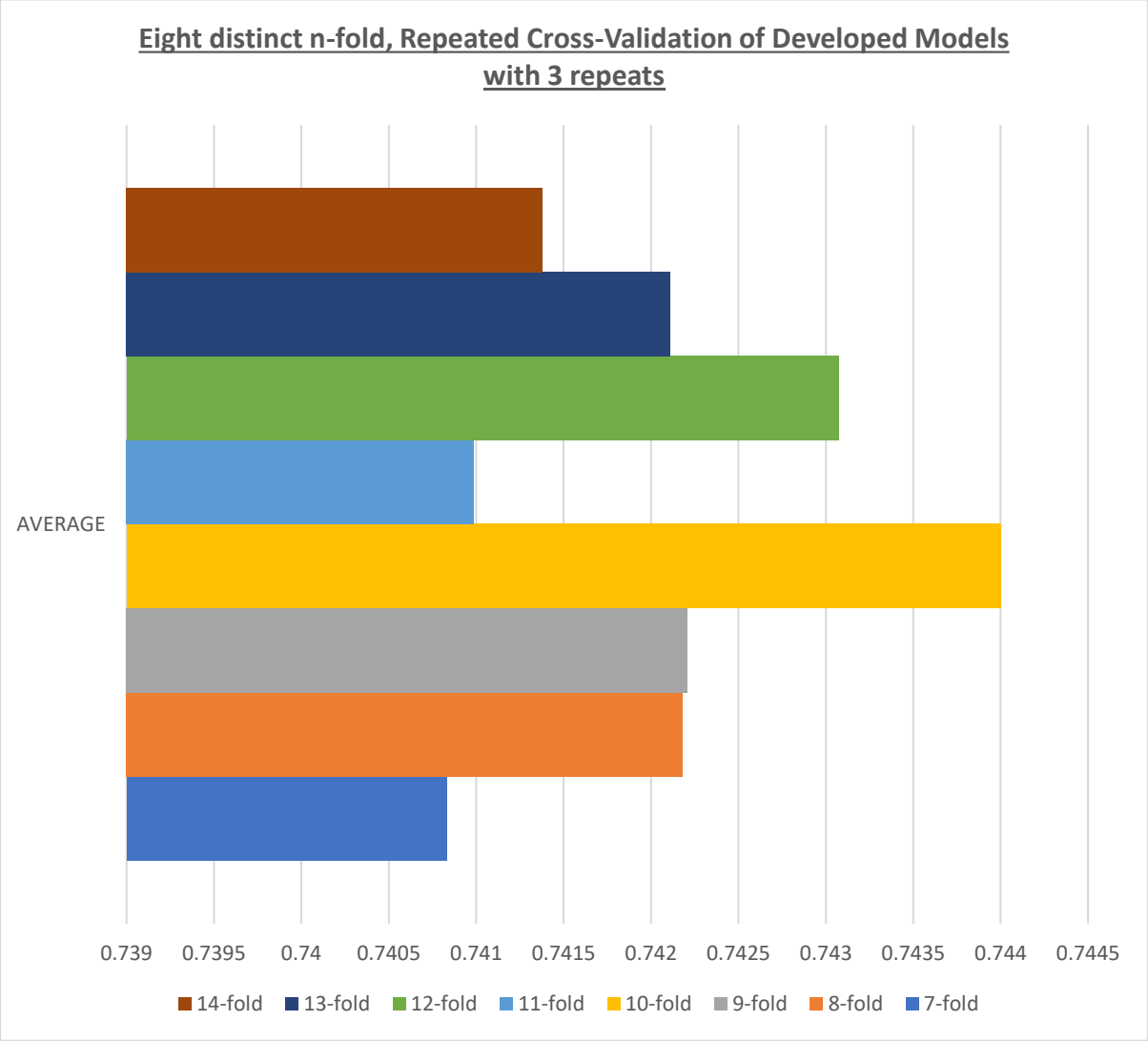
5.4. Results of Accuracy Metrix of n-fold, Repeated Cross-Validation (RepeatCV) of Developed Models with 3 repeats.

Table 5.4 indicates that the 10-fold has the highest average accuracy at 0.744 meanwhile the 7-fold has the lowest average accuracy at 0.740831.

Table 5.4 :- Eight distinct n-fold, Repeated Cross-Validation of Developed Models with 3 repeats

Models	Accuracy							
	7-fold	8-fold	9-fold	10-fold	11-fold	12-fold	13-fold	14-fold
Random Forest	0.765697	0.766002	0.766612	0.769333	0.7642412	0.76765	0.767336	0.76768
k-Nearest Neighbour	0.67665	0.684022	0.687001	0.686	0.6876468	0.6873127	0.685977	0.682703
Support Vector Machines	0.766996	0.767674	0.766669	0.769667	0.7641895	0.7629526	0.770648	0.767922
Decision Tree	0.734327	0.732328	0.731699	0.737667	0.73263	0.7386172	0.726046	0.726996
Logistic Regression	0.759663	0.759354	0.760994	0.761333	0.7542466	0.7592951	0.760009	0.761649
Naïve Bayes	0.741655	0.743711	0.740284	0.74	0.7429752	0.7426213	0.742646	0.741328
Average	**0.740831**	**0.742182**	**0.74221**	**0.744**	**0.7409882**	**0.7430748**	**0.74211**	**0.74138**

Graph 5.4 indicates that the 10-fold has the highest average accuracy at 0.744 meanwhile the 7-fold has the lowest average accuracy topping below than 11-fold.



Graph 5.4 :- Eight distinct n-fold, Repeated Cross-Validation of Developed Models with 3 repeats

```
> resultsA <- resamples(baseModels)
> summary(resultsA)

Call:
summary.resamples(object = resultsA)

Models: rf, knn, svmRadial, rpart, glm, nb
Number of resamples: 21

Accuracy
               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
rf        0.6363636 0.7482517 0.7692308 0.7656967 0.7762238 0.8461538    0
knn       0.6478873 0.6643357 0.6713287 0.6766496 0.6853147 0.7342657    0
svmRadial 0.6853147 0.7482517 0.7692308 0.7669959 0.7902098 0.8251748    0
rpart     0.6853147 0.7132867 0.7412587 0.7343267 0.7552448 0.7832168    0
glm       0.6643357 0.7342657 0.7552448 0.7596629 0.7832168 0.8461538    0
nb        0.6783217 0.7202797 0.7482517 0.7416551 0.7676056 0.7972028    0
```

Figure 5.4.1 :- Snippet of results of 7-fold, Repeated Cross-Validation of Developed Models with 3 repeats

```
> resultsA <- resamples(baseModels)
> summary(resultsA)

Call:
summary.resamples(object = resultsA)

Models: rf, knn, svmRadial, rpart, glm, nb
Number of resamples: 24

Accuracy
               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
rf        0.7280000 0.7529762 0.7661290 0.7660022 0.7788978 0.8095238    0
knn       0.6160000 0.6633226 0.6840118 0.6840222 0.6992127 0.7460317    0
svmRadial 0.7142857 0.7380952 0.7630323 0.7676741 0.7952381 0.8467742    0
rpart     0.6693548 0.7194355 0.7349355 0.7323284 0.7480159 0.7840000    0
glm       0.7177419 0.7375714 0.7529841 0.7593536 0.7750896 0.8225806    0
nb        0.7016129 0.7222222 0.7420635 0.7437105 0.7604762 0.8064516    0
```

Figure 5.4.2 :- Snippet of results of 8-fold, Repeated Cross-Validation of Developed Models with 3 repeats

```
> resultsA <- resamples(baseModels)
> summary(resultsA)

Call:
summary.resamples(object = resultsA)

Models: rf, knn, svmRadial, rpart, glm, nb
Number of resamples: 27

Accuracy
               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
rf        0.7181818 0.7399051 0.7657658 0.7666123 0.7828010 0.8303571    0
knn       0.6306306 0.6787469 0.6875000 0.6870013 0.7013514 0.7500000    0
svmRadial 0.6964286 0.7477477 0.7657658 0.7666690 0.7918509 0.8303571    0
rpart     0.6666667 0.7072072 0.7363636 0.7316989 0.7522727 0.7909091    0
glm       0.6785714 0.7432432 0.7657658 0.7609936 0.7882883 0.8125000    0
nb        0.6936937 0.7175032 0.7410714 0.7402844 0.7657658 0.8125000    0
```

Figure 5.4.3 :- Snippet of results of 9-fold, Repeated Cross-Validation of Developed Models with 3 repeats

```
> resultsA <- resamples(baseModels)
> summary(resultsA)

Call:
summary.resamples(object = resultsA)

Models: rf, knn, svmRadial, rpart, glm, nb
Number of resamples: 30

Accuracy
          Min. 1st Qu. Median      Mean 3rd Qu. Max. NA's
rf        0.69  0.7500  0.775 0.7693333    0.80 0.84    0
knn       0.61  0.6600  0.690 0.6860000    0.71 0.76    0
svmRadial 0.70  0.7400  0.760 0.7696667    0.79 0.85    0
rpart     0.61  0.7100  0.745 0.7376667    0.77 0.79    0
glm       0.70  0.7325  0.760 0.7613333    0.79 0.84    0
nb        0.68  0.7200  0.740 0.7400000    0.76 0.80    0
```

Figure 5.4.4 :- Snippet of results of 10-fold, Repeated Cross-Validation of Developed Models with 3 repeats

```
> resultsA <- resamples(baseModels)
> summary(resultsA)

Call:
summary.resamples(object = resultsA)

Models: rf, knn, svmRadial, rpart, glm, nb
Number of resamples: 33

Accuracy
               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
rf        0.6888889 0.7362637 0.7608696 0.7642412 0.7912088 0.8260870    0
knn       0.6333333 0.6703297 0.6813187 0.6876468 0.6956522 0.7802198    0
svmRadial 0.6555556 0.7333333 0.7692308 0.7641895 0.7934783 0.8351648    0
rpart     0.6813187 0.7065217 0.7333333 0.7326300 0.7582418 0.8043478    0
glm       0.6777778 0.7252747 0.7608696 0.7542466 0.7888889 0.8131868    0
nb        0.6777778 0.7222222 0.7391304 0.7429752 0.7608696 0.8131868    0
```

Figure 5.4.5 :- Snippet of results of 11-fold, Repeated Cross-Validation of Developed Models with 3 repeats

```
> resultsA <- resamples(baseModels)
> summary(resultsA)

Call:
summary.resamples(object = resultsA)

Models: rf, knn, svmRadial, rpart, glm, nb
Number of resamples: 36

Accuracy
               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
rf        0.6987952 0.7492470 0.7710843 0.7676500 0.7880809 0.8192771    0
knn       0.6024096 0.6626506 0.6946357 0.6873127 0.7044966 0.7738095    0
svmRadial 0.6666667 0.7373064 0.7710843 0.7629526 0.7951807 0.8333333    0
rpart     0.6626506 0.7087278 0.7365175 0.7386172 0.7717656 0.8072289    0
glm       0.6626506 0.7373064 0.7590361 0.7592951 0.7880809 0.8674699    0
nb        0.6746988 0.7228916 0.7469880 0.7426213 0.7522590 0.8333333    0
```

Figure 5.4.6 :- Snippet of results of 12-fold, Repeated Cross-Validation of Developed Models with 3 repeats

```
> resultsA <- resamples(baseModels)
> summary(resultsA)

Call:
summary.resamples(object = resultsA)

Models: rf, knn, svmRadial, rpart, glm, nb
Number of resamples: 39

Accuracy
               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
rf        0.6753247 0.7402597 0.7662338 0.7673364 0.7922078 0.8441558    0
knn       0.5974026 0.6710526 0.6883117 0.6859774 0.7097070 0.7532468    0
svmRadial 0.6883117 0.7516234 0.7662338 0.7706483 0.7922078 0.8421053    0
rpart     0.5384615 0.6883117 0.7402597 0.7260458 0.7532468 0.8311688    0
glm       0.6753247 0.7272727 0.7532468 0.7600092 0.7792208 0.8441558    0
nb        0.6363636 0.7272727 0.7435897 0.7426457 0.7661943 0.8311688    0
```

Figure 5.4.7 :- Snippet of results of 13-fold, Repeated Cross-Validation of Developed Models with 3 repeats

```
> resultsA <- resamples(baseModels)
> summary(resultsA)

Call:
summary.resamples(object = resultsA)

Models: rf, knn, svmRadial, rpart, glm, nb
Number of resamples: 42

Accuracy
               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
rf        0.6619718 0.7333236 0.7746479 0.7676802 0.8000293 0.8591549    0
knn       0.5694444 0.6550763 0.6901408 0.6827027 0.7183099 0.7464789    0
svmRadial 0.6760563 0.7387031 0.7746479 0.7679223 0.7916667 0.8333333    0
rpart     0.6338028 0.6912167 0.7342527 0.7269962 0.7579225 0.8055556    0
glm       0.6760563 0.7323944 0.7638889 0.7616486 0.7909331 0.8333333    0
nb        0.6056338 0.7042254 0.7464789 0.7413276 0.7746479 0.8450704    0
```

Figure 5.4.8 :- Snippet of results of 14-fold, Repeated Cross-Validation of Developed Models with 3 repeats

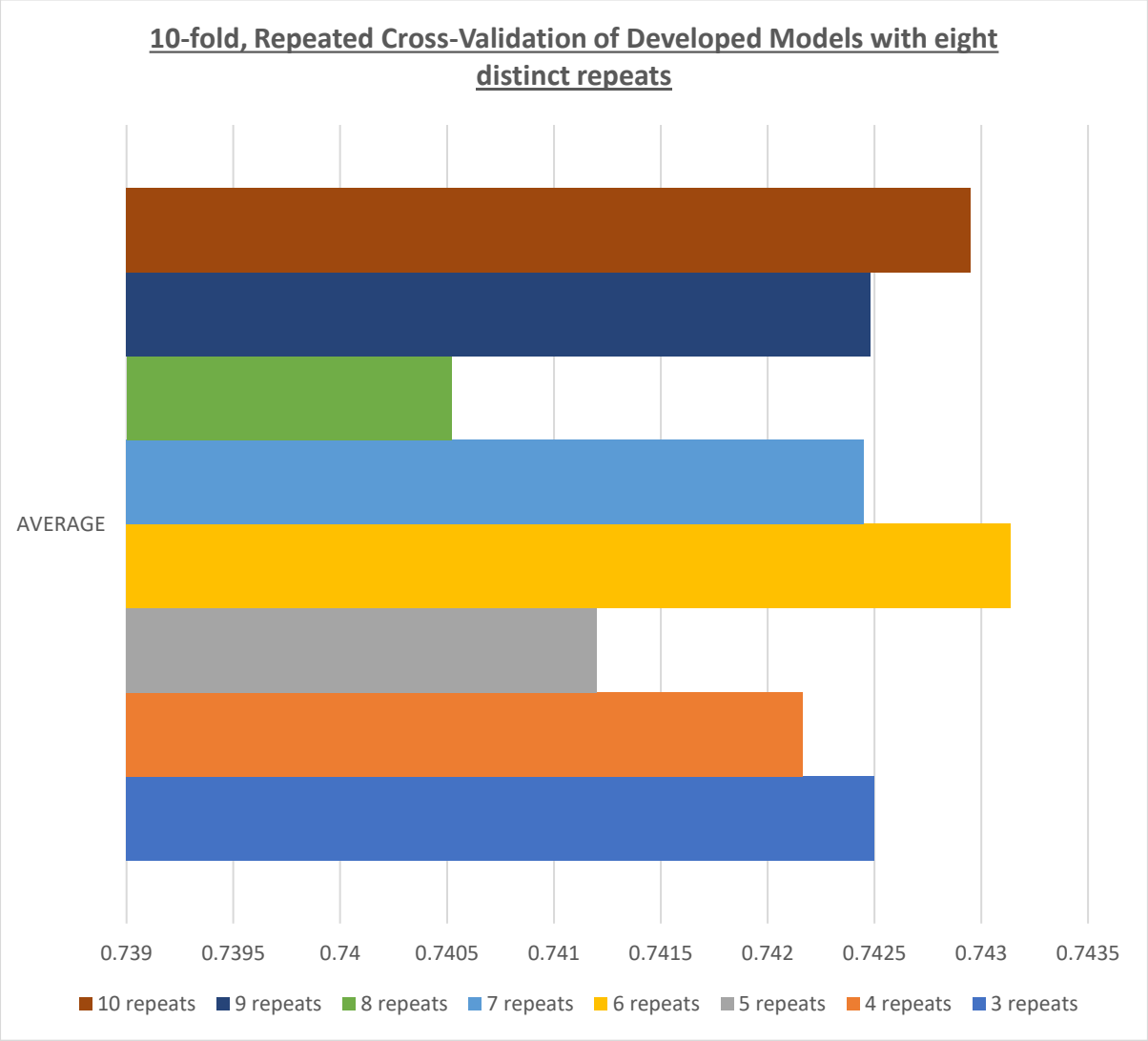
5.5. Results of Accuracy Metrix of 10-fold, Repeated Cross-Validation (RepeatCV) of Developed Models with n repeats.

Table 5.5 indicates that the 6 repeats has the highest average accuracy at 0.7431389 meanwhile the 8 repeats has the lowest average accuracy at 0.740520833.

Table 5.5 :- 10-fold, Repeated Cross-Validation of Developed Models with eight distinct repeats

Models	Accuracy							
	3 repeats	4 repeats	5 repeats	6 repeats	7 repeats	8 repeats	9 repeats	10 repeats
Random Forest	0.766333	0.76875	0.762	0.769167	0.769	0.76788	0.76844	0.7663
k-Nearest Neighbour	0.684	0.6855	0.6796	0.684333	0.68086	0.67825	0.68078	0.6854
Support Vector Machines	0.769333	0.7655	0.7652	0.768667	0.768	0.76563	0.76856	0.7679
Decision Tree	0.733667	0.732	0.7296	0.731167	0.73271	0.72913	0.73211	0.7347
Logistic Regression	0.760333	0.7575	0.7624	0.759333	0.76014	0.75888	0.75922	0.7595
Naïve Bayes	0.741333	0.74375	0.7484	0.746167	0.744	0.74338	0.74578	0.7439
Average	0.742499983	0.742166667	0.7412	0.7431389	0.742452383	0.740520833	0.742481483	0.74295

Graph 5.5 indicates that the 6 repeats has the highest average accuracy topping above 0.743 meanwhile the 8 repeats has the lowest average accuracy at 0.7405.



Graph 5.5 :- 10-fold, Repeated Cross-Validation of Developed Models with eight distinct repeats

```
> resultsA <- resamples(baseModels)
> summary(resultsA)

Call:
summary.resamples(object = resultsA)

Models: rf, knn, svmRadial, rpart, glm, nb
Number of resamples: 30

Accuracy
          Min. 1st Qu. Median      Mean 3rd Qu. Max. NA's
rf        0.73  0.7500  0.760 0.7663333  0.7775 0.83    0
knn       0.60  0.6700  0.680 0.6840000  0.7075 0.74    0
svmRadial 0.71  0.7500  0.760 0.7693333  0.7900 0.84    0
rpart     0.69  0.7100  0.735 0.7336667  0.7500 0.81    0
glm       0.70  0.7425  0.760 0.7603333  0.7800 0.82    0
nb        0.69  0.7200  0.740 0.7413333  0.7575 0.80    0
```

Figure 5.5.1 :- Snippet of results of 10-fold, Repeated Cross-Validation of Developed Models with 3 repeats

```
> resultsA <- resamples(baseModels)
> summary(resultsA)

Call:
summary.resamples(object = resultsA)

Models: rf, knn, svmRadial, rpart, glm, nb
Number of resamples: 40

Accuracy
          Min. 1st Qu. Median    Mean 3rd Qu. Max. NA's
rf        0.67  0.7400   0.77 0.76875  0.7900 0.86    0
knn       0.59  0.6600   0.69 0.68550  0.7100 0.74    0
svmRadial 0.69  0.7500   0.77 0.76550  0.7800 0.82    0
rpart     0.65  0.7100   0.74 0.73200  0.7525 0.81    0
glm       0.67  0.7400   0.76 0.75750  0.7800 0.82    0
nb        0.63  0.7175   0.75 0.74375  0.7700 0.81    0
```

Figure 5.5.2 :- Snippet of results of 10-fold, Repeated Cross-Validation of Developed Models with 4 repeats

```
> resultsA <- resamples(baseModels)
> summary(resultsA)

Call:
summary.resamples(object = resultsA)

Models: rf, knn, svmRadial, rpart, glm, nb
Number of resamples: 50

Accuracy
          Min. 1st Qu. Median   Mean 3rd Qu. Max. NA's
rf        0.67  0.7400  0.755 0.7620  0.7900 0.85    0
knn       0.60  0.6625  0.680 0.6796  0.7000 0.77    0
svmRadial 0.69  0.7400  0.770 0.7652  0.7900 0.83    0
rpart     0.63  0.7000  0.730 0.7296  0.7575 0.82    0
glm       0.68  0.7325  0.760 0.7624  0.7900 0.85    0
nb        0.67  0.7200  0.750 0.7484  0.7775 0.81    0
```

Figure 5.5.3 :- Snippet of results of 10-fold, Repeated Cross-Validation of Developed Models with 5 repeats

```
> resultsA <- resamples(baseModels)
> summary(resultsA)

Call:
summary.resamples(object = resultsA)

Models: rf, knn, svmRadial, rpart, glm, nb
Number of resamples: 60

Accuracy
          Min. 1st Qu. Median      Mean 3rd Qu. Max. NA's
rf        0.70  0.7475  0.770 0.7691667  0.7900 0.87    0
knn       0.62  0.6675  0.685 0.6843333  0.7100 0.74    0
svmRadial 0.70  0.7500  0.770 0.7686667  0.7900 0.85    0
rpart     0.63  0.7175  0.740 0.7311667  0.7525 0.81    0
glm       0.67  0.7400  0.760 0.7593333  0.7800 0.83    0
nb        0.68  0.7300  0.750 0.7461667  0.7600 0.82    0
```

Figure 5.5.4 :- Snippet of results of 10-fold, Repeated Cross-Validation of Developed Models with 6 repeats

```
> resultsA <- resamples(baseModels)
> summary(resultsA)

Call:
summary.resamples(object = resultsA)

Models: rf, knn, svmRadial, rpart, glm, nb
Number of resamples: 70

Accuracy
          Min. 1st Qu. Median      Mean 3rd Qu. Max. NA's
rf        0.68  0.7500  0.770 0.7690000  0.7900 0.84    0
knn       0.61  0.6625  0.685 0.6808571  0.7000 0.74    0
svmRadial 0.69  0.7500  0.770 0.7680000  0.7900 0.84    0
rpart     0.64  0.7100  0.730 0.7327143  0.7575 0.80    0
glm       0.64  0.7400  0.765 0.7601429  0.7800 0.84    0
nb        0.66  0.7200  0.750 0.7440000  0.7700 0.82    0
```

Figure 5.5.5 :- Snippet of results of 10-fold, Repeated Cross-Validation of Developed Models with 7 repeats

```
> resultsA <- resamples(baseModels)
> summary(resultsA)

Call:
summary.resamples(object = resultsA)

Models: rf, knn, svmRadial, rpart, glm, nb
Number of resamples: 80

Accuracy
          Min. 1st Qu. Median     Mean 3rd Qu. Max. NA's
rf        0.66  0.7500  0.770 0.767875  0.7925 0.84    0
knn       0.60  0.6600  0.675 0.678250  0.7100 0.76    0
svmRadial 0.70  0.7400  0.770 0.765625  0.7800 0.84    0
rpart     0.63  0.7075  0.730 0.729125  0.7500 0.84    0
glm       0.68  0.7300  0.760 0.758875  0.7800 0.86    0
nb        0.64  0.7200  0.750 0.743375  0.7625 0.80    0
```

Figure 5.5.6 :- Snippet of results of 10-fold, Repeated Cross-Validation of Developed Models with 8 repeats

```
> resultsA <- resamples(baseModels)
> summary(resultsA)

Call:
summary.resamples(object = resultsA)

Models: rf, knn, svmRadial, rpart, glm, nb
Number of resamples: 90

Accuracy
          Min. 1st Qu. Median      Mean 3rd Qu. Max. NA's
rf        0.64  0.7425   0.78 0.7684444  0.7900 0.87    0
knn       0.62  0.6600   0.68 0.6807778  0.7000 0.76    0
svmRadial 0.67  0.7400   0.77 0.7685556  0.8000 0.85    0
rpart     0.66  0.7100   0.73 0.7321111  0.7500 0.84    0
glm       0.66  0.7325   0.76 0.7592222  0.7900 0.85    0
nb        0.64  0.7300   0.74 0.7457778  0.7675 0.83    0
```

Figure 5.5.7 :- Snippet of results of 10-fold, Repeated Cross-Validation of Developed Models with 9 repeats

```
> resultsA <- resamples(baseModels)
> summary(resultsA)

Call:
summary.resamples(object = resultsA)

Models: rf, knn, svmRadial, rpart, glm, nb
Number of resamples: 100

Accuracy
          Min. 1st Qu. Median   Mean 3rd Qu. Max. NA's
rf        0.69  0.7400   0.77 0.7663  0.7900 0.85    0
knn       0.60  0.6675   0.68 0.6854  0.7100 0.76    0
svmRadial 0.67  0.7400   0.77 0.7679  0.8000 0.83    0
rpart     0.64  0.7100   0.74 0.7347  0.7525 0.82    0
glm       0.66  0.7400   0.76 0.7595  0.7825 0.84    0
nb        0.62  0.7200   0.74 0.7439  0.7700 0.82    0
```

Figure 5.5.8 :- Snippet of results of 10-fold, Repeated Cross-Validation of Developed Models with 10 repeats

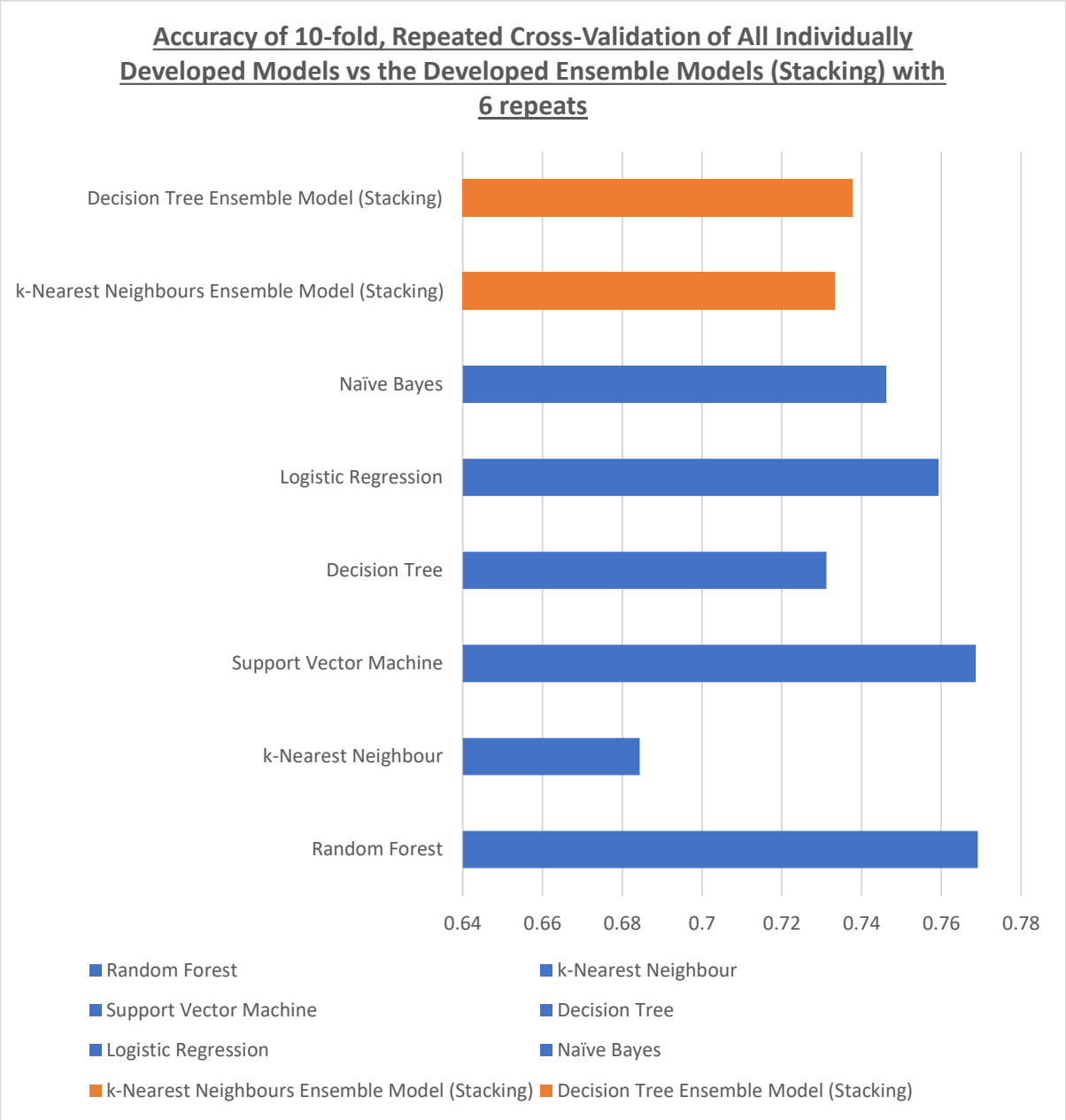
5.6. Comparison of Accuracy Metrix of 10-fold, Repeated Cross-Validation (RepeatCV) of Developed Ensemble Models (Stacking) based on the Two Least Accurate Developed Individual Models vs All Developed Machine Learning Models with 6 repeats

Table 5.6 indicates that the highest accurate model is Random Forest at 0.769167, meanwhile the lowest accurate model is k-Nearest Neighbour at 0.684333.

Table 5.6 :- Accuracy of 10-fold, Repeated Cross-Validation of All Individually Developed Models vs the Developed Ensemble Models (Stacking) with 6 repeats

Models	Accuracy
Random Forest	0.769167
k-Nearest Neighbour	0.684333
Support Vector Machines	0.768667
Decision Tree	0.731167
Logistic Regression	0.759333
Naïve Bayes	0.746167
k-Nearest Neighbours Ensemble Model (Stacking)	0.7332500
Decision Tree Ensemble Model (Stacking)	0.7376944

Graph 5.6 indicates that both the Ensemble Models (Stacking) highlight improvement over their counterparts of individual developed models which are Decision Tree and k-Nearest Neighbours.



Graph 5.6 :- Accuracy of 10-fold, Repeated Cross-Validation of All Developed Models vs the Developed Ensemble Models (Stacking) with 6 repeats

```
> print(knn_ensembleStack)
A knn ensemble of 2 base models: knn, rpart

Ensemble results:
k-Nearest Neighbors

6000 samples
   2 predictor
   2 classes: 'NO', 'YES'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 5400, 5400, 5400, 5400, 5400, 5400, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.7164444  0.2623256
  7  0.7272500  0.2854574
  9  0.7332500  0.2969904

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 9.
```

Figure 5.6.1 :- Snippet of results of k-Nearest Neighbours Ensemble Model (Stacking)

```
> print(DT_ensembleStack)
A rpart ensemble of 2 base models: knn, rpart

Ensemble results:
CART

6000 samples
   2 predictor
   2 classes: 'NO', 'YES'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 5400, 5400, 5400, 5400, 5400, 5400, ...
Resampling results across tuning parameters:

  cp           Accuracy   Kappa
  0.006666667  0.7376944  0.2893275
  0.026666667  0.7343333  0.3172651
  0.117777778  0.7195833  0.2159314

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.006666667.
```

Figure 5.6.2 :- Snippet of results of CART (Decision Tree) Ensemble Model (Stacking)

5.7 Discussion

In section 5.1, the True Negative, True Positive, False Negative & False Positive have been derived for all the developed Random Forest, k-Nearest Neighbour, Support Vector Machines, Decision Tree, Logistic Regression, and Naïve Bayes machine learning models for a total of 20 iterations and an average for each section have been tabulated (refer Table 5.1.1 to Table 5.1.6. The snippets of these results have also been added to the report to proof the tests, however, only Iteration # 1, 7, 15 and 20 have been added and the rest have been omitted to promote better readability of the report. In the original results which were obtained from the R IDE, the confusion matrix has two columns and two rows. As for the column, the header is named as Reference while as for the row, the header is named as Prediction. The confusion matrix as it name suggest might be confusing to interpret from the snippets, however it will be easier if "NO" is understood as positive class. This is because "NO" simply means non-default status, whereby "YES" simply means default status. The essentials of this information is that when a value falls under column "NO" and row "NO", it is True Positive, meaning it is correctly classified as non-default status. The opposite goes to column "YES" and row "YES", whereby it is True Negative, meaning it is correctly classified as defaulted status. As for the False Positives, the instances are predicted as non-default status whereby they are actually default status and this is known as Type I error. As for the False Negatives, the instances are predicted as default status whereby they are actually non-default status and this is known as Type II error.

In section 5.2, Graph 5.2 indicates that the False Positives from the developed models are higher than the False Positives in the literature. The False Negatives however, is lower in the developed models as compared to those obtained from literature. In section 5.3, the misclassification rate and accuracy has been obtained from Table 5.2 by using Equation 1 and Equation 2 respectively and the results have been tabulated in Table 5.3. As seen in Table 5.3, the two lowest accurate models are k-Nearest Neighbour at 0.71175 and Decision Tree at 0.71025. However, the lowest accurate models in the literature are Decision Tree at 0.7, Support Vector Machine and Logistic Regression at 0.725. The average accuracy of the literature models is also higher than of the develop models at 0.746 & 0.740, respectively,

whereas the average misclassification rate is lower compared to the developed models at 0.254 & 0.260, respectively.

Therefore, since there exist a discrepancy in misclassification rate and accuracy between the developed and literature models, a k-fold repeated cross-validation model evaluation method was implemented. In the beginning, the optimum and appropriate *n*-fold could not be determined, therefore, a middle ground of 10-fold was selected, and subsequently, *n* = {7,8,9,11,12,13,14} were applied to the experiment. The results of this experiment in section 5.4 were tabulated to Table 5.4. Graph 5.4 was computed, and based on it, the highest average accurate fold was determined to be 10-fold. During this step, the number of repeats were kept constant to 3 repeats, in order to promote fair and consistent results across all folds. Comparing Table 5.3, Table 5.4 and Graph 5.4, it is noted that the average accuracy of all developed models in Table 5.4/Graph 5.4 is still lower at 0.744 compared to 0.746 of the literature models in Table 5.3, however, it is higher than the average accuracy of all developed models in Table 5.3 by 0.004. Not only that, Table 5.4 and Graph 5.4 indicates that a higher number of *n*-folds does not necessarily mean a higher average accuracy of all developed models. Hence, it can be said that a 10-fold repeated cross validation at 3 repeats has improved the overall accuracy of all developed models.

In the hot pursuit to find the optimum repeats for all developed models, the same 10-fold repeated cross-validation model evaluation method was used, but this time with variable *n* repeats. Similar to the *n*-fold previously, the appropriate *n* repeats that would yield the highest accuracy could not be determined, therefore, the familiar 3 repeats was selected, and subsequently, *n* = {4, 5, 6, 7, 8, 9, 10} were selected. The results were tabulated in Table 5.5 and Graph 5.5 was computed. According to the results in Table 5.5/Graph 5.5, it is noted that the highest average accuracy is determined to be *n* = 6 repeats which stands at 0.743. During this step, the number of folds were kept constant to 10-fold, in order to promote fair and consistent results across all repeats. A point to be noted here is that there is an overall decrease in accuracy compared to the highest average accuracy of 0.744 in Table 5.4. In fact, the much similar *n* = 3 repeats in Table 5.5 had drop in average accuracy by 0.002 compared to Table 5.4. Not only that, Table 5.5 and Graph 5.5 indicates that a

higher number of *n* repeats does not necessarily mean a higher average accuracy of all developed models. It is believed whereby the variance of the repeated cross-validation model evaluation method had caused this occurrence. Nevertheless, the highest average accuracy yielding factors which are 10-fold, repeated cross-validation with 6 repeats are pointed out forward for the next experiment.

As for section 5.6, the accuracy of 10-fold, repeated cross-validation of the k-Neareast Neighbours ensemble model (stacking) and Decision Tree ensemble model (stacking) with 6 repeats were calculated and tabulated in Table 5.6. The results of all other individually developed models using 10-fold, repeated cross-validation with 6 repeats were also tabulated in Table 5.6. Graph 5.6 was computed to indicate the comparison between all the models. Based on Graph 5.6 & Table 5.6, it is clear that the k-Neareast Neighbours ensemble model (stacking) and Decision Tree ensemble model (stacking) had improved the accuracy of the two lowest individually developed models which are k-Nearest Neighbours at 0.684 and Decision Tree at 0.731. Based on Table 5.3, it is also clear that the k-Neareast Neighbours ensemble model (stacking) and Decision Tree ensemble model (stacking) had improved the accuracy of the two lowest individually developed models which are k-Nearest Neighbours at 0.71175 and Decision Tree at 0.71025. Based on Table 5.6, the highest accuracy amongst the both developed ensemble models happens to be Decision Tree ensemble model (stacking) at 0.738 compared to k-Neareast Neighbours ensemble model (stacking) at 0.733. However, based on Table 5.3 and Table 5.6, Random Forest is still the highest accuracy yielding model even before and after the 10-fold, Repeated Cross-Validation with 6 repeats. This is because Random Forest is an ensemble of independent decision trees (Gicić and Subasi 2018) and since it is a specific type of bootstrap aggregating bagging, bootstrapping procedures achieves better performance because it decreases the variance of the model without increasing the bias which explain why Random Forest achieves high accuracy compared to other developed models (Chen 2018).

In short, accuracy in this project simply means if a machine learning model can accurately classify a default status or non-default status based on the German credit data. Stacking ensemble method is proven to improve the accuracy of base classifiers as stacking trains a set of classifiers in parallel and then the learning is

done by a meta-learner (Armaki et al. 2017. The advantages of using stacking is that it can be used to form an ensemble model by combining several different classifiers in order to leverage their respective advantages, for example, in a two-layer stacking method, several different classifiers are trained in the first layer, and then the prediction results of the first layer are used as new explanatory features of stacked classifier in the second layer (Zhang et al. 2019). Therefore, stacking can be considered as a fully customizable hybrid machine learning system as it hosts various types of base and meta-classifiers to improve base classifiers' accuracy (Armaki et al. 2017).

6.0 Reflection

Being a part of this research paper happens to be the first time for me but will definitely would not be the last. Due to COVID-19, I had to study remotely, and this also means that I had choose a title remotely. Machine learning has always been an area that arouses my interests and that is why one day when I had woken up from my sleep, I had magically obtained a title which would involve credit scoring using machine learning methods. Another thing that interest me is also finance.

When I think of finance, I would usually think of those who are at a disadvantage in this world, those who could not afford basic amenities, or go to a college to obtain a degree and etc. Hence, my main motivation of doing this research is to dive into the world of finance and in one day hope to help problems that are faced by the disadvantaged.

Apart from that, studying during a COVID-19 is different from being in person at college as the experience is different. Not only that, I did this project over the period of less than 7 weeks. I had sought to obtain an extension due to a change in supervisor but was denied. I had also sought to do this research over the period of 14 weeks but was also denied in it. Finally, I had managed to complete this research, but I do feel that if I were given the privilege of time, I would explore more in this research.

Apart from that, I had grown to be familiar with the R programming language. I would like to thank R developers for making things easier for me to do this research less than 7 weeks. My future motivation is to be a R Open Source Developer. I had learned the Open Source skills from Open Source Development taught by Mr. Shahriman. Another skills that I have learned is from Intelligent Agents which was taught by Ms. Vasuky.

7.0 Conclusion

As a conclusion, I have successfully managed to answer Research Question 1 (RQ1) and Research Question 2 (RQ2). As for RQ1, the two lowest classification technique was apparently k-Nearest Neighbour and Decision Tree on a German credit data set. As for RQ2, the Stacking ensemble method had improved the accuracy of the k-Nearest Neighbour and Decision Tree model on a German credit data set. The Decision Tree ensemble model (stacking) at an accuracy of 0.738 and the k-Neareast Neighbours ensemble model (stacking) at an accuracy of 0.733 has improved the accuracy of the two lowest individually developed models which are k-Nearest Neighbours at 0.71175 & Decision Tree at 0.71025 before using 10-fold, Repeated Cross-Validation technique and also k-Nearest Neighbours at 0.684 & Decision Tree at 0.731 after using 10-fold, Repeated Cross-Validation technique. Stacking is suitable for ensemble application involving other base classifiers.

7.1 Research Contribution

This research is aimed to prove a point whereby Stacking is a suitable ensemble method used in a credit scoring application. The findings of this research is to convince financial institutions that just because a base classifier yields a low accuracy, that does not mean it is up to no good. Since Stacking can be used on such base classifiers, financial institutions are recommended to use this ensemble method for their credit scoring applications especially with other high accuracy models.

7.2 Future Work

As for the future works in this research, the 10-fold, Repeated Cross Validation technique could be applied on Australian, Iranian, Polish and Japanese datasets in order to evaluate other credit datasets to increase the estimate of mean model performance and to significantly avoid overfitting of data. In addition to that, Stacking should be used on a much higher accurate base classifier model such as Random Forest. This would then help to significantly increase the predicting capability of the credit scoring model.

In addition, feature selection could also be used to improve the overall accuracy of machine learning models. As mentioned in the literature review, there is almost an equal number of people who uses and do not uses feature selection. However, when using feature selection, it is crucial to consider a suitable feature selection method. For example, Forward Feature Selection, Backward Feature Elimination, and Exhaustive Feature Selection. Finally, multiple predictor classes can be selected to increase the information being displayed in Confusion Matrix.

8.0 References

ABID, L., Zaghdene, S., Masmoudi, A. and Ghorbel, S. (2017) "Bayesian Network Modeling: A Case Study Of Credit Scoring Analysis Of Consumer Loans Default Payment". Asian Economic And Financial Review [online] 7 (9), 846-857. available from <http://doi.org/10.18488/journal.aefr.2017.79.846.857> [10 November 2020]

Ala'raj, M. and Abbod, M. (2016a) "A New Hybrid Ensemble Credit Scoring Model Based On Classifiers Consensus System Approach". Expert Systems With Applications [online] 64, 36-55. available from <https://doi.org/10.1016/j.eswa.2016.07.017> [10 November 2020]

Ala'raj, M. and Abbod, M. (2016b) "Classifiers Consensus System Approach For Credit Scoring". Knowledge-Based Systems [online] 104, 89-105. available from <https://doi.org/10.1016/j.knosys.2016.04.013> [10 November 2020]

Ali, M. (2015) "Performance of Three Classification Techniques in Classifying Credit Applications Into Good Loans and Bad Loans: A Comparison".

Armaki, A., Fallah, M., Alborzi, M. and Mohammadzadeh, A. (2017) "A Hybrid Meta-Learner Technique For Credit Scoring Of Banks' Customers". Engineering, Technology & Applied Science Research [online] 7 (5), 2073-2082. available from <https://doi.org/10.48084/etasr.1361> [10 November 2020]

Bank Negara Guidelines (2010) available from <https://www.bnm.gov.my/documents/20124/963937/20100101_DFI_PL_0005.pdf> [1 February 2021]

Bequé, A., Coussement, K., Gayler, R. and Lessmann, S. (2017) "Approaches For Credit Scorecard Calibration: An Empirical Analysis". Knowledge-Based Systems [online] 134, 213-227. available from <https://doi.org/10.1016/j.knosys.2017.07.034> [10 November 2020]

Brownlee, J. (2020) Repeated k-Fold Cross-Validation for Model Evaluation in Python. [online] available from <https://machinelearningmastery.com/repeated-k-fold-cross-validation-with-python/> [1 February 2021].

Brownlee, J. (2018) A Gentle Introduction To K-Fold Cross-Validation [online] available from <https://machinelearningmastery.com/k-fold-cross-validation/> [1 February 2021]

Chen, Z. (2018) "The Application Of Tree-Based Model To Unbalanced German Credit Data Analysis". MATEC Web Of Conferences [online] 232, 01005. available from <https://doi.org/10.1051/matecconf/201823201005> [10 November 2020]

Chen, K., Yadav, A., Khan, A. and Zhu, K. (2020) "Credit Fraud Detection Based On Hybrid Credit Scoring Model". Procedia Computer Science [online] 167, 2-8. available from <https://doi.org/10.1016/j.procs.2020.03.176> [10 November 2020]

Chou, C., Hsieh, S. and Qiu, C. (2017) "Hybrid Genetic Algorithm And Fuzzy Clustering For Bankruptcy Prediction". Applied Soft Computing [online] 56, 298-316. available from <https://doi.org/10.1016/j.asoc.2017.03.014> [10 November 2020]

Dahiya, S., Handa, S. and Singh, N. (2015) "Credit Scoring Using Ensemble Of Various Classifiers On Reduced Feature Set". Industrija [online] 43 (4), 163-174. available from <https://doi.org/10.5937/industrija43-8211> [1 February 2021]

Dastile, X., Celik, T. and Potsane, M. (2020) "Statistical And Machine Learning Models In Credit Scoring: A Systematic Literature Survey". *Applied Soft Computing* [online] 91, 106263. available from <https://doi.org/10.1016/j.asoc.2020.106263> [1 February 2021]

García, V., Marqués, A. and Sánchez, J. (2014) "An Insight Into The Experimental Design For Credit Risk And Corporate Bankruptcy Prediction Systems". Journal Of Intelligent Information Systems [online] 44 (1), 159-189. available from <https://doi.org/10.1007/s10844-014-0333-4> [10 November 2020]

Gicić, A. and Subasi, A. (2018) "Credit Scoring For A Microcredit Data Set Using The Synthetic Minority Oversampling Technique And Ensemble Classifiers". Expert Systems [online] 36 (2), e12363. available from <https://doi.org/10.1111/exsy.12363> [10 November 2020]

Guarascio, M., Manco, G. and Ritacco, E., 2019. Knowledge Discovery in Databases. *Encyclopedia of Bioinformatics and Computational Biology*, [online] pp.336-341. Available at: <https://doi.org/10.1016/B978-0-12-809633-8.20456-1> [Accessed 1 February 2021].

Harris, T. (2015) "Credit Scoring Using The Clustered Support Vector Machine". Expert Systems With Applications [online] 42 (2), 741-750. available from <https://doi.org/10.1016/j.eswa.2014.08.029> [10 November 2020]

He, H., Zhang, W. and Zhang, S. (2018) "A Novel Ensemble Method For Credit Scoring: Adaption Of Different Imbalance Ratios". Expert Systems With Applications [online] 98, 105-117. available from <https://doi.org/10.1016/j.eswa.2018.01.012> [10 November 2020]

Lessmann, S., Baesens, B., Seow, H. and Thomas, L. (2015) "Benchmarking State-Of-The-Art Classification Algorithms For Credit Scoring: An Update Of Research". European Journal Of Operational Research [online] 247 (1), 124-136. available from <https://doi.org/10.1016/j.ejor.2015.05.030> [10 November 2020]

Livieris, I., Kiriakidou, N., Kanavos, A., Tampakas, V. and Pintelas, P. (2018) "On Ensemble SSL Algorithms For Credit Scoring Problem". Informatics [online] 5 (4), 40. available from <https://doi.org/10.3390/informatics5040040> [10 November 2020]

Louzada, F., Ara, A. and Fernandes, G. (2016) "Classification Methods Applied To Credit Scoring: Systematic Review And Overall Comparison". Surveys In Operations Research And Management Science [online] 21 (2), 117-134. available from <https://doi.org/10.1016/j.sorms.2016.10.001> [10 November 2020]

Masmoudi, K., Abid, L. and Masmoudi, A. (2019) "Credit Risk Modeling Using Bayesian Network With A Latent Variable". Expert Systems With Applications [online] 127, 157-166. available from <https://doi.org/10.1016/j.eswa.2019.03.014> [10 November 2020]

Nalić, J., Martinović, G. and Žagar, D. (2020) "New Hybrid Data Mining Model For Credit Scoring Based On Feature Selection Algorithm And Ensemble Classifiers". *Advanced Engineering Informatics* [online] 45, 101130. available from <https://doi.org/10.1016/j.aei.2020.101130> [1 February 2021]

Qu, Y., Quan, P., Lei, M. and Shi, Y. (2019) "Review Of Bankruptcy Prediction Using Machine Learning And Deep Learning Techniques". Procedia Computer Science [online] 162, 895-899. available from <https://doi.org/10.1016/j.procs.2019.12.065> [10 November 2020]

Saidi, M., Settouti, N., El Habib Daho, M. and El Amine Bechar, M. (2018) "Comparison Of Ensemble Cost Sensitive Algorithms: Application To Credit Scoring Prediction." in International Conference On Advanced Aspects Of Software Engineering (ICAASE18) [online] held 2018. CEUR Workshop Proceedings, 56-61. available from <http://ceur-ws.org/Vol-2326/paper7.pdf> [1 February 2021]

Soui, M., Gasmi, I., Smiti, S. and Ghédira, K. (2019) "Rule-Based Credit Risk Assessment Model Using Multi-Objective Evolutionary Algorithms". Expert Systems With Applications [online] 126, 144-157. available from <https://doi.org/10.1016/j.eswa.2019.01.078> [10 November 2020]

Thanawala, D., 2019. *CREDIT RISK ANALYSIS USING MACHINE LEARNING AND NEURAL NETWORKS*. Postgraduate. Michigan Technological University.

Tomczak, J. and Zięba, M. (2015) "Classification Restricted Boltzmann Machine For Comprehensible Credit Scoring Model". Expert Systems With Applications [online] 42 (4), 1789-1796. available from <https://doi.org/10.1016/j.eswa.2014.10.016> [10 November 2020]

Tripathi, D., Edla, D., Cheruku, R. and Kuppili, V. (2019) "A Novel Hybrid Credit Scoring Model Based On Ensemble Feature Selection And Multilayer Ensemble Classification". Computational Intelligence [online] 35 (2), 371-394. available from <https://doi.org/10.1111/coin.12200> [10 November 2020]

Wang, G., Ma, J., Huang, L. and Xu, K. (2012) "Two credit scoring models based on dual strategy ensemble trees". Knowledge-Based Systems [online] 26, 61-68. available from <https://doi.org/10.1016/j.knosys.2011.06.020> [Accessed 1 February 2021].

Wang, Y., Zhang, Y., Lu, Y. and Yu, X. (2020) "A Comparative Assessment Of Credit Risk Model Based On Machine Learning —A Case Study Of Bank Loan Data". Procedia Computer

Science [online] 174, 141-149. available from <https://doi.org/10.1016/j.procs.2020.06.069> [10 November 2020]

Wang, Z., Li, Y. and Zhou, X. (2016) "A Statistical Control Chart For Monitoring High-Dimensional Poisson Data Streams". Quality And Reliability Engineering International [online] 33 (2), 307-321. available from <https://doi.org/10.1002/qre.2005> [10 November 2020]

Xiao, H., Xiao, Z. and Wang, Y. (2016) "Ensemble Classification Based On Supervised Clustering For Credit Scoring". Applied Soft Computing [online] 43, 73-86. available from <https://doi.org/10.1016/j.asoc.2016.02.022> [10 November 2020]

Xiao, J., Zhou, X., Zhong, Y., Xie, L., Gu, X. and Liu, D. (2020) "Cost-Sensitive Semi-Supervised Selective Ensemble Model For Customer Credit Scoring". Knowledge-Based Systems [online] 189, 105118. available from <https://doi.org/10.1016/j.knosys.2019.105118> [1 February 2021]

Yan, J. and Han, S. (2018) "Classifying Imbalanced Data Sets By A Novel RE-Sample And Cost-Sensitive Stacked Generalization Method". *Mathematical Problems In Engineering* [online] 2018, 1-13. available from <https://doi.org/10.1155/2018/5036710> [10 February 2021]

Zhang, W., He, H. and Zhang, S. (2019) "A Novel Multi-Stage Hybrid Model With Enhanced Multi-Population Niche Genetic Algorithm: An Application In Credit Scoring". Expert Systems With Applications [online] 121, 221-232. available from <https://doi.org/10.1016/j.eswa.2018.12.020> [10 November 2020]

Zhang, W., Yang, D., Zhang, S., Ablanedo-Rosas, J., Wu, X. and Lou, Y. (2020) "A Novel Multi-Stage Ensemble Model With Enhanced Outlier Adaptation For Credit Scoring". Expert Systems With Applications [online] 165, 113872. available from <https://doi.org/10.1016/j.eswa.2020.113872> [10 November 2020]

Zhongguo, Y., Hongqi, L., Liping, Z., Qiang, L. and Ali, S. (2017) "A Case Based Method To Predict Optimal K Value For K-NN Algorithm". Journal Of Intelligent & Fuzzy Systems

[online] 33 (1), 55-65. available from <https://doi.org/10.3233/JIFS-161062> [10 November 2020]

Zhao, Z., Xu, S., Kang, B., Kabir, M., Liu, Y. and Wasinger, R. (2015) "Investigation And Improvement Of Multi-Layer Perceptron Neural Networks For Credit Scoring". Expert Systems With Applications [online] 42 (7), 3508-3516. available from <https://doi.org/10.1016/j.eswa.2014.12.006> [10 November 2020]

Appendix A:

Blackboard

SafeAssign Originality Report

INDIVIDUAL PROJECT • Step 1 : SafeAssign Checking

RAVEN THEVAR A/L BALASUBRAMANIUM -

Total Score:

Low risk 7 %

Submission UUID: f30bf0d2-64a1-76e4-8743-15a89ee3a66e

Total Number of Reports 1	Highest Match 7 % Raven_SafeAssign.pdf
Average Match 7 %	Submitted on 02/22/21 10:22 PM GMT+8
Average Word Count 23,038 Highest: Raven_SafeAssign.pdf	