UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Universität Paderborn
Fakultät für Wirtschaftswissenschaften
Department Wirtschaftsinformatik
Lehrstuhl für Wirtschaftsinformatik, insb. Data Analytics

Master Thesis

Active Learning in Credit Scoring

by

Joel Cedric Fotso Tenku
6810782
joelf@mail.uni-paderborn.de

submitted to
Professor Dr. Oliver Müller

Submitted on 15. February 2022

Statutory declaration:

I hereby declare under oath that I have done this work independently, without outside assistance and without using any means other than those indicated. Thoughts, tables, sketches, drawings, pictorial representations, etc. taken directly or indirectly from external sources (including electronic sources) are identified as such without exception. The thesis has not yet been submitted in the same or similar form or in part as part of another examination.

Paderborn, 15.02.2022

Joel Cedric Fotso Tenku

Abstract

The main objective of credit scoring models is to predict whether an applicant, if accepted, will repay on time or not. This can be done by estimating the probability of default repayment of loan applicants in order to decide which ones, of a large number of applicants to grant credit and which ones to decline. Such models are trained only on data from previously granted loans, where the repayment behavior has been observed. The repayment behavior of rejected applications cannot be observed. Therefore, the sample of rejected applications is discarded. Thus, the training data would not be representative of the general population of applicants and could bias the estimated scoring model. Such a biased score could lead to a misguided lending strategy or decision. Moreover, sample bias deteriorates the predictive performance of the credit scoring models. There are different methods to solve the sample bias problem in the credit scoring models. In this thesis, the active learning method will be investigated. In Active learning, learner strategies actively participate in selecting the specific instances from which to learn. In order words, learning algorithms will choose applications from the set of rejected applications (unlabelled applications) that will be able to increase the predictive performance of the credit scoring models when they are included into the training data, thus leading to reduce sample bias. These applications should then be granted a loan. This study first employs various credit scoring models, including Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors, Support Vector Machine, Naïve Bayes classifier, and Neural Networks. These models will be then investigated in the active learning process, where active learning scenarios such as Pool-Based and Stream-Based active learning scenarios with different learner strategies, namely Uncertainty Sampling, Density Weighting and Query By Committee, will be implemented. The sample for this study is the publicity real-world dataset of 50000 US mortgage borrowers, provided by International Financial Research. We then evaluate the performance of each credit scoring model by using performance metrics including the accuracy and the AUC. By modelling the active learning approach, a slight improvement in the performance of credit scoring models is achieved. This thesis investigates an approach that is different from those of various preceding studies to solve the sample bias problem in credit scoring models. Finally, we compare our results with work from different authors.

Keywords: Credit Scoring, Sample Bias, Active Learning, mortgage borrowers, International Financial Research, Pool Based Active Learning, Stream Based Active Learning, Uncertainty Sampling, Density Weighting, Query By Committee, Logistic Regression, Decision Tree, Support Vector Machine, K-Nearest Neighbors, Naïve Bayes, Random Forest, Neural Networks.

Table of Contents

List of Abbreviations

AUC	Area Under the Curve
ROC	Receiver Operating Characteristic
AUROC	Area Under ROC
BVP	Bivariate Probit model with selection
QBC	Query By Committee
KNN	K-Nearest Neighbors
SVM	Support Vector Machine
NN	Neural Networks
NB	Naïve Bayes classifier
LR	Logistic Regression
RF	Random Forest
DT	Decision Tree
IFR	International Financial Research
MLE	Maximum Likelihood Estimation
PCC	Percentage of cases Correctly Classified
LDA	Linear Discriminant Analysis
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
KS	Kolmogorov-Smirnov Statistics
KL	Kullback-Leibler
FICO	Fair Isaac Corporation
LTV	Loan to Value
GDP	Gross Domestic Product
FPR	False Positive Rate
TPR	True Positive Rate
PG	Partial Gini-index

BS	Bier-Score
MLP	Multilayer Perceptron
WESML	Weighted Exogenous Sample Maximum Likelihood Estimator
CART	Classification And Regression Trees

List of Figures

List of Tables

1 Introduction

With the increasing demand for credit in recent years, financial organizations are similarly facing the problem of differentiation between good borrowers (those who are likely to repay the credit) and bad borrowers (those who have a high probability of default repayment) to granting credit (West, 2000). This has created the need for a system that will be able to predict the riskiness of lending to a credit applicant in order to allow financial institutions to make decisions about whether or not to grant a loan. For this purpose, the credit scoring method was introduced. The credit scoring method has become essential in the field of credit risk and is widely used by financial institutions and lenders to predict the risk of default that a borrower will fail to make the required payments (Thomas, Crook & Edelman, 2017). Moreover, it provides other benefits including improving accurate credit assessment, improving efficiency of credit agents and credit decisions, reducing human bias in lending decisions, reducing the cost of the credit process, time and effort, ensuring an effective and accountable management of financial risks while enhancing the credit management process and quantification of expected losses (Blöchlinger & Leippold, 2011).

The main objective of the credit scoring method is to assess the creditworthiness of an applicant in order to reduce the risk associated with a bad loan (Hand & Jacka, 1998). In other words, credit scoring is mainly used to predict the probability of repayment by credit applicants. The creditworthiness reflects the level of credit risk and helps banking institutions to decide whether an application of the individual should be approved or declined. For this, credit information of the applicant from past transaction must be collected first, and then used to estimate the probability of defaulting. This probability indicates the creditworthiness of loan applicant and is used for lending decision (Hand & Henley 1997). Thus, the decision whether or not to lend credit is taken by comparing the estimated probability of defaulting with a cut-off score. If the probability of defaulting is higher than the cut-off score, the applicant will therefore get the loan, while the application will be rejected if the probability of defaulting is lower than the cut-off score (Hand & Henley 1997). Generally, the lower the probability of defaulting of the applicant, the lower the risk of non-payment of credit and vice versa (Hand, 2005). This process is widely used in personal credit cards, consumer loans and mortgages (Einav, Jenkins & Levin., 2013). For building borrower scores, various statistical models such as linear discriminant analysis, linear regression, random forests, gradient boosting, neural networks can be used (Henley, 1995).

Furthermore, there are two decision-making processes for credit scoring used by banks, namely subjective scoring, and statistical scoring (Schreiner, 2003). The difference between them is that subjective scoring provides a qualitative judgment based on the input of an expert, while statistical scoring provides a prediction of risk as a probability, based on rules and statistical methods.

However, the credit scoring model suffers from sample bias because it is trained on a sample of applications which is not representative of the whole population of applicants (Banasik, Crook & Thomas, 2003). Indeed, credit scoring models are trained on samples of previously accepted applications only, where the repayment performance of applicants has been observed (Heckman, 1979). The repayment behavior of the rejected applications is not observed, and they are therefore discarded, thus creating sampling bias. Hence, the modelling of credit scoring models results in biased parameters of the population model, which could therefore lead to a misguided lending strategy or decision. Different approaches have been proposed by various authors to solve sample bias in credit scoring models. Boyes, Hoffman and Low (1989) for example proposed the use bivariate probit model with two sequential events as the dependent variables. This model measures the ability of the borrower to repay the loan and the decision to grant the loan or not depend on this ability. Hand and Henley (1994) proposed the use of the Reject inference. Greene (1998) developed a similar binary choice model for sample selection that is relevant for the modelling of credit scoring by commercial banks. Jacobson and Roszbach (2003) also followed the same methods. Banasik, Crook and Thomas (2003) presented a sample selection correction using a bivariate probit model with selection (BVP) to reduce the sample bias. Nevertheless, the most common way used to address the sample selection problem is the reject inference method (Barakova, Glennon & Palvia, 2013).

1.1 Purpose

This thesis lies in making use of the active learning approach to solve the sample bias problem that arises by building credit scoring models on accepted applications only. With this approach, the set of rejected applications (unlabelled applications) will be explored and the most informative will be selected to be included into the training set. We investigate two active learning scenarios, namely Pool-based and Stream-based active learning. In each scenario, we implement various query strategies including Uncertainty sampling, Query By Committee and Density Weighting. In a next step, credit scoring models such as Linear Regression, K-Nearest Neighbors, Support Vector Machine, Naive Bayes classifier, Decision Tree, Random Forest, and Neural Networks will be explored in each query strategy. Furthermore, we shall evaluate the performance of each model by computing the accuracy and the AUC. In this thesis, we are using the publicity real-world dataset of 50000 US mortgage borrowers provided by International Financial Research (IFR). IFR is a website that provides advise in area of risk management and prudential regulation of financial institutions such as data science, data collection, credit risk analytics, regulation compliance, etc. Furthermore, it provides also financial data for credit scoring applications, of which the most popular and free data examples are Home Equity Loans (HMEQ), Mortgage, Loss Given Default (LGD), and Ratings database.

1.2 Thesis Outline

The first section of the paper introduced credit scoring and some of the techniques used in the modelling of this approach. Furthermore, it explained the problem related to these techniques and highlighted some of the methods used to solve this problem. The second section will introduce the background including the description of the problem and the related work. The third section will present the credit scoring models and the active learning method for solving the sample bias problem in credit scoring models. In addition, it will discuss metrics that can be used to evaluate the performance of these models. In the fourth section, the methodology used for the data preparation will be outlined and missing data and some feature selection techniques will be discussed. Moreover, Python was used throughout the entire pre-processing and modelling stages. The fifth section will present the results after applying the credit scoring models to our dataset. The sixth section will discuss the results of the models and compare them with those of related work. The seventh and final section will conclude our paper.

2 Background

2.1 Problem Description

Credit scoring is one the most important tools used by financial institutions to support the decision-making process (Al Amari, 2002). It also minimizes the credit process costs and efforts, allocates credit to good borrowers, differentiates between good and bad credit (Kim & Sohn, 2004) and predicts the bad creditors (Lim & Sohn, 2007). The decision on whether or not to grant the credit is made on the basis of the comparison between the threshold score and the score of the application that is estimated using the credit scoring models (Banasik, Crook & Thomas, 2003). Indeed, the threshold score is determined in several ways. In the first way, Banks either assign a minimum score for an accepted loan application or calculate it. The second way is the one most used by banks (Hsai, 1978). It is based on the historical data and the repayment behavior of the loan by the applicants. If the score of the application is smaller than the threshold, the application is rejected, otherwise it is accepted (Hand & Henley, 1997). However, the sample for training credit scoring models is not representative of the general population of applicants, thus leading to a biased estimate (Banasik, Crook & Thomas, 2003). Indeed, the credit scoring models are only based on applications where the repayment behavior has been observed. In other words, they are trained on a sample of applications that previously were accepted and for which the repayment behavior is known (Crook, Edelman & Thomas, 2007). Applications that have been rejected are not taking into consideration. Therefore, this could bias the estimated credit scoring model and thus deteriorates the predictive performance. Generally, the sample bias occurs when a randomly selected sample from the entire population is used for model estimation

(Heckman, 1979). As a consequence, the question arises, how the sample bias problem in the credit scoring models can be resolved? For this purpose, the active learning method will be investigated in this thesis to solve this problem. Active learning consists of selecting and adding into the training set the most informative rejected applications that can increase the predictive performance of credit scoring models (Settles, 2010), thus leading to reduce sample bias. In the next section, different approaches developed by the authors to solve sample bias in credit scoring models will be explored.

2.2 Related Work

Many authors have studied the sample bias problem in credit scoring models and have proposed various methods for solving it. In this section, we provide a retrospect of these previous studies. Table 2.1 summarizes relevant literature about the topic sample bias. These texts have been found by mean of the Google Scholar search engine and databases including semantic scholar, science direct, the Association for Computing Machinery (ACM), JSTOR, and academia and as well as from keywords such as “credit scoring problem”, “credit rating problem”, “bias”, “sample selection”, “sample bias”, “sample selection bias”, “sample bias solution”, “credit scoring model and sample selection bias”.

Concepts / Articles	Algorithms	Description of algorithms	Datasets	Variable selections
An Econometric Analysis of the Bank Credit Scoring Problem (Boyes, Hoffman & Low, 1989)	Bivariate Probit model with WESML	This approach is based on two events, namely the decision for granting loan or not and the ability of the applicant to repay it.	Credit Card data set from a single large institution between 1977 and 1980.	T-statistics, probability of granting and defaulting
Can reject inference ever work? (Hand & Henley, 1993/4)	Reject inference: Extrapolation, distribution, and methods using supplementary information	They attempt to infer the true creditworthiness of rejected applications that will then be added to the accepted applications in order to minimize sample bias.	Accept and Reject sample from the County Court judgement.	Probability of granting
Reject inference applied to logistic regression for credit scoring (Joanes, 1993/4)	Reject inference: the iterative reclassification and the Augmentation	Augmentation used to make a distinction between good and bad accepted applicants by an estimate of the probability of acceptance for	Artificial Sample with accepted and rejected cases.	Threshold, Posterior probability

		each applicant; and the iterative reclassification is used for classifying rejected applicants (good or bad) based on accepted applicants' rule.		
Sample Selection in credit-scoring models (Greene, 1998)	Sample selection models: Binary choice model, Regression model with sample selection, model for the number of occurrences.	The binary model indicates whether an applicant is defaulting or not; the regression model with sample selection combines the correlation between spending behavior and default behavior; the model for the number of occurrences explores the most significant variables, the number of derogatory reports in an applicant's credit history.	Credit Card dataset generated by a major credit-card vendor in 1991.	Correlation, t-ratio
Bank lending policy, credit scoring and value at-risk (Jacobson & Roszback, 2003)	Bivariate Probit approach with MLE	It is used for estimating an unbiased credit scoring model, which is used for evaluating the credit risk of the portfolio through the Value at Risk (VaR) measure.	Loan applicant dataset from the Swedish lending institution between September 1994 and August 1995.	Univariate relation with the target variable, correlation, t-statistic
Sample Selection bias in credit scoring models (Banasik, Crook & Thomas, 2003)	Bivariate probit model with selection (BVP)	The BVP improves performance prediction of a good-bad model (based only on accepted applications) by selecting the most correlated variables (the most significant variables)	Sample of application for a credit product collected in a fixed period within 1997; Sample of Scottish, English, and Welsh applicants.	Correlation between variables, probability between variables and the target variable.

Does reject inference really improve the performance of application scoring models? (Crook & Banasik, 2004)	Reject inference: Re-weighting (with weights of evidence) and Extrapolation method	In re-weighting assigning, an accept-reject models is estimated using accepted or rejected cases and then used to decide which applicants and variables can be used for estimating good-bad model. The extrapolation is used to infer a good-bad classification from a model estimated using accepted applicants only.	Sample of Scottish, English, and Welsh applicants; Sample of applicants for credit within a fixed period during 1997.	Correlation
Credit Scoring and the sample selection bias (Parnitzke, 2005)	Reject inference methods: enlargement, reweighting and extrapolation	These techniques try to infer missing information (creditworthiness) for the rejected applicants based on the behavior of accepted applicants.	Real data set from Fahrmeir and Hamerle.	Correlation
The impact of sample bias on consumer credit scoring performance and profitability (Verstraeten & Poel, 2005)	Reject inference namely the augmentation and iterative reclassification	In the augmentation, accepted cases are weighted inversely proportional to the probability of accepted cases in order to make a distinction between good and bad accepts and the iterative reclassification consists of using classification rule to classify rejected cases.	Consumer credit data from Belgian direct-mail-order company.	Leap-and-bound algorithm with a likelihood score χ^2
Reject inference, augmentation, and sample selection (Banasik & Crook, 2007)	weighted logistic regression (Augmentation), BVP, Weighted BVP	The BVP corrects the model (probit model or logistic regression) by including variables that may affect the probability of a case being accepted, and the	Sample of Scottish, English, and Welsh applicants.	Correlation

		weighted logistic regression is mainly used to correct the misspecification problem of the model that arises when rejected cases are present.		
Credit Rating Models Considering Sample Selection Biases (Fan, Yang & Zhang, 2008)	Bivariate Probit Model with Maximum Likelihood Estimator (MLE)	It improves the precision of estimated parameter made from probit model and identify the variables that affect the precision of the default rate.	Commercial loans data from China's commercial bank collected between December 1991 and February 2004.	Probability between variables and the target variable, Correlation
Sample Selection Bias in Acquisition Credit Scoring Models: An Evaluation of the Supplemental-Data Approach (Barakova, Glennon & Palvia, 2013)	Method of proxy credit performance for rejected applications.	This method shall allow to infer how rejected applications, if accepted, would affect the performance of the credit scoring model.	Nationally representative sample of credit bureau records with a large number of credit card applicants during the period 1999 to 2009.	Acceptance rate

Concepts / Articles	Features of the dataset	Evaluation metrics	Limitation of the procedure
An Econometric Analysis of the Bank Credit Scoring Problem. (Boyes, Hoffman & Low, 1989)	The dataset contains 4632 credit card applicants of whom 3711 were granted credit and 921 were denied credit. It comprises 42 variables for model estimation divided into three parts namely personal characteristics with 12 variables, Economic variables with 16 variables and financial variables with 14 variables.	Precision, sensitivity, accuracy	Estimate is low when data set is limited
Can reject inference ever work? (Hand & Henley, 1993/4)	The sample of a number of weeks since the last County Court judgement graded as: 1 (1-26 weeks), 2 (27-52 weeks), 3 (53-104 weeks), 4 (105-208 weeks), 5 (209-312 weeks), and 6 (greater than 313 weeks, or no Country Court judgement.	Predictive performance	Computationally more expensive

Reject inference applied to logistic regression for credit scoring. (Joanes, 1993/4)	Sample of 57 'goods' (coded y=1), 18 'bads' (y=0), and 12 rejects (y=2) with three variables : age, years, account and accommodation status.	Classification accuracy, sensitivity	These methods take behavior of the acceptations to represent the characteristics of the rejects, which is not sufficiently.
Sample Selection in credit-scoring models. (Greene, 1998)	The dataset consists of 13444 applications (of which 10499 were accepted) and includes 63 variables divided into 10 categories (such as market data, credit bureau data, expenditure, demographics variables….) were used for model estimation.	Predictive performance	They does not guarantee an improvement in the performance.
Bank lending policy, credit scoring and value at-risk. (Jacobson & Roszback, 2003)	The dataset consists of 13338 applications with 57 variables of which 16 variables (such as age, male, income, loansize, house, divorce, bigcity, balinc, limit, zerolimit, entrepr…) were used for model estimation.	KS, confusion matrix	Expensive to analyze information from rejected applicants.
Sample Selection bias in credit scoring models (Banasik, Crook & Thomas, 2003)	The first dataset consist of 8139 applicants and includes 40 variables (age, time…). Weights of evidence transformation of explanatory variables and the binary variables were used for model estimation: The second dataset has 2540 Scottish and 9668 English and Welsh with 48 variables of which 30 variables (weights of evidence of variables) were used for model estimation.	AUROC; confusion matrix	It suffers from missing data (Mislevy, 1991).
Does reject inference really improve the performance of application scoring models? (Crook & Banasik, 2004)	The first sample consists of 2540 Scottish applicants (of which 1543 classified as good and 997 as bad) and 9668 English and Welsh applicants (of which 6439 classified as good and 3229 as bad). It contains 46 variables (age, time, bureau variable..). The second sample has 8139 cases of which 5413 cases accepted and 2726 cases denied.	PCC, ROC, AUROC	They do not improve the performance of the classification model.
Credit Scoring and the sample selection bias (Parnitzke, 2005)	The data set consists of 1000 cases, of which 700 cases are good and 300 cases are bad, with 20 explanatory variables and one target variable (credit status), of which only 7 variables were used for model estimation.	Predictive performance, confusion matrix	Model improvement depends on the specificity of the dataset.
The impact of sample bias on consumer	The dataset contains 36039 accepted cases of which 629 cases are default and 5471	PCC, AUC, profitability	Model improvement depends on the

credit scoring performance and profitability (Verstraeten & Poel, 2005)	rejected cases of which 107 cases default and 45 variables (such as demographic variables, financial variables default variables…). Of those 31 variables were selected for model estimation.		specificity of the dataset.
Reject inference, augmentation, and sample selection (Banasik & Crook, 2007)	Sample of 2540 Scottish (of which 1543 good and 997 bad) and 9668 English and Welsh (of which 6439 classified as good and 3229 as bad). It contains 26 variables (such as bureau variables, proprietary variables, age, occupation). Of those, 16 variables were used for model estimation.	Accuracy, AUROC	Model improvement depends on the specificity of the dataset.
Credit Rating Models Considering Sample Selection Biases (Fan, Yang & Zhang, 2008)	The dataset contains 2798 granted loans and 299 rejected loans. It includes 38 variables. Of those, 11 variables (default, rate, loansize, estate, mortgage, maturity, approval, rmb, capt, comty, comown) were used for model estimation.	Chi2, McFadden R^2	It provides better results when the dataset is large.
Sample Selection Bias in Acquisition Credit Scoring Models (Barakova, Glennon & Palvia, 2013)	The data set contains information from grantors of consumer credit and collectors of public records and variables divided into five parts: (credit amount variables, credit type variables, new credit variables, payment variables, credit score variables).	KS, forecast accuracy	Costly due to obtaining the bureau data; it does not totally eliminate sample bias.

Table 2.1: Literature review.

3 Models Presentation

In this section, we first present different credit scoring models and then active learning strategies, which will be used to solve the sample bias problem in credit scoring models.

3.1 Credit scoring models

Credit scoring models are widely used by financial institutions in order to distinguish good borrowers from bad borrowers. The process of credit scoring consists of classifying the applicants into two classes namely the good and bad class. Several machine learning algorithms have been used for building credit scoring models. The most widely used are the Logistic Regression, Support Vector Machines, Decision Trees, Neural Networks, K-Nearest-Neighbors, and Naïve Bayes classifier (Crook, Edelman & Thomas, 2007). These models will be applied in this study on the mortgage data set.

3.1.1 Logistic Regression

The Logistic Regression (LR) model is the most popularly used statistical technique in credit scoring applications (Hand & Henley, 1997). It is a parametric method used to assess the relationship between a dependent categorical variable (output) and one or more independent variables (predictors) (Cox, 1958). To do this, it uses the logistic function method, which evaluates the probabilities of an output variable. Moreover, the MLE method is used to perform most logistic regressions with more than one independent variable (Freund & William, 1998). Thus, the output variable in the LR model can be binary (two classes), multinomial (more than two classes) or ordinal. The logistic function is given by the formula (Akindaini, 2017):

$$f(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

where k represents the number of independent variables. Thus, the LR model for default loan can be formulated as follows:

$$\frac{f(x)}{1 - f(x)} = e^{-(\beta_0 + \beta_1 x + \cdots + \beta_k x_k)}$$

3.1.2 Decision Tree

The Decision Tree (DT) model is a tree in which each internal node is classified with an input factor and each leaf node is labeled with the class or probability over the classes *(*Rokach & Mai, 2008). Still known as recursive partitioning (Hand & Henley, 1997) or CART, the DT model is mainly applied for building prediction models from data. Moreover, a classification tree is a non-parametric technique that analyses dependent or categorical variables as a function of continuous explanatory variables (Breiman et al., 1984). It takes a finite number of unordered values. On the other hand, a regression tree is designed for dependent variables that take continuous values (Loh, 2014). The tree is thus constructed by recursively partitioning the data at each node on the basic of an input. Then, the best partition is selected from all possible partitions and the optimal sub-tree is the one with the lowest cost of misclassification (Zekic-Susac, Sarlija & Bensic, 2004).

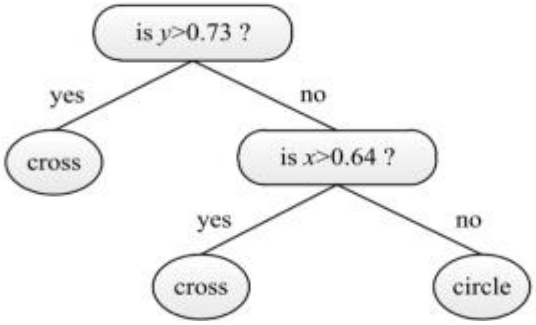

Figure 3.1: Simple example of decision tree

Source: Ensemble Methods Foundations and Algorithms (Zhou, 2012).

One of the main problems of the DT model is that the relevance of features for the classification is not considered. To solve this problem, the C4.5 algorithm was introduced (Salzberg, 1994).

3.1.3 K-Nearest Neighbors

K-Nearest-Neighbors (KNN) is one of the most widely approaches used for the classification (Fix & Hodges, 1952). Still called Lazy learner, the KNN classifier is one of the oldest machine-learning non-parametric techniques (Altman, 1992). It is a distance-based algorithm. In the KNN process, each object is classified by several votes of its neighbours. In other words, the class with the highest number of votes or frequency among its k nearest neighbours is that of the object (Fix & Hodges, 1989). The selection of the suitable distance metric is an important step in the process. As distance metrics in the KNN model, we have for example Euclidean, Manhattan, Chebyshev, and Hamming distance. The KNN process can be described as follows: a KNN learner selects the k instances from the training set that are closest to the test instance first, and then the test instance will be classified to the majority class among the k instances (Fix & Hodges, 1989).

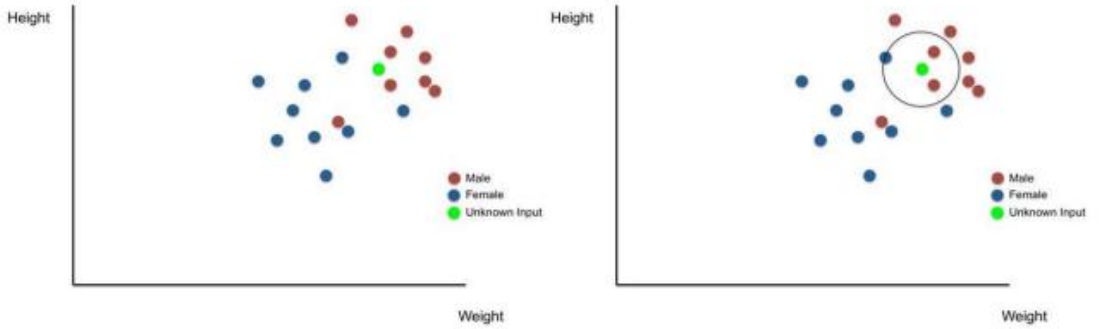

Figure 3.2: Classification of an instance by a 3-nearest neighbor classifier

Source: Brilliant

3.1.4 Support Vector Machine

The Support Vector Machine (SVM) model is a supervised machine learning algorithm for classification and regression (Cortes & Vapnik, 1995). The SVM process can be described as follows: first, it finds a hyper-plane that separates training observations to maximize the margin (smallest vertical distance between observations and the hyper-plane). Then, the optimal hyper-plane is determined on the basis of observations within the margin called support vectors (Cortes & Vapnik, 1995). Hence, observations outside the margin do not influence the hyperplane (Bagherpour, 2017). Thus, from the data training with two classes, SVM can build a model that classifies the data test into one or the other. However, the use of SVM poses two main problems: the choice of the optimal input feature subset and the best kernel parameters (Huang, Chen & Wang, 2007).

Furthermore, the choice of the subset of features has an impact on the best kernel parameters and vice versa (Fröhlich & Chapelle, 2003)

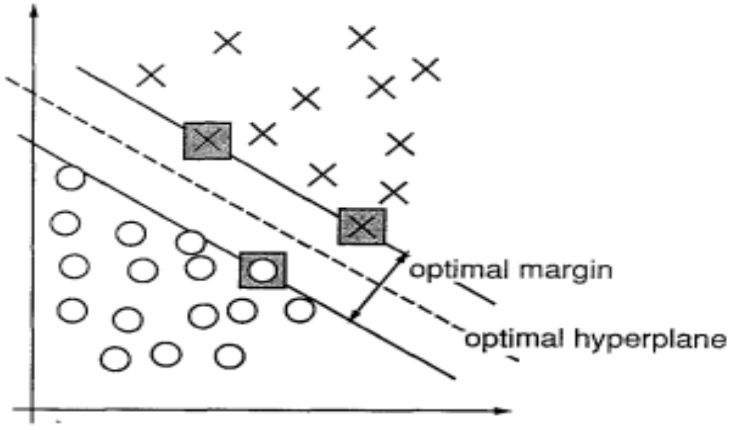

Figure 3.3: The margin maximization in SVM

Source: Support-Vector Networks (Cortes & Vapnik, 1995).

3.1.5 Naïve Bayes classifier

The Naïve Bayes (NB) classifier is a simple probabilistic classifier, generally used to predict the probability that a given sample belongs to a particular class (Domingos & Pazzani, 1997). It is based on the Bayes' theorem (also known as class conditional independence), which assumes that the features are independent of each other (Rish, 2001). Using the Bayes' theorem, the NB classifier can be formulated as follows (Rish, 2001):

$$p(C_k|X) = \frac{p(C_k)\, p(X|C_k)}{p(X)}$$

$p(C_k|X)$ is a posteriori probability of X (probability of event C_k, given the event X is true). $p(C_k)$ is the priori of C_k (the prior probability, i.e., Probability of event before evidence is seen. $p(X)$ is the evidence. $p(X|C_k)$ the likelihood.

3.1.6 Random Forest

The Random Forest (RF) model is an ensemble of decisions trees in which each tree is constructed and trained independently from randomly selected samples (Breiman, 2001). Moreover, the distribution of samples is similar for all trees in the forest (Breiman, 2001). In the RF model, the n-tree sample is randomly generated from the training data. And for each sample, the best split is selected based on the number of randomly selected variables at each split (Tan, Yan & Zhu, 2019). So, the outcome of each tree represents the class that is the most common of the classes. The class with the highest votes will thus become the prediction of model (Tan, Yan & Zhu, 2019). The main advantage of using the RF model is that it can evaluate the importance of the variables by ranking the performance of each variable, which can be done by estimating and scrambling the predictive value of variables (Akindaini, 2017).

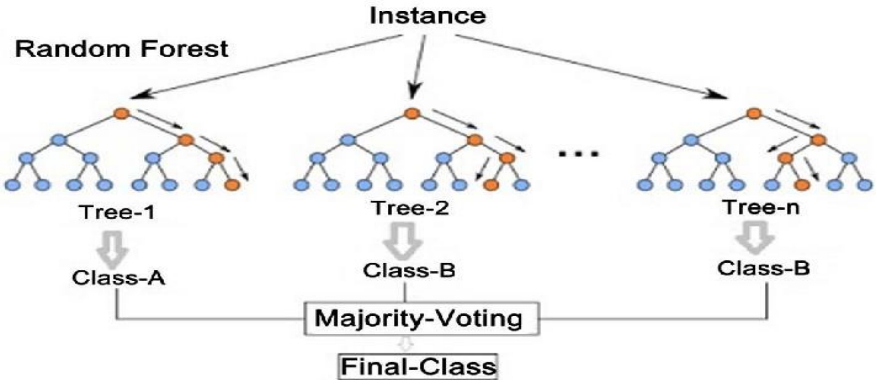

Figure 3.4: Random Forest

Source: LearnOpenCV.

3.1.7 Neural Networks

The use of Neural Networks (NN) has increased considerably in recent years in demand forecasting applications (Zhang et al., 1998). In the NN, the training process and linear or non-linear variables are employed to identify variables that can be used for a better decision-making outcome (Abdou & Pointon, 2011). The prediction of the target in the NN depends on the number of input variables, the number of hidden layers, and the different learning rates (Okasha, 2014). Thus, the NN, also known as multilayer perceptron (MLP), consists of an input layer, one or more hidden layers and an output layer. The input layer consists of nodes of variables (for example applicants' characteristics) and the output layer corresponds to the target variable (for example creditworthiness). Thus, information transmitted to the input nodes propagates through the hidden layers to provide an output to the output node. The nodes in the layers are connected by links with variable weights, which are estimated using the backpropagation rule (Rumelhart, McClelland & Williams, 1986). Furthermore, the predicted output value is compared with the observed value and the difference between them is measured by using the error or cost function (Okasha, 2014).

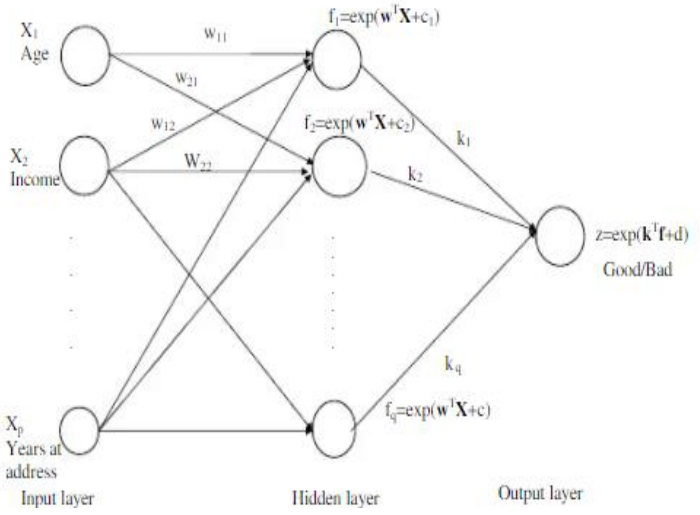

Figure 3.5: Neural Network

Source: Recent developments in consumer credit risk assessment (Crook, Edelman & Thomas, 2007).

3.2 Active Learning method

The recent proliferation of available digital data, which are generally unlabeled, has increased the need for labeling. Labeling of data must be done manually and is therefore a time consuming and expensive task in itself (Settles, 2010). To address this challenge, the active learning method was introduced (Ilhan & Amasyali, 2014). Active learning (Settles, 2010) is a semi-supervised learning approach that aims to achieve high prediction accuracy with as few training labels as possible. It is based on the idea that a classifier works best if it can select the queries and data from which to learn itself. Thus, a learning algorithm learns from features of labeled instances, interactively selects then the most informative samples from a large set of unlabeled instances, queries their class label by the oracle so that the accuracy can be maximum, and adds them finally into the training process (Beygelzimer, Dasgupa & Langford, 2009). Furthermore, the more informative instances are added into the training set, the more the predictive performance of the classifier increases (Ghasemi et al., 2011). Some criteria, such as uncertainty sampling, Query by Committee (QBC), Expected Gradient Length, and Variance Reduction, can be used to measure the informativeness of an instance (Fu, Zhu & Li, 2012). In general, choosing the most informative samples from the unlabeled set is an essential step in Active learning. This increases the performance of the model and reduces the computational cost for labeling (Zhu & Whu, 2006). Therefore, the active learning method is very useful in reducing the number of labeled data needed to achieve a given level of performance (Thompson, 1999). The process of applying active learning is summarized in Figure 3.6:

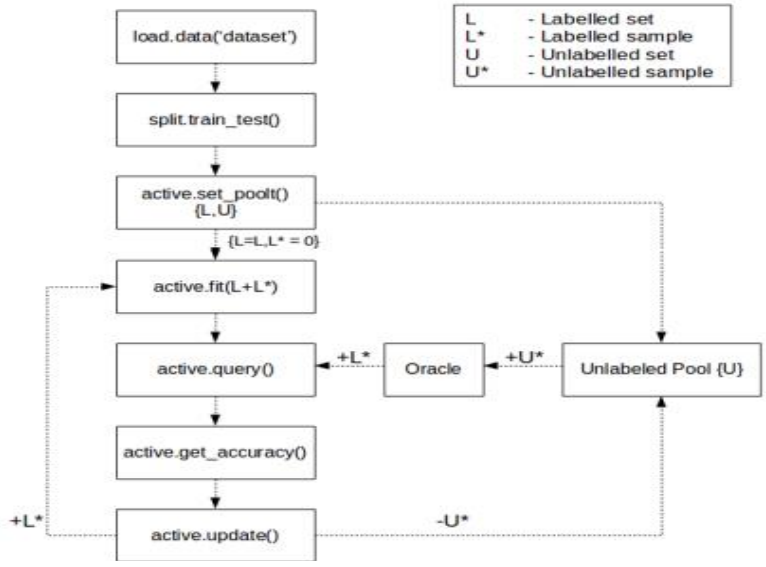

Figure 3.6: Active learning model

Source: An analysis of active learning strategies for sequence labeling (Settles & Craven, 1998).

Figure 3.6 provides a general overview of how active learning works. The procedure starts by loading and splitting the data into train and test set. Then, a pool of labeled instances L and unlabeled instances U is determined. After that, a model is trained from the current labeled set L. In a next step, all instances in the pool of unlabeled instances U are queried by the learner and the most potential instances U* based on selection criteria are selected to be labeled by an oracle. These instances are then removed from the pool of unlabeled instances. After that, the new labeled instances L* are added to the training set L to update the accuracy. This process is repeated until the desired accuracy of the model is achieved.

Figure 3.7 shows the active learning algorithm. It generally consists of a two-step process, namely an initialization step and a close-loop step of query and retraining (Seung, Opper & Sompolinsky, 1992). First, a small number of samples from a set of labeled samples is randomly selected and used as a training set. Then, this training set is used to train the classifier. After that, the following loop can be performed until the stopping criteria are satisfied: a set of unlabeled instances is queried, and a class label is assigned to each of them. Then, the most informative unlabeled instance from the unlabeled pool is selected, labeled by an oracle, and added into a training set. After that, the added instance is removed from the unlabeled pool. Finally, the classifier is retrained using the new training set.

Input: initial small training set *L*, and pool of unlabeled data set *U.*

Use *L* to train the initial classifier *C*

Repeat

- Use the current classifier *C* to label all unlabeled examples in *U*
- Select the most informative unlabeled example from the unlabeled pool U and ask oracle H for labeling.
- Augment L with this new example, and remove it from U
- Use L to retrain the current classifier *C*

Until the predefined stopping criterion SC is met

Figure 3.7: General active learning algorithm

Source: Active learning With Sampling by Uncertainty and Density for Data Annotation (Zhu et al., 2010).

In Active learning, one distinguishes three different scenarios in which various query strategies can be implemented. In the following sub-sections, these scenarios will be explored.

3.2.1 Active Learning Scenarios

There are three main types of scenarios in active learning (Settles, 2010): Membership Query Synthesis, Stream-Based Selective Sampling and Pool-Based Sampling. In this thesis, two of them, namely the stream and pool based active learning, will be investigated.

3.2.1.1 Stream-Based Selective Sampling

Also known as selective sampling (Cohn, Atlas & Ladner, 1994) or sequential active learning, it is based on the key assumption that obtaining an unlabeled instance is free and its query is not expensive. The unlabeled instance is first queried and can then be labeled or not. In the stream-based selective sampling, there are several ways to determine whether the instance can be labeled or not. These methods rely on the evaluation of informativeness measure (Dagan & Engelson, 1995). On the basis of this measure, the informative instance is labeled while the less informative instance is not labeled (Dagan & Engelson, 1995). Thereby, the process of active learning in stream-based starts by querying the pool of unlabeled instances using query strategies and then the most informative of them will be selected for labeling (Settles, 2010). Based on the approach of informativeness measure, a minimum threshold is used to evaluate the informativeness of the unlabeled instance. If the assessment of the instance exceeds the threshold, it will be queried and labeled (Dagan & Engelson, 1995). After that, instances that have been selected for labeling are added one by one into the training set (Settles, 2010). This scenario uses different thresholds depending on query strategies for evaluating unlabeled instances. The stream-based selective sampling is an appropriate way for weighting the informativeness of the unlabeled instances (Settles, 2010). However, the set of multiple thresholds may be regarded as a disadvantage (Sun & Wang, 2010). Figure 3.6 illustrates the process of active learning in stream based selective sampled. This process starts with a given labeled instances L, the pool of unlabeled instances U, a number of queries B, and the query strategy to measure the informativeness of unlabeled instances. First, the classifier is trained, using the set of labeled instances. Then, the pool of unlabeled instances is queried using a query strategy. Depending on the number of queries, the most informative unlabeled instances (i. e., those with a score above the minimum threshold) are randomly selected and added into the training set. Finally, these instances are removed from the pool of unlabeled instances. This process is then repeated until the stopping criteria are met.

Given: Labeled set L, unlabeled pool U, query strategy $\emptyset(\cdot)$, query batch size B, threshold τ

repeat

// learn a model using the current L

θ = train (L);

while b = 1 **to** B **do**

// query the most informative instance

$X_b^* = random_{X \epsilon U}(\emptyset(X) > \tau)$;

// move the labeled query from U to L

$L = L \cup (X_b^*, label(X_b^*))$;

$U = U - X_b^*$;

end

Until some stopping criterion;

Figure 3.8: Stream-based active learning

Source: An Analysis of Active Learning Strategies for Sequence Labeling Tasks (Settles & Craven, 2008).

3.2.1.2 Pool-Based Selective Sampling

Unlike the stream-based selective sampling, a learning algorithm in the pool-based selective sampling (Lewis & Gale, 1994) analyses a set of unlabeled instances and then selects the most informative ones. In other words, the algorithm first queries instances from the pool of unlabeled instances using informativeness measures, ranks them, and select the best instances regardless of their individual order (Settles & Craven, 2008). The process of active learning in the pool-based selective sampling is showed in Figure 3.7. Given a small set of labeled instances L and a large pool of unlabeled instances U, the process begins with training the classifier on the current labeled set. Depending on the number of queries, the pool of unlabeled instances is then queried (by using query strategies) for selecting the most informative one (the one with a higher score), which is then added to the current labeled set. Finally, the selected instance is removed from the pool of unlabeled instances. This process is repeated until some stopping criteria are met or a certain level of accuracy is achieved.

Given: Labeled set L, unlabeled pool U, query strategy $\emptyset(\cdot)$, query batch size B

repeat

// learn a model using the current L

θ = train(L);

for b = 1 **to** B **do**

// query the most informative instance

$X_b^* = argmax_{X \epsilon U} \emptyset(\cdot)$;

// move the labeled query from U to L

$L = L \cup (X_b^*, label(X_b^*))$;

$U = U - X_b^*$;

End

Until some stopping criterion;

Figure 3.9: Pool-based active learning

Source: An Analysis of Active Learning Strategies for Sequence Labeling Tasks (Settles & Craven, 2008).

In these scenarios, several query strategies can be applied as described by (Settles, 2010).They will be explored in the next section .

3.2.2 Query strategies

There are many query strategies in active learning. In this study, we will examine three categories, including uncertainty sampling, query by committee and density weighted.

3.2.2.1 Uncertainty Sampling

The uncertainty sampling strategy is the most used query strategy in active learning (Lewis & Gale, 1994). It is used to reduce the number of samples to be labeled by the human or computer, thus resulting in a classifier with lower error rates (Settles, 2010). The key idea behind this strategy is the iteratively selection and query of class labels of the most uncertain unlabeled instances (Lewis & Catlett, 1994). This is based on how to measure the uncertainty of each unlabeled instance. In general, the process of the Uncertainty sampling can be summarized as follows (Sun & Wang, 2010):

1. The first step consists of using the labeled instances to train the classifier
2. Second: predicting the label with each unlabeled instance

3. Third: selecting the most uncertain instance that will be labeled by the expert
4. Fourth: putting the most uncertain instance into the training set to train the classifier again

Three metrics are used in the uncertainty sampling strategy to evaluate the uncertainty of unlabeled instances.

- **Least Confident**

It is used to measure the uncertainty of the instances. Moreover, it selects the instance with the most likely class label (Settles, 2010). In other words, the least confident metric explores the instances with the highest posterior probability. The most informative unlabeled instance using the least confident metric can be determined as follows (Culotta & McCallum, 2005):

$$x^*_{LC} = argmax_x - P_\theta(\hat{y}|x)$$

with

$$\hat{y} = argmax_y * P_\theta(y|x)$$

($P_\theta(\hat{y}|x)$ is the posterior probability under the model θ of the most likely class $\hat{y}$ being the true class of the instance x. x^*_{LC} is the most informative instance). This metric computes the expected 0/1 loss, which corresponds to the error rate that the instance x is mislabeled. However, this metric does not use information on the remaining class label, which may be considered a disadvantage (Settles, 2010).

- **Margin sampling**

The margin sampling metric represents the difference between the highest and second highest probability for an instance (Settle, 2010). In other words, the margin sampling measure is used to compute the gap between the first and second most likely class . Moreover, it can also be used for samples with a certain level of doubt regarding class prediction. The margin sampling allows to quantify how easy it is (high margin) or difficult (low margin) to verify the label of the sample. The higher the margin, the more likely the class prediction of the sample is (Settle, 2010). This means that instances with the highest margin scores are more uncertain and are therefore selected for the labeling process. The higher the margin score of an instance, the more informative it is. Instances with small margin scores are less informative and are therefore not labeled (Settle, 2010). Thus, the learning algorithm is employed to query an unlabeled sample using the margin sampling, and instances with high margin score are then labeled. The most informative instance in an unlabeled set using the margin sampling is determined as follows (Scheffer, Decomain & Wrobel, 2001):

$$x^*_M = argmin_x P_\theta(\hat{y}_1|x) - P_\theta(\hat{y}_2|x)$$

The most informative instance x^*_M using margin sampling is the difference between the probabilities of posterior probability under the model θ of the first and second most likely class $\hat{y}_1$, $\hat{y}_2$, respectively being the true class of the instance x. However, the margin sampling measure does not work very well for large label sets (Settles, 2010).

- **Entropy**

It is the most popular uncertainty sampling metric derived from information theory (Tang, Luo & Roukos, 2002). It can be interpreted as an information-retrieval measure that corresponds to the uncertainty over the entire output prediction distribution (Fu, Zhu & Li, 2012). Thus, all unlabeled instances are queried using the entropy metric and the instance with the highest entropy score is selected for labeling. This can be determined as follows (Shannon, 1948):

$$x^*_H = argmax_x - \sum_i P_\theta(y_i|x) log P_\theta(y_i|x)$$

($P_\theta(y_i|x)$ is the posterior probability under the model θ of a range of over all possible labeling y_i being the true class of the instance x, and x^*_E is the most informative instance using the entropy metric). This metric is similar to margin when the classification is binary (Tong & Koller, 2002).

The following tables summarize the formulation, objective and considered class label of each uncertainty metric:

Metric	Formulation	$\emptyset P_\theta(\hat{y}\|x)$
Least Confident	$x^*_{LC} = argmax_x 1 - P_\theta(\hat{y}\|x)$	$1 - P_\theta(\hat{y}\|x)$
Margin	$x^*_M = argmax_x [P_\theta(\hat{y}_2\|x) - P_\theta(\hat{y}_1\|x)]$	$P_\theta(\hat{y}_2\|x) - P_\theta(\hat{y}_1\|x)$
Entropy	$x^*_H = argmax_x \sum_y -P_\theta(y\|x) log P_\theta(y\|x)$	$\sum_y -P_\theta(y\|x) log P_\theta(y\|x)$

Table 3.1: Uncertainty Metrics

Source: An analysis of active learning strategies for sequence labeling task (Settles & Craven, 1998).

Uncertainty measures	Considered class label	Objective
Least confidence	The one with the highest posterior probability	Decrease the error rate

Sample margin	The first two most probable class labels	Decrease the error rate
Entropy	Over the whole output prediction distributions	Reduce the log-loss

Table 3.2: Major uncertainty measures for uncertainty sampling
Source: A survey on instance selection for active learning (Fu, Zhu & Li, 2012).

Table 3.2 shows that the Entropy metric is used when the aim is to minimize the log-loss function, while Margin and Least Confident metric are more appropriate to reduce the rate misclassification.

In this study, this approach will be investigated with classifiers presented in the section 3.1.

3.2.2.2 Query By Committee

The Query By Committee (QBC) strategy is based on a vote on the class label for unlabeled instances by each member of committee (Settles, 2010). This committee member is built from a set of classifiers trained on the same labeled sample and is used to predict the label of the instance. Thus, the final prediction class of the instance is that with most of the votes of the committee members (Seung, Opper, & Sompolinsky, 1992). In the QBC approach, the pool of unlabeled instances is explored and only the most informative ones are selected for labeling. The most informative instances are those with the strongest disagreement among committee members (Settles, 2010). Moreover, the QBC approach is mainly aimed at minimizing version space, or hypothesis space as much as possible with as few labeled instances as possible (Seung, Opper & Sompolinsky, 1992). In other words, it is mostly used to find an optimal hypothesis within a version space, also called query space, which represents the set of models or hypotheses that are built from the same labeled training data. The more models there are, the smaller the size of the version space and the fewer instances to label (Settle, 2010). The implementation of the QBC algorithm is based on two essential tasks (Argamon-Engleson & Dagan, 1999):

1. The construction of a committee of hypothesis or models corresponding to different regions of the version space
2. And the design of some measure to evaluate the disagreement between committee members.

This process allows the learner to decide whether the label of an unlabeled instance is queried or not. Thus, a committee of different hypotheses or classifiers is first created by the learner from the current labeled training set, and then unlabeled instances are queried by each committee member using disagreement measures, and finally the instances, for which the committee members disagree on the most, are selected for labeling (Argamon-Engleson & Dagan, 1999). There are two main metrics for measuring disagreement among committee members:

❖ Vote Entropy

The vote entropy metric measures disagreement by the entropy of the classification distribution "voted for" by committee members (Zhao & al., 2008). It represents the value of the uniformity of classes that each committee member has assigned to an instance. The most informative instance using this metric can be found as follows (Dagan & Engelson, 1995):

$$x^*_{VE} = argmax_x - \sum_i \frac{V(y_i)}{C} log \frac{V(y_i)}{C}$$

(x^*_{VE} is the most informative instance using the Vote Entropy. y_i is the range over all possible labelings, and $V(y_i)$ is the number of votes that committee members' prediction assigns to a label, and C corresponds to the committee size). The higher the vote entropy, the greater the disagreement among committee members, and vice versa. When the vote entropy is zero, all committee members agree (Settle, 2010).

❖ Kullback-Leibler divergence

The Kullback-Leibler (KL) divergence (Kullback & Leibler, 1951) is used to measure the difference between two probability distributions. It thus makes it possible to measure the strength of the uncertainty disagreement between the different classes that have been assigned by the members of committee to the instance by computing the difference between the votes of the members of committee. The most informative instance using the KL divergence can be determined as follows (McCallum & Nigam, 1998):

$$x^*_{KL} = argmax_x \frac{1}{C} \sum_{c=1}^{C} D\left(P_{\theta^{(c)}} \| P_C\right),$$

Where:

$$D\left(P_{\theta^{(c)}} \| P_C\right) = \sum_i P_{\theta^{(c)}}(y_i|x) \, log \frac{P_{\theta^{(c)}}(y_i|x)}{P_C(y_i|x)}$$

($\theta^{(c)}$corresponds to a particular model in the committee, C is the entire committee, $P_C(y_i|x) = \frac{1}{C}\sum_{c=1}^{C} P_{\theta^{(c)}}(y_i|x)$ represents the consensus probability of the range over all possible labeling y_i being the correct label of the instance x).

Table 3.3 summarizes the query strategies and their objectives:

QBC measures	Considered class label	Objective
KL divergence	The one with the highest degree of disagreement	Reduce the version space
Vote Entropy	Vote assigned by the committee member over	Reduce the version space

	the whole output distributions	

Table 3.3: Disagreement measures in the QBC.
Source: A survey on instance selection for active learning (Fu, Zhu & Li, 2012).

Committee building in the QBC approach requires classifiers and the QBC measures, which measure disagreement as variance between the output predictions of committee member (Burbidge, Rowland & King, 2007). As with the uncertainty sampling, the QBC approach will be implemented with credit scoring models (presented in Section 3.1).

3.2.2.3 Density-Weighted method

The density-weighted method is derived from the combination of a similarity and uncertainty metric and is based on the idea that the model should focus on instances that are close to each other in order to avoid non-informative outliers (Settles, 2010). With this strategy, the most informative instances are those which are both uncertain and representative of the sample distribution (Settles, 2008). Therefore, this approach suggests the use of the average similarity of an instance to all the other instances from the unlabeled set to measure its informativeness (Settles, 2010). Determining the most informative instance using this method is done through the following equation (Settles, 2008):

$$x^*_{ID} = argmax_x \phi_A(x) * (\frac{1}{U}\sum_{u=1}^{U} sim(x, x^{(u)}))^{\beta}$$

(where $\phi_A(x)$ is the informativeness of the instance x according to some base query strategy A, such as an uncertainty sampling or QBC approach, sim is a similarity metric, $(\frac{1}{U}\sum_{u=1}^{U} sim(x, x^{(u)}))^{\beta}$ represents the weight of the informativeness of x by its average similarity to all other instances in the unlabeled set, subject to a parameter β that controls the relative importance of the density term).

This approach will be implemented using the credit scoring models presented in the section 3.1 and the uncertainty sampling strategy.

3.3 Performance Evaluation Measures

The evaluation process is one of the most important steps in the development of a model (Baesens et al., 2003). It is used to see how well a model fits a different data set and captures the real relationships between variables. Moreover, the evaluation metrics are also used to compare the performance of models in order to select the model with the highest predictive power. There are several performance measures for the evaluation of a model in credit scoring application. Most studies generally implement a single performance measure, which is divided into three categories, namely discriminatory ability, accuracy of

probability predictions and correctness of prediction (Lessmann et al., 2015). The table below displays the different evaluation criteria for each performance measure:

Performance Measure	Evaluation Metrics
Correctness of Predictions	❖ Kolmogorov-Smirnov Statistics (KS) ❖ Percent Correctly Classified (PCC)
Discriminatory ability	❖ Area Under the Curve (AUC) ❖ Partial Gini Index (PG) ❖ H-Measure
Accuracy of probability predictions	❖ Brier-Score (BS)

Table 3.4: Performance Measures
Source: Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research (Lessman et al., 2015).

- **Correctness of Predictions**

The PCC and KS statistics are used to measure the correctness of categorical predictions. The following table provides a brief description of these performance measures:

Correctness of predictions metrics	Description
Percent Correctly Classified (PCC)	Still known as classification accuracy, the PCC represents the correct classification rate of observations. It requires separate class predictions, which is determined by comparing $P(+\|x)$ to threshold 'τ' and assigning 'x' to the positive class if $P(+\|x) > \tau$, and assigning 'x' to the negative class if $P(+\|x) < \tau$. However, 'τ' is function of the costs associated with granting credit to bad (Hand, 2005).
Kolmogorov-Smirnov Statistics (KS)	KS relies on $P(+\|x)$. It takes into consideration a fixed reference point. KS is used to measure the maximum difference between two cumulative distributions (positive and negative cases) (Thomas et al., 2002).

Table 3.5: Correctness of prediction performance.

❖ **Accuracy of probability predictions**

In this category, one distinguishes only the Bier-Score (BS). This metric assess the accuracy of probability predictions of the whole distribution. According to Hernandez, Flach and Ferri (2011), BS corresponds to the mean-squared error between $P(+|x)$ and zero-one response variable.

❖ **Discriminatory ability**

Various performance metrics including the AUC, H-measure, and PG, are used to measure the discriminatory ability. The following table illustrates a brief description of these performance measurements:

Discriminatory ability metrics	**Description**
Area under the curve (AUC)	The AUC is the probability that a positive instance, randomly selected, will score higher than a randomly selected negative instance. (Lessmann et al., 2015).
H-Measure	The H-measure evaluates the performance of the classifier from the expected minimum misclassification. It is between zero (random classifier) and one (perfect classifier). (Lessmann et al., 2015).
Partial Gini Index (PG)	The PG is based only on one part of the probability distribution $P(+\|x) \leq b$ (Pundir & Seshadri, 2012).

Table 3.6: Discriminatory Ability Performance Measure

The AUC metric is commonly used to evaluate the performance of models. It is based on the ROC (Receiver Operator Characteristics) curve. Also called “Lorentz diagram”, it measures the performance of a model to classify instances as positive or negative (Abdou & Pointon, 2011). This performance can also be evaluated from the confusion matrix (Zhang & Bhattacharyya, 2004), which is presented as follows:

	Predicted Condition = Positive	**Predicted Condition = Negative**
True Condition = Positive	True Positive – **TP** (correctly predicted a true condition)	False Negative – **FN** (wrongly predicted a negative condition)

True Condition = Negative	False Positive – **FP** (wrongly predicted a true condition)	True Negative – **TN** (correctly predicted a negative condition)

Table 3.7: Confusion Matrix.
Source: An Introduction to ROC Analysis (Fawcett, 2006)

Also called a two-by-two matrix, the confusion matrix shows the combinations of the number of actual and predicted observations of a data set (Abdou, 2009). The rows represent actual positive and negative classes of observations, and the columns represent predicted positive and negative classes of observations. An observation is called a true positive if it is classified in the positive class, and indeed belongs to the positive class. It is called false negative if it is classified in the negative class, while it is positive. On the other hand, it is called false positive if it is classified in the positive class while it belongs to the negative class. It is called true negative if it is classified in the negative class and indeed belongs to the negative class. From the confusion matrix, performance measures such as the accuracy, the precision, the F1 score and recall can be determined:

- Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$ It represents the predictive performance of a model. It is the probability that the set of predicted labels corresponds exactly to the set of actual labels.
- Precision = $\frac{TP}{TP+FP}$. It measures the percentage of positive cases from total predicted cases
- Recall = $\frac{TP}{TP+FN}$. It measures the percentage of how many total positives cases were predicted correctly with the model.
- F1 score = $\frac{2*precision*recall}{precision+recall}$. It is the balance between precision and recall

The ROC curve is a two-dimensional graph in which the True Positive Rate (TPR) (also known as sensitivity) is represented on the vertical axe and the False Positive Rate (FPR) (1 – specificity) is represented on the horizontal axe (Abdou & Pointon, 2011). The FPR is the rate of false positive instances from all positive instances and the TPR is instead the rate of true positive instances from all positive instances. The coordinates of each axe are between 0 and 1. The best possible prediction corresponds to 1 for the sensitivity on the vertical axe and to 1 for 1-specificity on the horizontal axe, which indicates that there are no FN. The diagonal line (from the left bottom to the top right corner) separates the ROC space into two equal parts. All points above the diagonal represent good classification results and all points below the diagonal line represent poor classification results. Points along the diagonal line are random. The following figure provides an overview of the ROC curve:

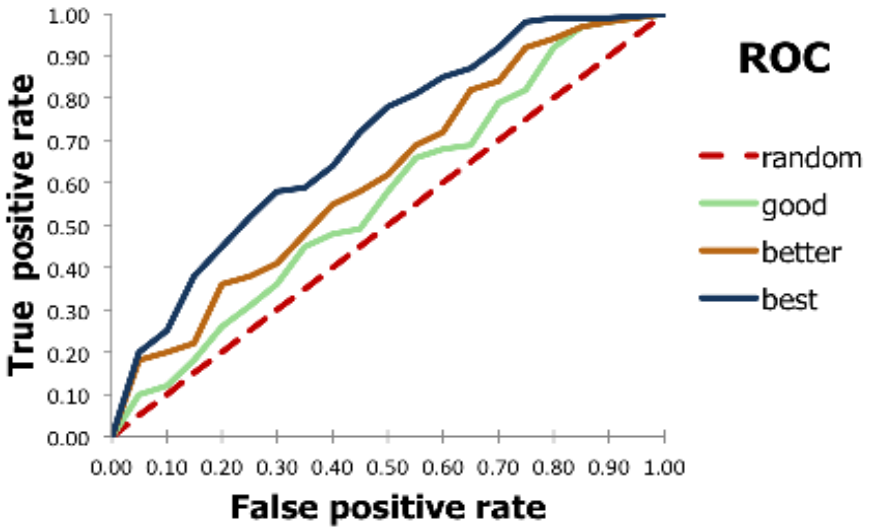

Figure 3.10: ROC Curves:

Source: OpenEye Scientific

The dataset used in this study is imbalanced in the class distributions (see section 4.2), which may affect performance measures such as BS and KS (Gong & Huang, 2012). However, the class imbalance does not affect the AUC, PG, and H-measure (Fawcett, 2006). For this reason, we use the performance measure of AUC to evaluate the performance of the credit scoring models.

4 Data Description

In this section, we present the dataset used in this study, which is publicly provided by IFR [1]. In the IFR, there are two links available for the datasets. For this study, the Mortgage dataset from creditriskanalytics.net[2] will be used. It is a publicity real-world dataset, which is in panel form and includes information about 50000 US mortgage borrowers over 60 periods. Each period corresponds to one month and the periods have been deidentified. The dataset is a random selection of mortgage loan level data gathered from the portfolios underlying US. Residential mortgage-backed securities (RMBS) and represents a random selection of mortgage loan-level data. It contains 622489 observations (recorded at different observation periods on 50000 borrowers), and 23 variables, i.e., 22 input variables, which provide information about the mortgage for each borrower, and the binary output variable “default”, which takes the value 1 for the defaulting borrower and 0 for the non-defaulting borrower. In this study, only the last observation period for each borrower will be considered.

1 www.internationalfinancialresearch.org

2 www.creditriskanalytics.net

4.1 Data Dictionary

This section provide information regarding each variable of the dataset, that are available publicly on the website of IFR. Table A-1 in the appendix summarized this information. Furthermore, not all variables will be used to build the credit scoring models. During the cleaning process of the data, some of these variables (non-significant variables) will be removed by using feature selection techniques which will be presented in section 4.3. The most important variables are highlighted in section 4.3.

4.2 Exploratory Analysis

In this part, we show how the observations in the target variable (default_time variable) are distributed. This repartition is shown in the graph and the table below:

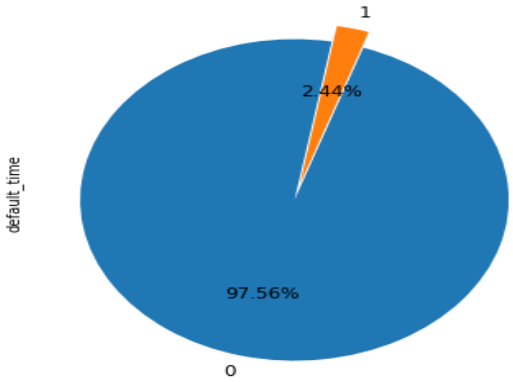

Figure 4.1: Distribution of classes in percentage

class	Number of borrowers	Percentage
0	607331	97.56%
1	15158	2.44%

Table 4.1: Class distribution

From the graph and table above, we see that the dataset is highly unbalanced in the distribution of the two classes default and non-default. Nearly 98% of the observations are assigned to the class 0 (non-default), while 2% of the observations are assigned to the class 1 (default). This imbalance in the class distribution may affect classifiers (Cieslak & Chawla, 2008). In other words, it becomes difficult for classifiers to recognize minor classes because they are influenced by major classes. This is the case with probabilistic models such as the LR model, where the probabilities of minor classes are underestimated (Cieslak & Chawla, 2008). Furthermore, with the imbalance in the class distribution, the recall performance of the classifiers is high while their sensitivity is very low (King & Zeng, 2001). To reduce the imbalance in the data, only the last observation period of each borrower will be considered. The new data distribution now consists of 50000 observations and is distributed as follows:

class	Number of borrowers	Percentage
0	34846	69.69%
1	15154	30.31%

Table 4.2: New class distribution

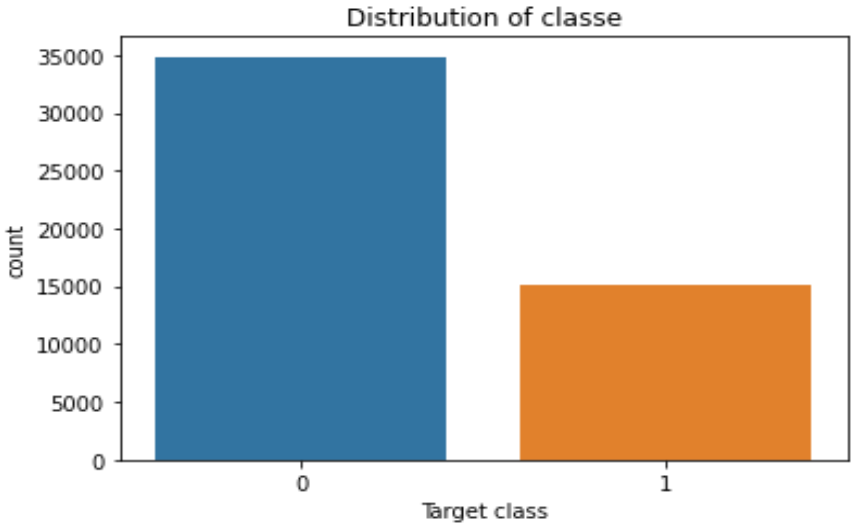

Figure 4.2: New class distribution

The figure above shows a decrease in the imbalance in the class distribution of the target variable “default_time”. The non-default class (class 0) now represents roughly 70% of the dataset, while the default class (class 1) represents 30% of the dataset. Comparing to the preceding class distribution, the gap between the two classes in the new distribution is not too big.

The following figures illustrate the target variable as a function of the time variables, namely time and maturity_time. They indicate the number of applications that are default and non-default at the observation time.

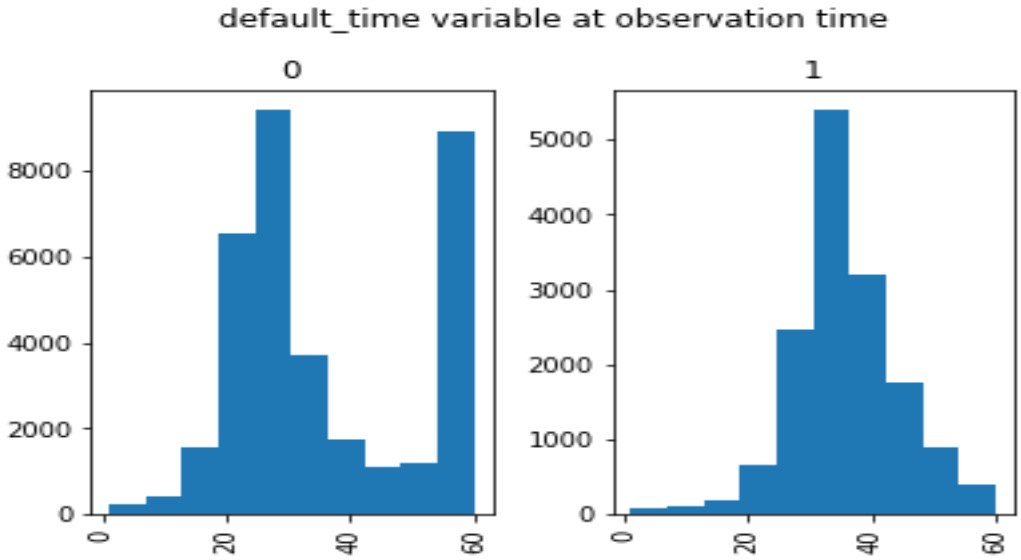

Figure 4.3: Default time variable in relation to observation time

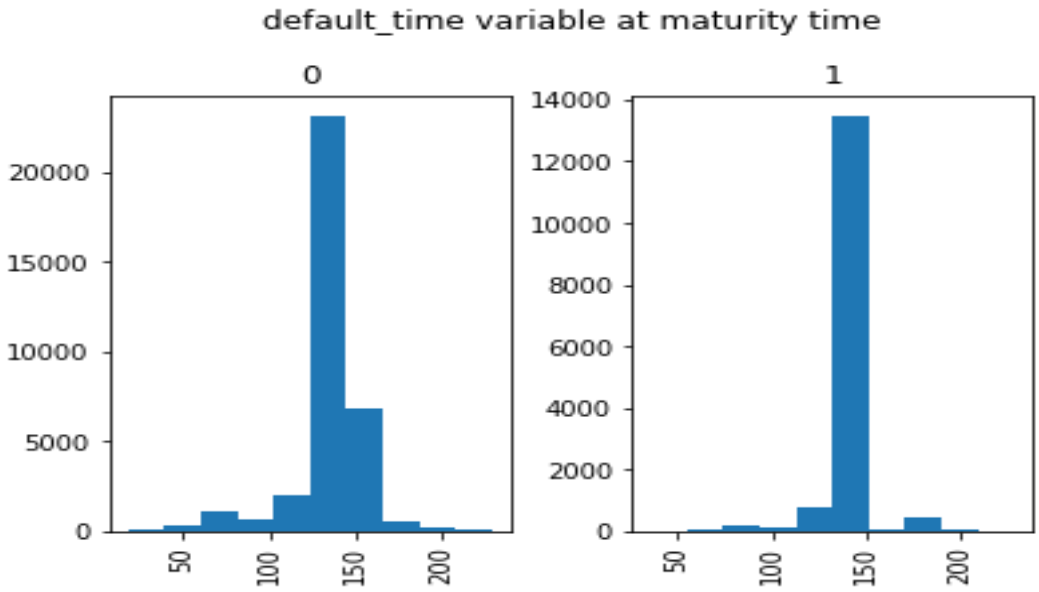

Figure 4.4: Default time variable in relation to maturity time

The two figures above are representations of the target variable (default_time variable) with the observation time and the maturity time. The number of defaulting (class 1) and non-defaulting borrowers (class 0) in the repayment of the mortgage loan is represented on the vertical axis and the time on the horizontal axis. In Figure 4.3, the number of defaulting borrowers increases approximately until the 30th observation period, when it reaches its maximum, and then decreases progressively until the last observation period. On the other hand, the number of non-defaulting borrowers increases until the 30th period and decreases approximately until the 45th period and increases again until the last period. Furthermore, the highest number of non-defaulting borrowers is recorded in the 30th and last period. This figure shows that there are more non-defaulting borrowers than defaulting borrowers within the observed period. Figure 4.4 however shows that the highest number of non-defaulting and defaulting borrowers is recorded between the 120th and 150th maturity time. Furthermore, it also shows that the number of non-defaulting borrowers at the maturity time is higher than the number of defaulting borrowers.

4.3 Data Wrangling

This section presents the feature selection techniques that were used to highlight the most important variables for building the credit scoring models. Three reasons explain why the variable selection is one of the most important steps before building the model (Guyon & Elisseeff, 2003). The first reason is that it helps to improve the prediction performance of the predictors. Second, it provides faster and more cost-effective predictors and finally provides a better understanding of the process. Even though there are many variable selection techniques, we mainly focused on three techniques.

4.3.1 Missing observations

Variables with missing values may lead to a poor estimation of the model. The more values are missing within the dataset, the less performing the model is. Therefore, it is important to explore the dataset first in order to identify variables with missing values. Then, the variables with the greatest number of missing values are removed from the dataset. In our dataset, only the LTV_time variable has missing values (18 missing observations). All other variables do not have missing values. Thus, the missing values in this variable will be dropped. Finally, all the variables in the dataset will be kept. Table A-2 in the appendix summarizes the number of missing observations in each variable from the dataset.

4.3.2 Correlation

Correlation is an important process for removing variables from the dataset that are not significant (highly correlated with each other) for modelling the credit scoring models. A distinction is made between the Pearson correlation and the p-value.

4.3.2.1 Pearson correlation

The Pearson correlation coefficient[3] is one of the statistical measures used to analyze the relationship between variables. It is a feature selection process that evaluates the linearity between two random variables. In other words, it describes how much two random variables are linearly correlated and how strong their statistical relationship is. The Pearson correlation coefficient is generally between -1 and 1. The first one indicates the negative linear correlation while the second one indicates the positive linear correlation between two variables. If the Pearson correlation is zero, there is no linear relationship between the variables. The higher the Pearson correlation, the more the variables correlate. Highly correlated variables affect the performance of the model. Before building credit scoring models, the most correlated variables are first identified and then removed from the dataset. The Pearson correlation figure of the mortgage dataset in the appendix (Figure A-1) shows that the correlation between the variables is very low except between variables status_time and payoff_time, where it equals 0.93. It indicates a strong relationship between these variables. The two variables are not linearly independent and will therefore be removed from the dataset.

4.3.2.2 P-value

The P-value is generally used to identify variables with predictive power. Low p-values (usually ≤ 0.05) indicate that the variables are significant. Thus, if the p-value is ≤ 0.05, the variable can be kept, otherwise the variable must be dropped.

3 https://www.spss-tutorials.com/pearson-correlation-coefficient/.

From the P-value table in the appendix (Table A-3), we clearly see that the variables such as id, REtype_CO_orig_time, Retype_SF_orig_time and balance_orig_time are not significant in the prediction of the target variable (their p-values are greater than 0.05) and should therefore be removed from the dataset. All other variables have predictive power (significant) and will therefore be kept for data modeling.

4.3.3 Outlier detection

Another technique used to remove non-significant variables is the outlier detection method. Outliers may affect model estimation and must therefore be dropped before training the data. This technique is used to determine the rate of outliers in each variable from the dataset. Thus, variables with a large number of outliers are deleted from the dataset. The results of the outlier detection method in the variables from the mortgage dataset as displayed in the table A-4 in the appendix have shown that the most severe case is the uer_time variable with almost 16% of the selected data. The data seem to have been measured correctly and reflect reality. Therefore, the variables can be kept. To ensure that the model estimate is not affected by outliers, a standardization technique will be used.

4.4 Feature Importance

This allows to understand which variables are the most important in the build of credit scoring models. To highlight the most important variables, *sklearn python Package* is used. This automatically ranks all variables available by its importance by building a model. One of the most widely used models is the DT model. By building this model, we see that the “LTV_time” variable is ranked the highest. This variable plays a key role in estimating of credit scoring models, followed by the variable “time”, gdp_time”, “FICO_orig_time”. The variable ranking is shown in Figure 4.5.

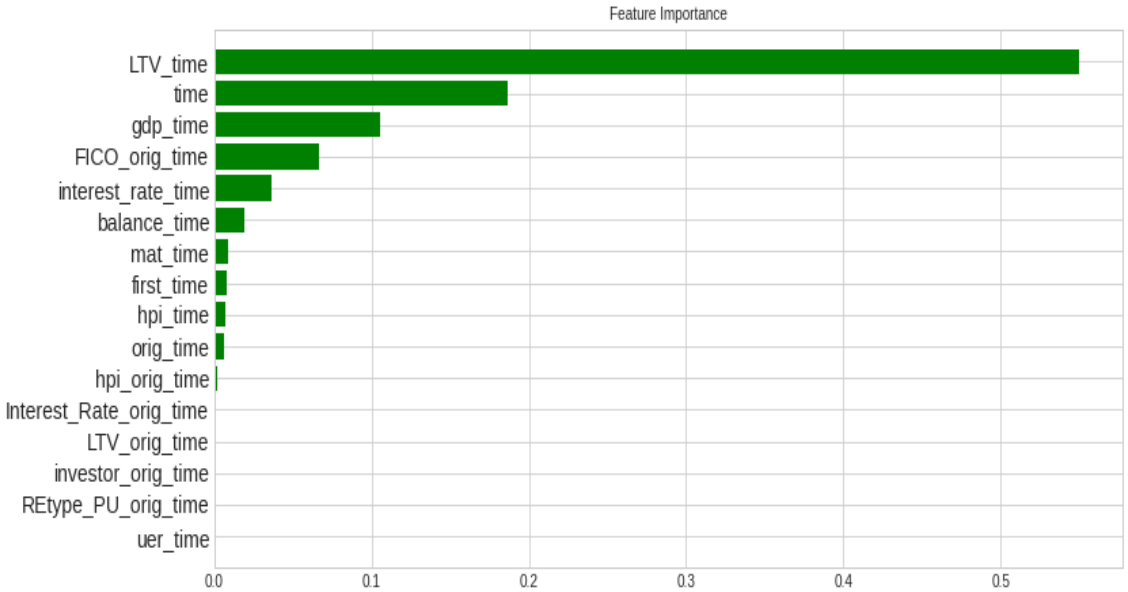

Figure 4.5: Feature Importance

4.5 Variables Statistical Description

Table A-5 in the appendix displays information about the variables from the dataset. It shows for each variable the minimal, the maximum, as well as the mean value, the standard deviation and also the variance. In addition, the 25%, 50% and 75% are also illustrated in this table. This table provides a clearer and more precise meaning of the data. Moreover, the most important variables for training credit scoring models (16 variables remaining after applying features selection techniques) and their correlation with each other and also with the target variable are presented in Figure A-2 in the appendix. This figure shows that the correlation between these variables is very low. Furthermore, the LTV_time variable correlates most with the target followed by gdp_time, uer_time, and hpi_orig_time variables and at the same time, the strength of their relationships with the target is very low.

4.6 Variables Visualization

In this section, we will visualize the most important variables in the building of credit scoring models and also their relationships with the target variable. Furthermore, the part of defaulting and non-defaulting borrowers in each variable representation will be illustrated:

- **Variable: LTV_time**

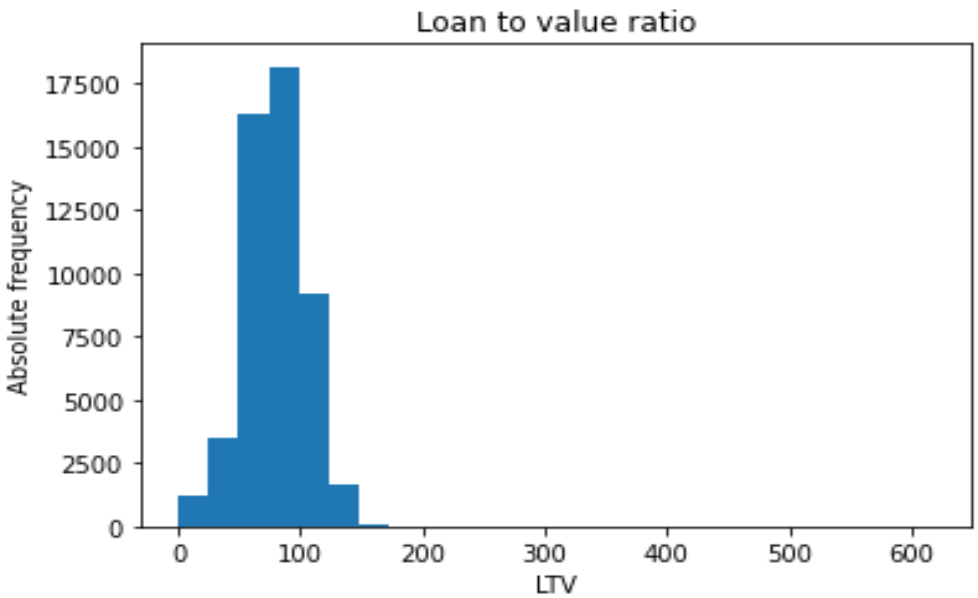

Figure 4.6: Loan to Value ratio at the observation time

The figure above shows the Loan To Value (LTV) score in percentage at the observation time for applications. It also shows that the LTV score of 100 is the most common for loan applications. At this the score, the number of mortgage applications is the highest. When the LTV score increases further, the number of applications decreases.

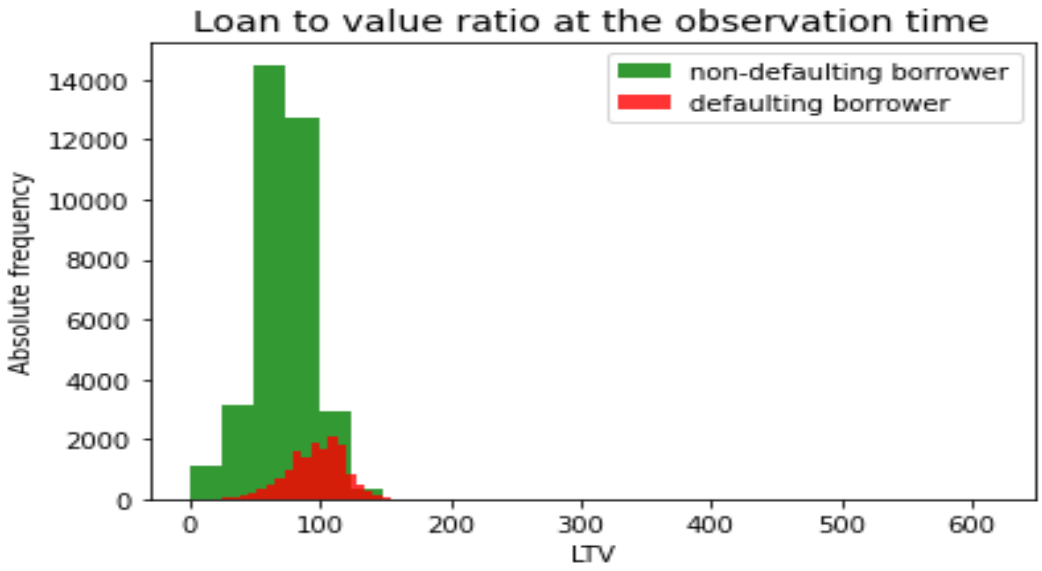

Figure 4.7: Loan to Value ratio distribution of Defaulters vs non-Defaulters

In the figure above of the LTV_time variable regarding to the target variable, we see that there are more non-defaulting borrowers than defaulting borrowers. The number of non-defaulting borrowers increases when the LTV score increases to about 80 and decreases to a near-minimum when it is above 80. At the same time, the number of defaulting borrowers increases until the LTV score reaches 100 and decreases when LTV is greater than 100, but it remains generally low compared to non-defaulting borrowers.

- **Variable: Time**

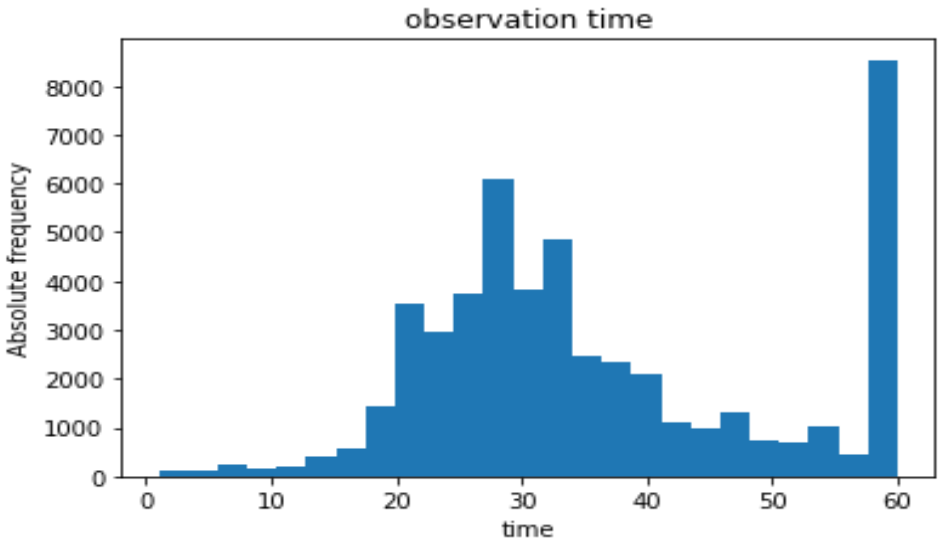

Figure 4.8: Time observing mortgage loans

The figure above shows that the majority of mortgage applications was observed between the 15th and the 30th period. Moreover, the highest number of the mortgage applications was recorded during the last observation period

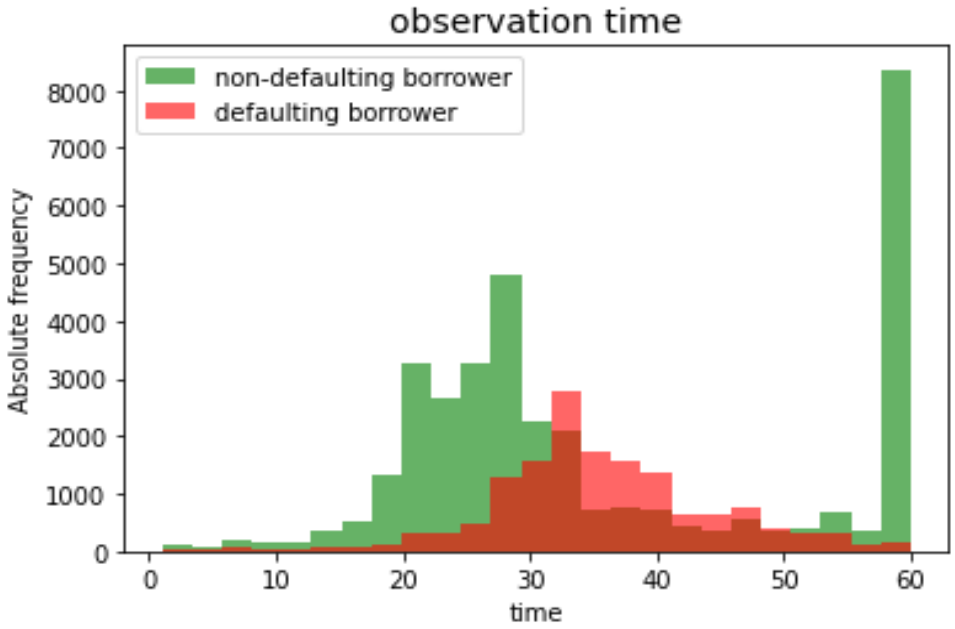

Figure 4.9: Time observing mortgage loans distribution of Defaulters vs non-Defaulters

Figure 4.9 shows that the number of non-defaulting borrowers is higher than the number or defaulting borrowers during the periods when there were the most mortgage applications (between the 15th and the 30th period and also in the last period).

- **Variable: gdp_time**

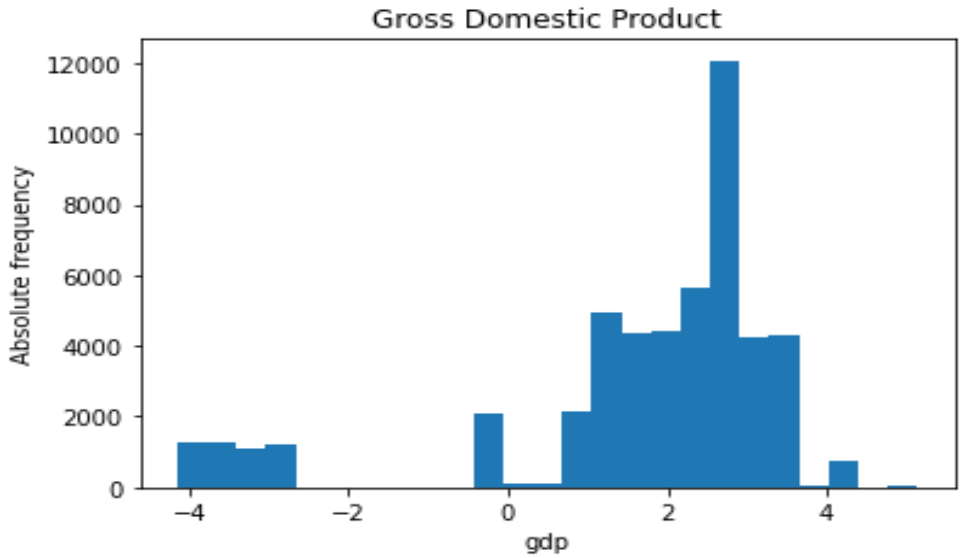

Figure 4.10: Gross Domestic Product at the observation time

The figure above shows that there are more mortgage applications when the Gross Domestic Product (GDP) score is positive. With a GDP of almost 3%, the number of mortgage applications is the highest

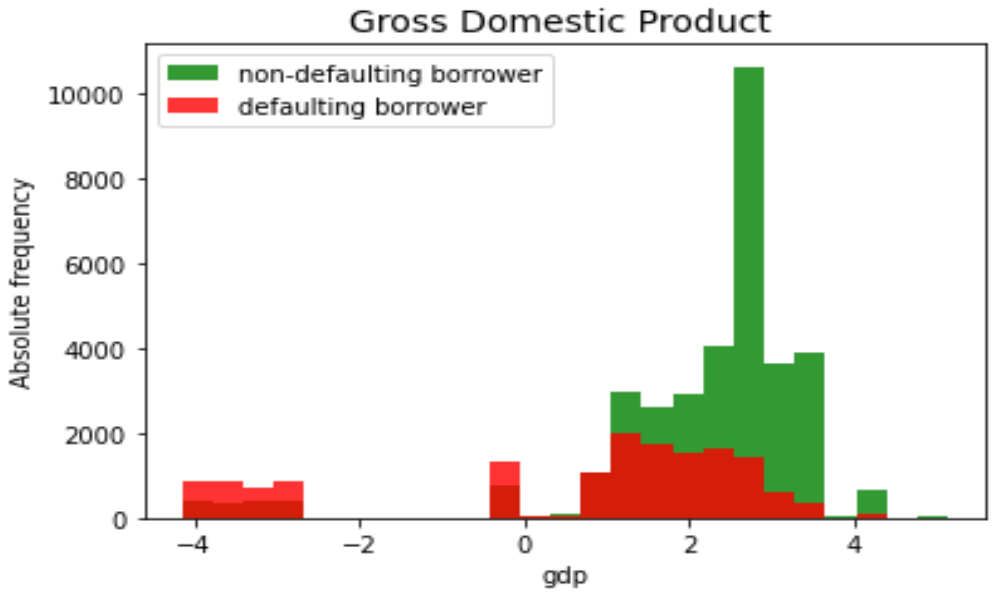

Figure 4.11: Gross Domestic Product of Defaulters vs non-Defaulters

Figure 4.11 shows that the number of non-defaulting borrowers is lower than the number of defaulting borrowers when the GDP is negative. However, the opposite is true when the GDP is positive.

- **Variable: FICO_orig_time**

From the figure below, we see an increasing number of mortgage applications when the FICO score increases to nearly 650 percent (correspondind to the highest number of mortgage applications) and a decreasing in mortgage applications when it increases beyond this point:

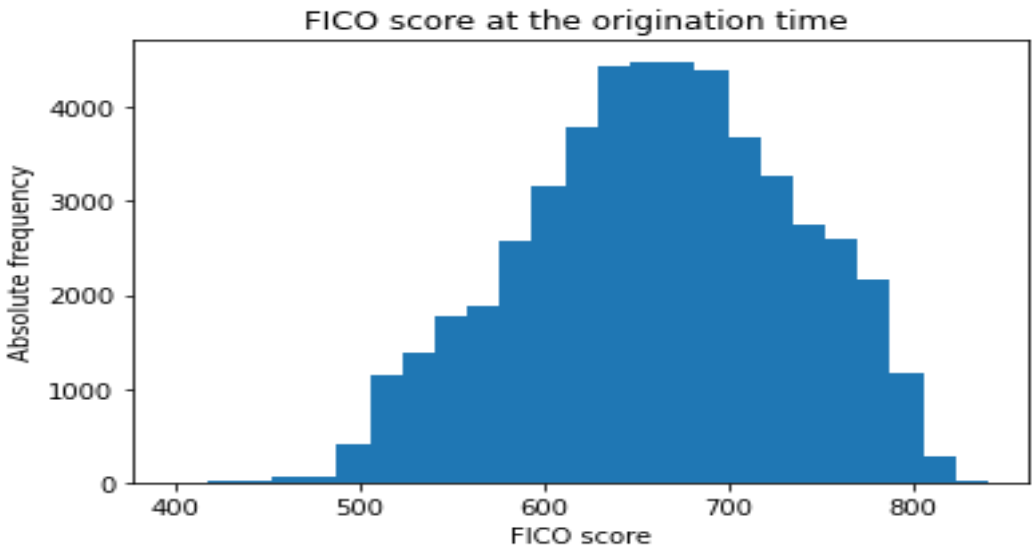

Figure 4.12: FICO score at the origination time

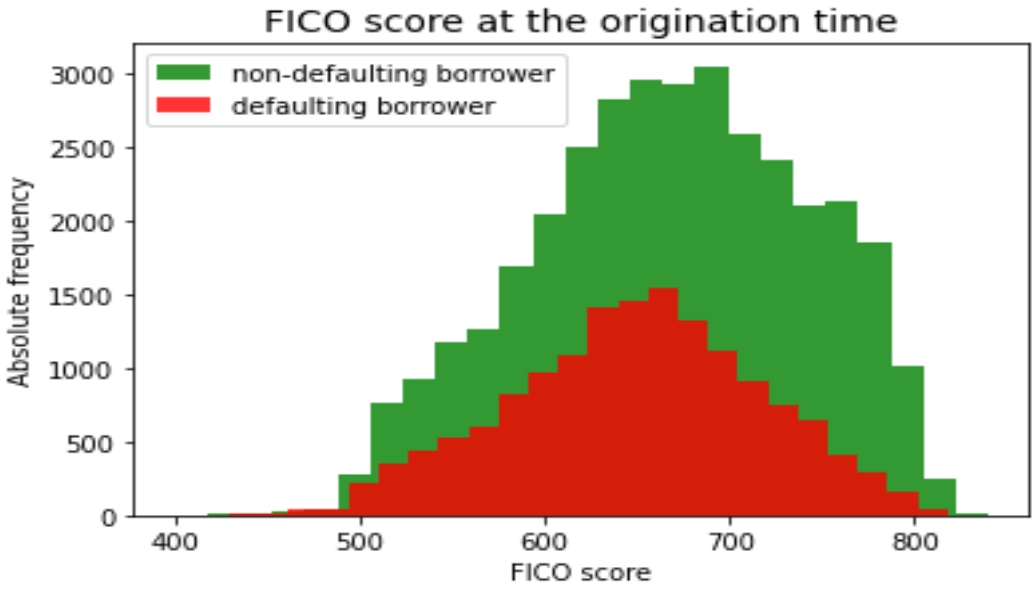

Figure 4.13: FICO score distribution of Defaulters vs non-Defaulters

Figure 4.12 shows the representation of the FICO_orig_time variable regarding the default time variable. It shows that there are more non-defaulting borrowers than defaulting borrowers at the time of origination. Furthermore, the number of defaulting and non-defaulting borrowers is the highest when the fico score is approximately equal to 680. Beyond this score, it decreases progressively to a minimum value close to zero.

- **Variable: interest_rate_time**

The figure below illustrates the observed interest rates applied to mortgage applications. It shows that the average interest rate is around 7% (the interest rate for which the number of mortgage applications is the highest). When the interest rate increases further, the number of mortgage applications decreases.

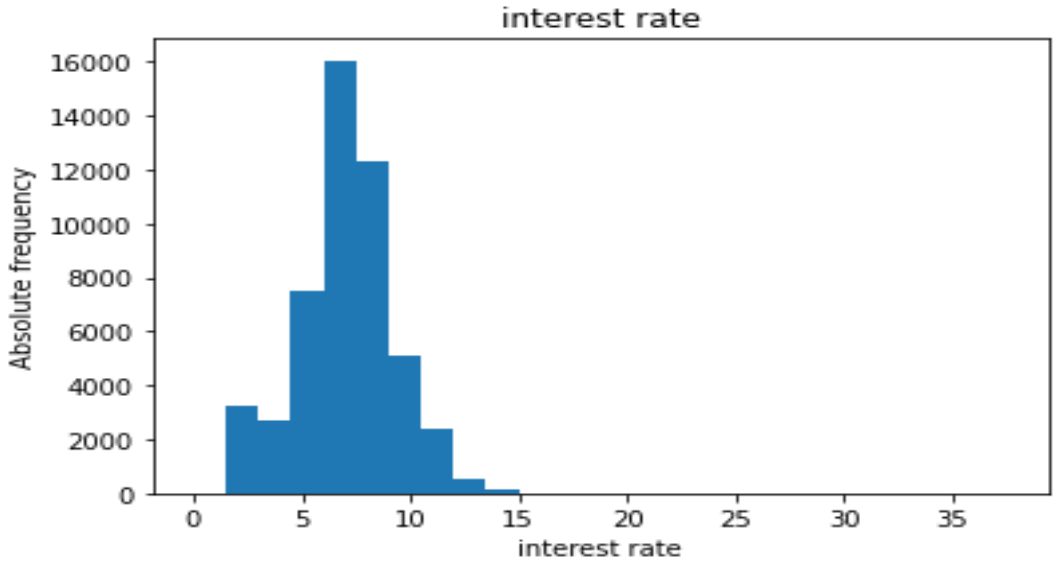

Figure 4.14: Interest rate at the observation time

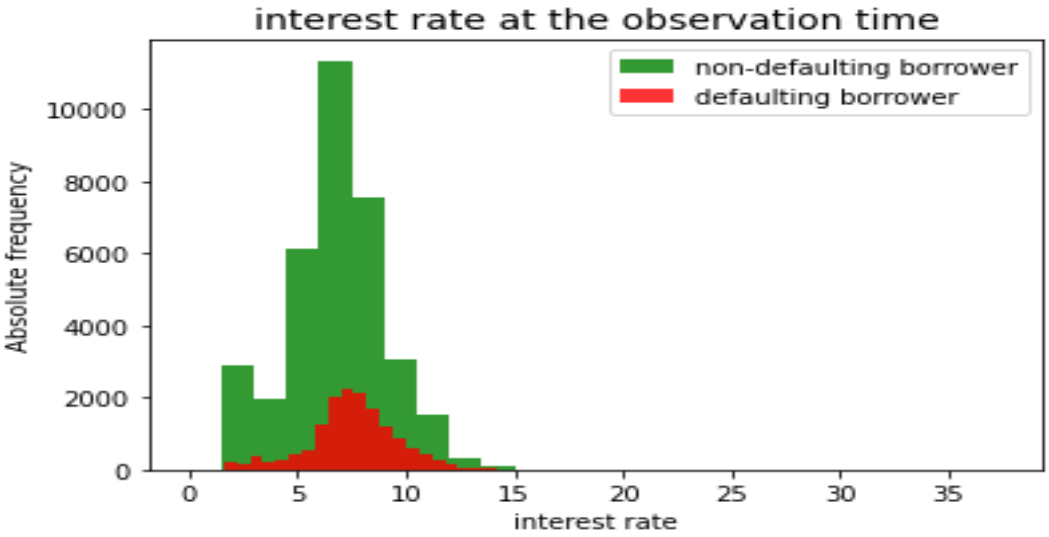

Figure 4.15: Interest rate distribution of Defaulters vs non-Defaulters

From Figure 4.15, we can clearly see that there are more non-defaulting borrowers than defaulting borrowers. The number of non-defaulting borrowers increases when the interest rate increases to reach 7% and then decreases progressively as the interest rate increases further. The same is true for defaulting borrowers.

5 Experiments and Results

In this section, we implement active learning strategies to select and add the most informative rejected applications into a training set to solve the problem of sampling bias and, thereby improving the predictive performance of credit scoring models. For this purpose, the final dataset obtained after the cleaning process is spit into a training and a test data. The training data represents 80% of the dataset and the test data 20% of the dataset. Then, the training and test data are standardized and scaled so that the estimation of the classifiers cannot be affected by the extension of the number of outliers. The scaling method is used to prevent variables with larger numerical values from dominating those with smaller numerical values when estimating the model (Hsu, Chang & Lin, 2003). The dataset contains only those applications for which the repayment behavior has been observed. To implement active learning strategies, we will assume that the training data is divided into two sets: the set of accepted applications and the set of rejected applications. The instances in the set of accepted applications will be randomly selected from the training data and the remaining instances will belong to the set of rejected applications. The repayment behavior of rejected applicants will be considered as non-observed. In summary, the database now includes a set of accepted applications on which credit scoring models will be trained, a set of rejected applications, which will be investigated throughout the active learning process, and the test data on which credit scoring models will be used to predict the creditworthiness of each applicant. Then, we use active learning strategies for selecting and adding the best rejected applications into the

training set. Finally, we compare the performance of credit scoring models based only on accepted applications with the performance of the same models when reducing sample bias. Additionally, we use credit scoring models to predict the creditworthiness of the applicants in the test set and evaluate their performance afterwards. These performances will then be compared with the performance of the credit scoring models on the test set when reducing sample bias.

1. Credit scoring models

The construction of credit scoring models is based on accepted applications only. Rejected applications are excluded. We randomly select about 10% of the observations from the training data, i.e., 5000 observations, which will represent the set of accepted applications, and on which the credit scoring models will be trained. The remaining instances in the training data will be considered as the pool of rejected applications. In this part, we build credit scoring models including the RF model, the SVM model, the KNN model, the NN model, the NB model, and the DT model.

- **LR**

The LR model is trained using the *"LogisticRegression"* function from the python package *"sklearn.linear_model"*. The construction of the model depends on a number of attributes, including the number of parameters (C), the number of iterations, the class weight, and the solver. The best number of parameters is found among a set of parameters using the "*GridSearchCV"* function, which includes the built LR model, the set of parameters and cross validation (cv=5). Furthermore, we assigned 500 to the number of iterations, considered the class weight to be balanced, and used a limited amount of computer memory (lbfg) for the estimation of the model. After training, we evaluated the performance of the LR model by calculating the accuracy and the AUC. Their values are summarized in Table 5.1. This estimated model is then used to predict the default class of applicants in the test set. Table 5.2 summarizes performance metric values of the LR model on the test set.

- **SVM**

To apply the SVM model to the data, we used the *"SVC"* function from the python package *"sklearn.svm"*. The first step is to determine the best parameter for which model accuracy is highest. This is done using the "*GridSearchCV"* function, which includes the built svc model, the parameter set, and the cross validation equals to 5. The SVM model is built from the "*SVC"* function including the "linear" as kernel, the "balanced" to the class weight attribute and "*True*" to the probability parameter. After training, the model evaluation is performed by calculating performance measures. The values of these performance metrics are presented in Table 5.1. Finally, the model is used to predict the default class of the borrowers in the test set. The output of the model for each predicted application is the

probability of default or non-default. Table 5.2 shows the values of performance metrics on the test set using the SVM model.

- **NB**

In the same way, the NB model is trained on only accepted applications. To do this, the *"GaussianNB"* function from the *"sklearn.naive_bayes"* package is used. The values of model performance measurements including accuracy and AUC are presented in Table 5.1. The estimated model is then used to predict the creditworthiness of applications in the test set. Performance metrics are used to evaluate the model on the test set. These results are shown in Table 5.2.

- **DT**

When applying the DT model on the training data, 8 trees were created with the *"tree.DecisionTreeClassifier"* function from the python package *"sklearn.tree"*. This model is useful to ease the understanding and interpretation. The values of performance metrics of this model during the training are summarized in Table 5.1. The estimated model is then used to predict the creditworthiness (default or not) of applicants in the test set. The performance metric values of this model during testing are presented in Table 5.2.

- **KNN**

The KNN model is built from the *"KNeighborsClassifier"* function that is available in the *"sklearn.neighbors"* package. This function comprises attributes such as the number of neighbors and weights. The KNN model is estimated on the number of neighbors equal to 22, representing the best parameter for which the classification accuracy is highest, and the designation *"uniform"* as weight. The values of performance measurements including the classification accuracy and AUC are shown in Table 5.1. After training, the estimated model is employed to predict the default class (default or not) for each applicant in the test set. Finally, this prediction is evaluated by computing performance metrics. These are shown in Table 5.2.

- **RF**

On the other hand, the RF model is constructed on the basis of a multitude of decision trees and outputting the class that represents the mode of the classes of the individual trees. Thus, to apply the RF model on the training data, the *"RandomForest"* function from the package *"sklearn.ensemble"* will be used. We first determined the best parameter among a set of parameters for which model accuracy is highest. This is done by using the *"GridSearchCV"* function including the RF model built, the set of parameters and the cross validation. Given the best parameter, the RF model resulted in the highest prediction rate with an accuracy of 1. In the same way, by applying the estimated model for predicting the default class of applications in the test set, it achieved the highest performance rate. The

values of the performance metrics on training set and test set are shown in Tables 5.1 and 5.2, respectively.

- **NN**

From the python package "*sklearn.neural_network* ", the *"MLPClassifier"* function is used to build the NN model. This function includes parameters such as hidden layer sizes, the learning rate, the solver, and the number of iterations. Before training the model on the data, we need to find the best size of the hidden layers and the best learning rate that provide the highest performance measurements of the model. The search is done on the set of hidden layer sizes and learning rate by using the *"GridSearchCV"* function, which comprises the NN model built and the number of cross validations equal to 5. After selecting the best combination of parameters, the model is then trained over 1000 iterations by using the *"adam"* function as solver. This model is then evaluated using performance metrics. The values of these metrics are summarized in Table 5.1. After that, the model is used to predict the default class of each application in the test set. Table 5.2 presents values of performance measurements of this model on the test set.

The following tables summarize the performance metric values for each model during the training and testing:

	Training	
Model	Accuracy	AUC
LR	0.7870	0.7732
SVM	0.7886	0.7724
NB	0.7966	0.7698
DT	0.8594	0.8318
KNN	0.8218	0.7807
RF	1	1
NN	0.8142	0.7629

Table 5.1: Performance of credit scoring models based on the set of accepted applications only

	Test	
Model	Accuracy	AUC
LR	0.7893	0.7737
SVM	0.7934	0.7751
NB	0.8	0.7746
DT	0.8033	0.7668
KNN	0.8178	0.7725
RF	0.8296	0.7870
NN	0.8197	0.7646

Table 5.2: Performance of credit scoring models on the test set

From Table 5.1, we see that all credit scoring models have a high predictive performance and AUC. The model with the highest performance is that of the RF model with accuracy and AUC equal to 1. This model is also the one with the highest performance metric values on the test set as shown in Table 5.2. Therefore, the RF model will not be investigated in the active learning process because it cannot be improved further.

The table below displays the confusion matrix for each classifier on the test set:

Models	Confusion matrix			
	TP	FP	TN	FN
LR	5693	1313	2198	793
KNN	6203	803	1973	1018
SVM	5750	1256	2182	809
DT	6010	996	2021	970
NB	5878	1128	2124	867
RF	6258	748	955	2036
NN	6319	687	1876	1115

Table 5.3: Confusion matrix of credit scoring models

Table 5.3 shows that the RF model is the one with the highest number of TP and TN and the lowest number of FP and FN.

2. Active learning method

The set of applications used for training credit scoring models is not representative of the general population of borrowers, resulting in biased estimation of credit scoring models The active learning method will be investigated to solve this problem. This approach explores the pool of rejected applications in order to highlight the applications that should be included in the process of credit scoring models based on expected performance improvements to reduce the sample bias in the training data. In this approach, query strategies will be implemented to select and added into the training set the most informative rejected applications that may be able to increase the predictive performance of credit scoring models that have been computed from the set of accepted applications only. Since the repayment behavior of rejected applications is not observed, it will first be determined bevor applying active learning. Generally, it will be inferred by using supplemental data (Banasik, Crook, & Thomas, 2003) or relying on the repayment behavior of accepted applications (Crook & Banasik, 2004). Furthermore, the target variable can also be imputed to the repayment behavior for rejected cases (Hand & Henley, 1994). In our case, the repayment behavior of rejected applications will be considered to be non-defaulting (Kiefer & Larsen, 2006). This means that, the non-default class (0) will be assigned to each rejected application in the active learning process. To apply active learning in python, the *'modAL'* package must first be installed. Then, the active learner model is built from the *"ActiveLearner"* function, which includes the estimator of

the credit scoring model, the query strategy, and the set of accepted application from which it learn. The query strategies are located in the python package *"modAL.uncertainty"*. In this section, two active learning scenarios have been implemented. In each scenario, various query strategies have been applied.

❖ **Scenario 1: Stream-based active learning**

In the Stream-based active learning scenario, the pool of rejected applications is first queried using query strategies and then the most informative ones are randomly selected based on the threshold and added one by one into the training set. Two query strategies have been implemented, namely uncertainty sampling and QBC.

➢ **Query strategy: Uncertainty sampling**

To select the most informative rejected applications, three types of classification uncertainty or selection techniques, also called uncertainty measures, are used in stream-based uncertainty sampling. These are classification uncertainty, classification margin and classification entropy. These three measures originate from the *"modAL.uncertainty"* package.

- **Classification entropy**

This metric is built from the *"classifier_entropy"* function, which comes from the python package *"modAL.uncetainty"*. Based on this uncertainty measurement, the most informative rejected applications (the most uncertain ones) are selected and then added one by one to the learning process. Moreover, it is combined with the estimation of the credit scoring model and as well as the set of accepted applications through the *"ActiveLearner"* function in order to build the learner model, which will be used to calculate the initial score of the credit scoring model from the set of accepted applications only. Then, this score will be updated as the most informative rejected applications are added during the learning process. The process of credit scoring models using classification entropy in the active learning process can be described as follows:

▪ **SVM**

In the SVM model, the learner model is constructed from the *"ActiveLearner"* function, which includes the previous estimation of the classifier as the estimator and the *"uncertainty_sampling"* as the query strategy. This learner model is then used to calculate the score of the model based on the set of accepted applications only. This score is then updated by adding the most informative rejected applications during the active learning process. Thus, the initial score of the SVM model is 0.7886. After that, the pool of rejected applications is queried using the *"classifier_entropy"* function, which includes the learner's SVM model built. By querying, the entropy score of the rejected applications is between 0.1 and 0.6. The higher the entropy score, the more informative the applications. Furthermore, each rejected application is explored and selected if its classifier entropy is greater than the threshold. On the basis of the minimum threshold, set at 0.6, we

randomly selected the 100 most informative applications from the pool of rejected applications and added them one by one into the training set. In other words, a rejected application is considered informative if its entropy score is greater than the threshold of 0.6. Then, the *".teach(X,y)"* function, where X represents the new observations and y the corresponding labels, is used to teach the selected instances. After adding each instance, the model score is improving as shown in Figure 5.1. The model score reached 0.7914 after adding the 100 best rejected applications. Finally, the added applications are removed from the pool of rejected applications using the *"np.delete"* function. The trained model is then evaluated using performance metrics, the values of which are summarized in Table 5.4. This model is then used to predict the creditworthiness of applicants in the test set. Performance metric values of the prediction are shown in Table 5.5.

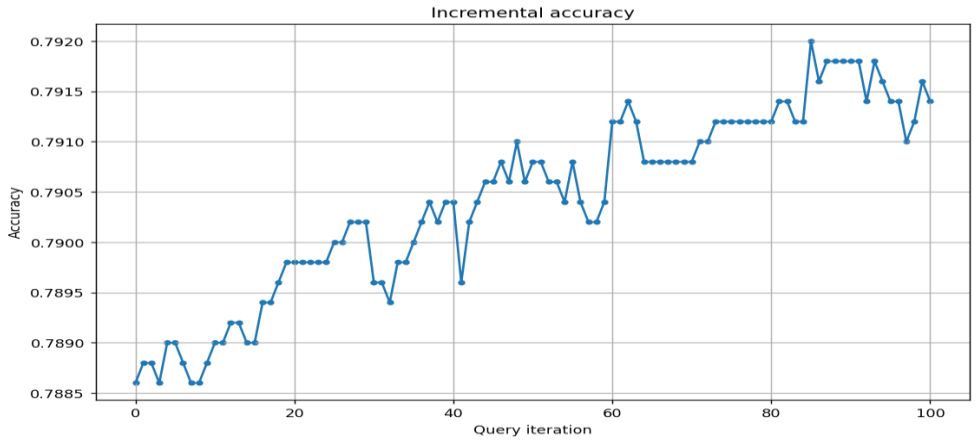

Figure 5.1: Accuracy curve of the SVM model when reducing sample bias

- **LR**

As with the SVM model, the score of the LR model is computed on the set of accepted applications from the learner model built using the *"ActiveLearner"* function and including the estimation of the LR model using the *"LogisticRegression"* function and the *"uncertainty_sampling"* as the query strategy. This score is then updated by adding the most informative rejected applications. After that, the pool of rejected applications is queried using the *"classifier_entropy"* function, which includes the learner's LR model previously built. The entropy score of the pool of rejected applications is between 0.1 and 0.6. Then, the 100 most informative applications with an entropy score greater than 0.5 (threshold) are randomly selected from the pool of rejected applications and added one by one to the process. Figure 5.2 shows how the score of the model improves with each addition of the most informative rejected application. Finally, the added applications are deleted from the pool of rejected applications in order to avoid it being added two or more times in the process. This model is then assessed using performance measurements. The values of the performance metrics of this model are shown in Table 5.4. The trained model is then used to predict the default class of applications in the test set. Performance metric values of this model on the test set are summarized in Table 5.5.

.

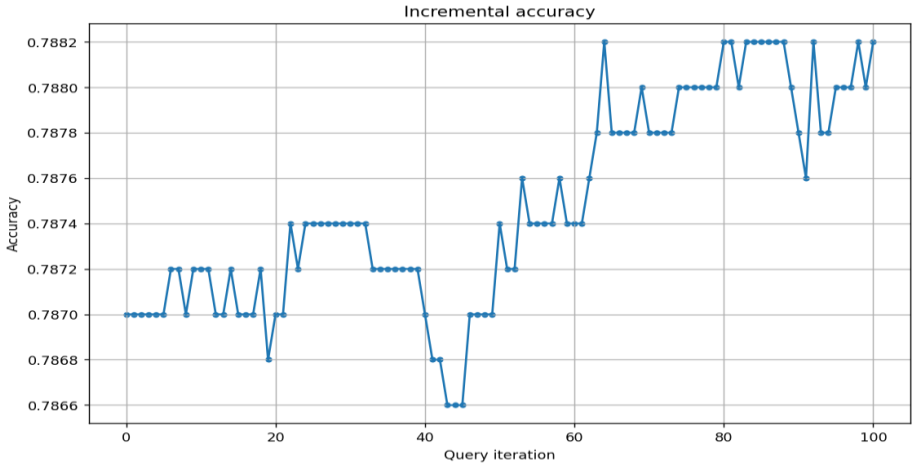

Figure 5.2: Accuracy curve of the LR model when reducing sample bias

- **KNN**

As with previous credit scoring models, the score of the KNN model is calculated on the set of accepted application only from the learner model built from the "*ActiveLearner"* function, which includes attributes such as the *"uncertainty_sampling"* as a query_strategy, the *"knn"* estimator estimated from the *"KNeighborsClassifier"* function and as well as the set of accepted applications. This score is then updated during the learning process. Then, the use of the classifier entropy function enables the quantification of the information contained in rejected applications, lying between 0 and 0.6. Rejected applications with entropy 0 are not informative and those with 0.6 are more informative. These are randomly selected and added to the training. Thus, in the learning process, the 100 most informative rejected applications with a classifier entropy score greater than the threshold, set at 0.5, are randomly selected, then added one by one to the training in order to update the model score, and finally removed from the set of rejected applications. As in SVM and LR models, the accuracy of the KNN model increases by adding the best rejected applications, as shown in Figure 5.3. The values of the performance metrics during training are shown in Table 5.4 and those of the prediction in the test set are summarized in Table 5.5.

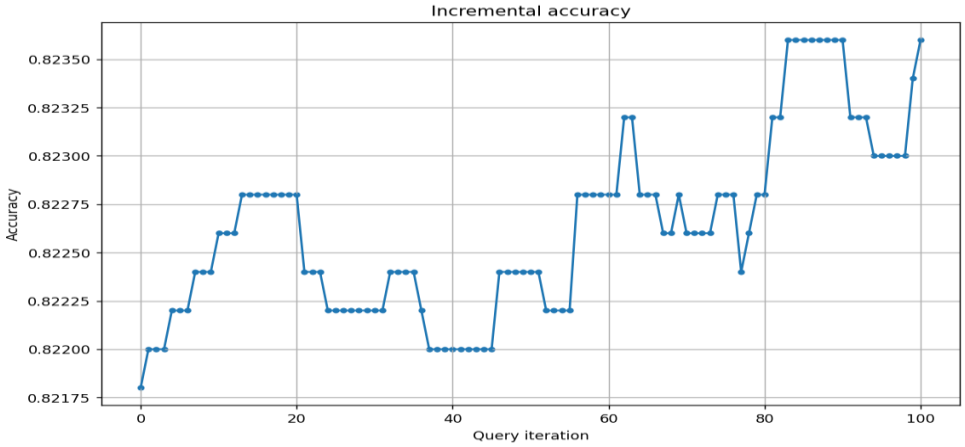

Figure 5.3: Accuracy curve of the KNN model when reducing sample bias

- **NB**

In the NB model, the score of the model, which is computed on the set of accepted applications from the learner model built from the *"ActiveLearner"* function, which integrates the *"GaussianNB()"* as estimator, the *"uncertainty_sampling"* as query strategy and also the set of only accepted applications, corresponds to the initial accuracy of the active learning process. In a next step, it is updated as the most informative rejected applications are added to the model. First, all applications from the pool of rejected applications are queried using the *"classifier_entropy"* function including the learner model built. Then, the most informative applications are selected, and then added one by one into the training set. After that, they are removed from the pool. From the pool of rejected applications, we randomly selected the 100 most informative rejected applications. An observation is considered more informative if its classification entropy score is greater than the minimum threshold of 0.5. After adding the most informative application to the process, the score is updated to achieve a higher point than the previous. This is illustrated in Figure 5.4. Tables 5.4 and 5.5 summarize performance measurement values of this model on training and test set, respectively.

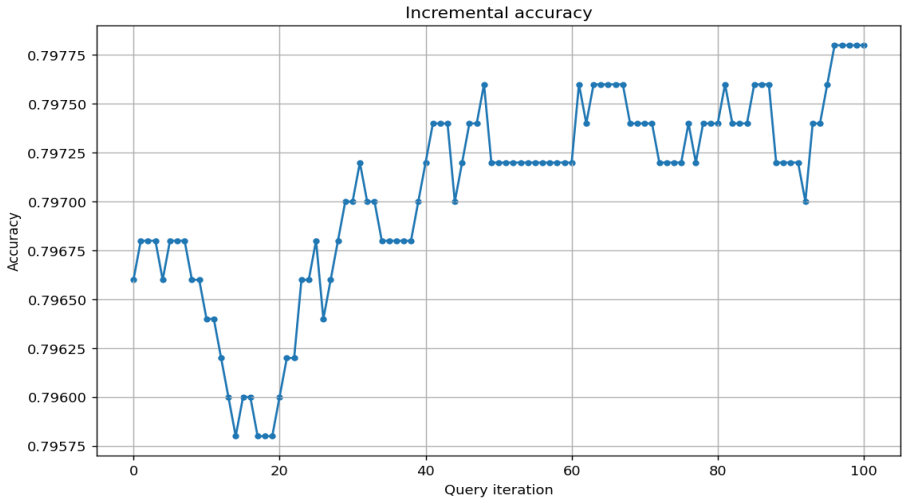

Figure 5.4: Accuracy curve of the NB model when reducing sample bias

- **DT**

For the DT model, the learning process is the same as for the previous models. This process begins with the determination of the model score, which is calculated on the set of accepted applications from the learner model built from the "*ActiveLearner"* function including the DT estimator, the classifier entropy attribute as query strategy and as well as the set of only accepted applications. Then, each rejected application from the pool is queried using the *"classifier_entropy"* function, which includes the learner model built. After that, the most informative rejected applications are randomly selected and then added one by one to the learning process. In this case, we selected 100 most informative applications, which have a classifier entropy score greater than the threshold of 0.5 and added them to the training. After each adding, the application is then

removed from the pool of rejected applications. Adding the most informative rejected applications does slightly improve the accuracy of the model as shown in the figure below. This model is then evaluated using performance metrics, the values of which are summarized in Table 5.4. This trained model is then used to make prediction of the creditworthiness of applicants in the test set. Table 5.5 summarizes performance metric values of this model on the test set.

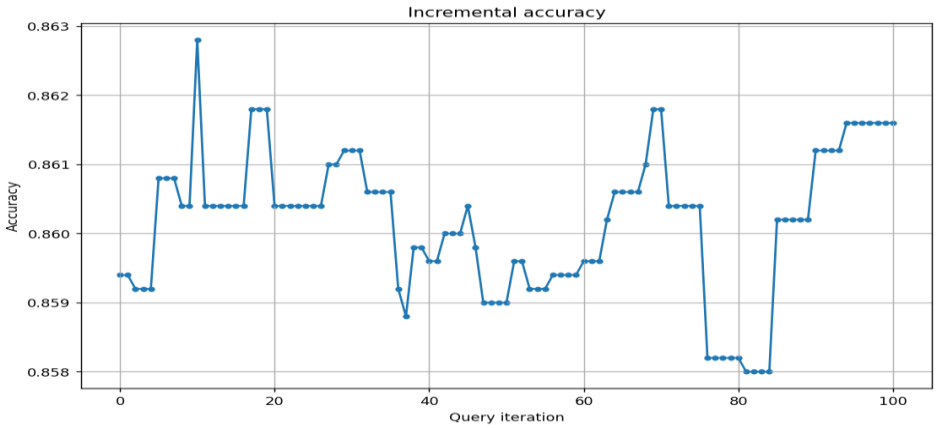

Figure 5.5: Accuracy curve of the DT model when reducing sample bias

- **NN**

In the NN model, the score of the model is computed on the sample of accepted applications from the learner model built using *"ActiveLearner"* function, which includes the *"uncertainty_sampling"* as the query strategy, the *"MLPClassifier()"* function as the estimator, and the set of accepted applications only. This score is then updated by the number of the most informative applications that are added into the training set. Then, the 100 most informative applications with a query greater than 0.6 were randomly selected from the pool of rejected applications and added one by one to the process. With each addition, the score is updated. This is shown in Figure 5.6. After each addition, the application is then removed from the set of rejected applications. The model is evaluated using performance measurements. The values of performance metrics of the NN model during the training are shown in Table 5.4. This model is then used to predict the default class of the applications in the test set. To evaluate the prediction of the model, performance metrics are used, the values of which are presented in Table 5.5.

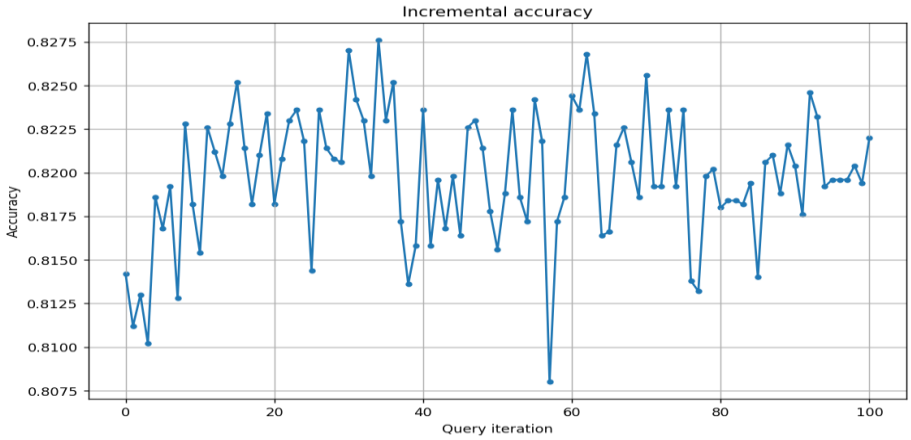

Figure 5.6: Accuracy curve of the NN model when reducing sample bias

The tables below summarize performance measurement values for each credit scoring model during training and test in the active learning process.

Model	Training set	
	Accuracy	AUC
LR	0.7882	0.7735
SVM	0.7914	0.7769
DT	0.8616	0.8318
KNN	0.8236	0.78
NB	0.7978	0.77
NN	0.822	0.7739

Table 5.4: Performance of credit scoring models when reducing sample bias using classification entropy.

By adding the most informative rejected applications into the training set based on the classification entropy metric, the performance of credit scoring models slightly increases (compared to Table 5.1) as shown in Table 5.4. The predictive performance of credit scoring models increases slightly when the sample bias is reduced. That is also true for AUC performance of each credit scoring model. Thus, the more informative rejected applications are added to the training, the more the performance of the model increases. Furthermore, the increase is higher in the NN model than in the other credit scoring models.

Model	Test set	
	Accuracy	AUC
LR	0.79054	0.7743
SVM	0.7941	0.775
DT	0.807	0.7677
KNN	0.8135	0.7606
NB	0.8016	0.7727
NN	0.8226	0.7722

Table 5.5: Performance of credit scoring models on the test set when reducing sample bias using classification entropy.

Increasing the performance of credit scoring models when reducing sample bias leads to a slight improvement in their performance in the test set (compared to Table 5.2) as shown in the table above. However, only the performance of the KNN model decreases on the test set. Moreover, as in the training process, the improvement is higher in the NN model in the test set. This performance can also be evaluated from the confusion matrix presented in the following table:

Models	Confusion matrix			
	TP	FP	TN	FN
LR	5708	1298	2195	796
KNN	6252	754	1881	1110
SVM	5767	1239	2172	819
DT	6054	952	1981	1010
NB	5918	1088	2096	895
NN	6288	718	1935	1056

Table 5.6: Confusion matrix of credit scoring models when reducing sample bias using classification entropy.

Adding the most informative rejected applications to the training improves the prediction as shown in Table 5.6. Compared to Table 5.3, this table shows an increase of the number of TP and a decrease of the number of FP in credit scoring models. Thus, the increase in the predictive performance of the model reduces the misclassification.

- **Classification uncertainty**

The process of each credit scoring model is the same as in classification entropy. The score for each credit scoring model is first computed on the sample of accepted applications from the learner model, which is built using the *"ActiveLearner"* function including the estimator of each credit scoring model, the "uncertainty_sampling" as a query strategy and the data set of only accepted applications. Using the *"ActiveLearner"* function, the model can learn from the features of accepted applications in order to select the best rejected applications to be added into the training set. The calculated score corresponds to the initial score of the learning process and will be updated during the active learning process. After that, the set of rejected applications is queried using the *"uncertainty_sampling"* function including the learner model. This function makes it possible to quantify the information contained in rejected applications. From there, the most informative queries are selected and added one by one into the training set. The most informative rejected applications are those with the highest uncertainty sampling scores. The more uncertain an application is, the more informative it is. Then, the active learning process is performed. It can be described as follows: from the pool of rejected applications, the 100 most informative applications, which exceeds the minimum uncertainty sampling score or threshold, are randomly selected and then added to the process. After that, the *".teach(X,y)"* is used to teach the selected applications. With each addition of the most informative rejected application, the score of credit scoring models is improving as shown in Figure 5.7. The added application is then removed from the pool of rejected applications using the *"np.delete"* function. Moreover, the trained model is used to predict the creditworthiness of the applicants in the test set.

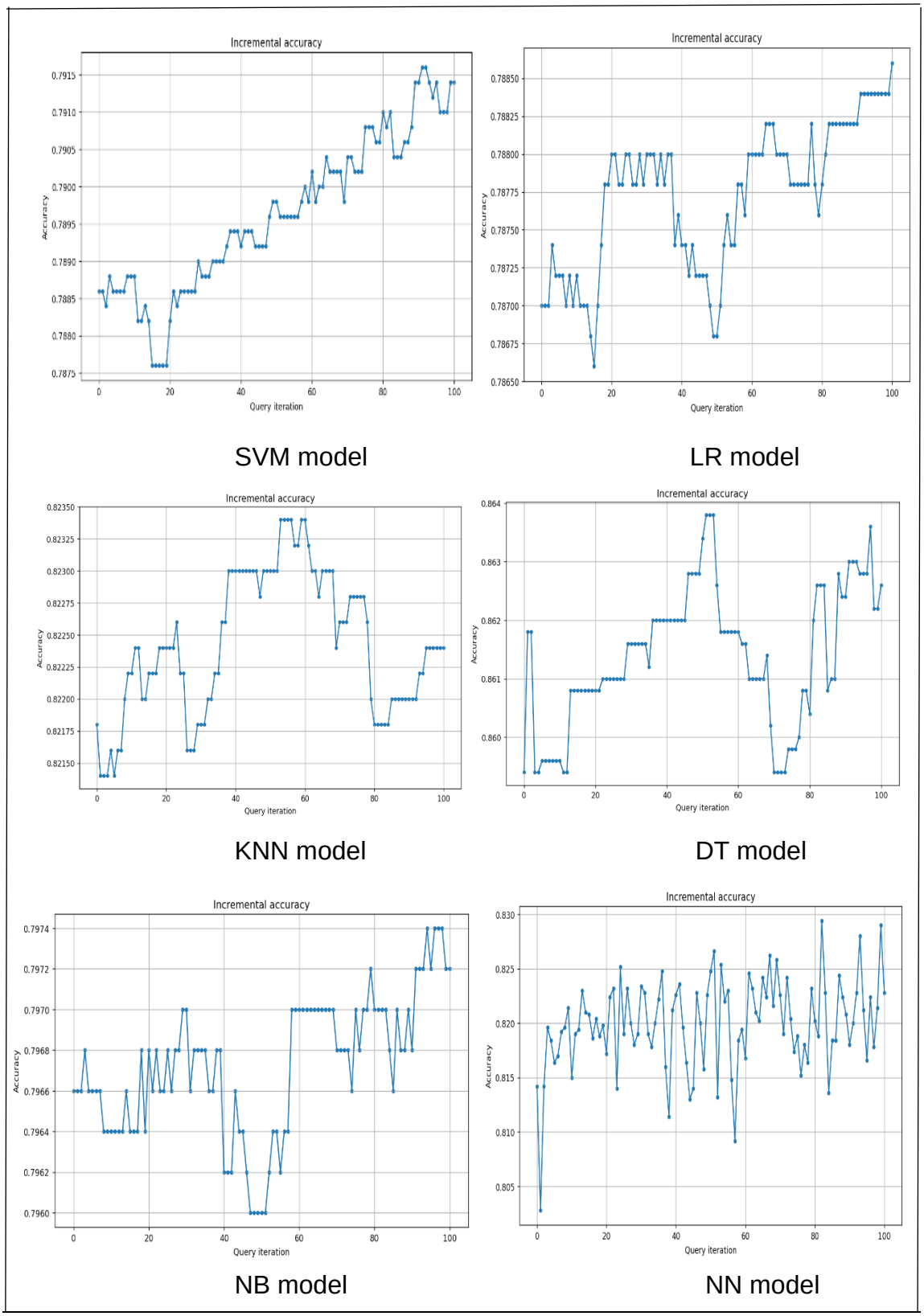

Figure 5.7: Accuracy curve of credit scoring models when reducing sample bias using classification uncertainty

The following tables summarize performance measurement values for each credit scoring model on the training and the test set, resulting from the use of classification uncertainty as query strategy in the active learning process.

Model	Training set	
	Accuracy	AUC
LR	0.7886	0.7736
SVM	0.7914	0.7739
DT	0.8626	0.83
KNN	0.8224	0.7747
NB	0.7972	0.77
NN	0.8228	0.7779

Table 5.7: Performance of credit scoring models when reducing sample bias using classification uncertainty

The table above shows that the use of classification uncertainty to select the most informative rejected applications that will then be added to the training for reducing sample bias in the training data improves the performance metrics of the credit scoring models slightly. The predictive performance and AUC of credit scoring models increase slightly (compared to table 5.1) when sample bias is reduced. Just like in classification entropy, the improvement is higher in the NN model than in the other credit scoring models.

Model	Test set	
	Accuracy	AUC
LR	0.79054	0.7734
SVM	0.7943	0.775
DT	0.801	0.7564
KNN	0.8126	0.7587
NB	0.8017	0.7731
NN	0.8248	0.777

Table 5.8: Performance of credit scoring models on the test set when reducing sample bias using classification uncertainty.

Table 5.8 shows the performance measurements values of the credit scoring models on the test set when reducing sample bias. It shows a certain improvement in the accuracy of credit scoring models. Compared to table 5.2, the AUC performance of some credit scoring models including LR, SVM and NN also increases slightly. On the other hand, it decreases moderately in the other credit scoring models such as KNN, DT and NB. Moreover, the evaluation of the performance of credit scoring models when reducing sample bias, can also be done by the confusion matrix presented in the table below:

Models	Confusion matrix			
	TP	FP	TN	FN
LR	5717	1289	2186	805
KNN	6256	750	1868	1123
SVC	5769	1237	2172	819
DT	6125	881	1853	1138
NB	5916	1090	2099	892
NN	6319	687	1876	1115

Table 5.9: Confusion matrix of credit scoring models when reducing sample bias using classification uncertainty

As outlined in Table 5.9, reducing sample bias using classification uncertainty improves the prediction of the creditworthiness of applicants. Compared to Table 5.3, this table shows an increase in the number of TP and a decrease in the number of FP when sample bias is reduced.

- **Classification margin**

The classification margin metric is a another informativeness measure used to quantify the information in applications. The query of a set of rejected applications is done with the *"classifier_margin"* function, which includes the learner model. This provides a score for each rejected application. The higher this score, the more informative the application. Thus, the most informative rejected applications are selected and added one by one into the training set during the active learning process. Like in previous processes, the active learning process using classification margin also begins with the determination of the score of the credit scoring model. This is calculated only on the basis of the set accepted applications from the learner model built from the "*ActiveLearner"* function including the core estimator of the credit scoring model, the "*uncertainty_sampling"* as query strategy, and only the set of accepted applications as training set. From this function, the learner model can learn from features of accepted applications in order to select the best rejected applications to be included in the process. Then, the 100 most informative applications with a margin score above the minimum threshold are randomly selected from the pool of rejected applications and added into the training set. Each addition improves the score of the credit scoring model as shown in Figure 5.8. Then, the added application is removed from the set of rejected applications. Finally, the trained model is used to make predictions of the creditworthiness of the applicants in the test set. Tables 5.10 and 5.11 summarized performance metric values of the credit scoring models on training and test set when reducing sample bias using classification margin.

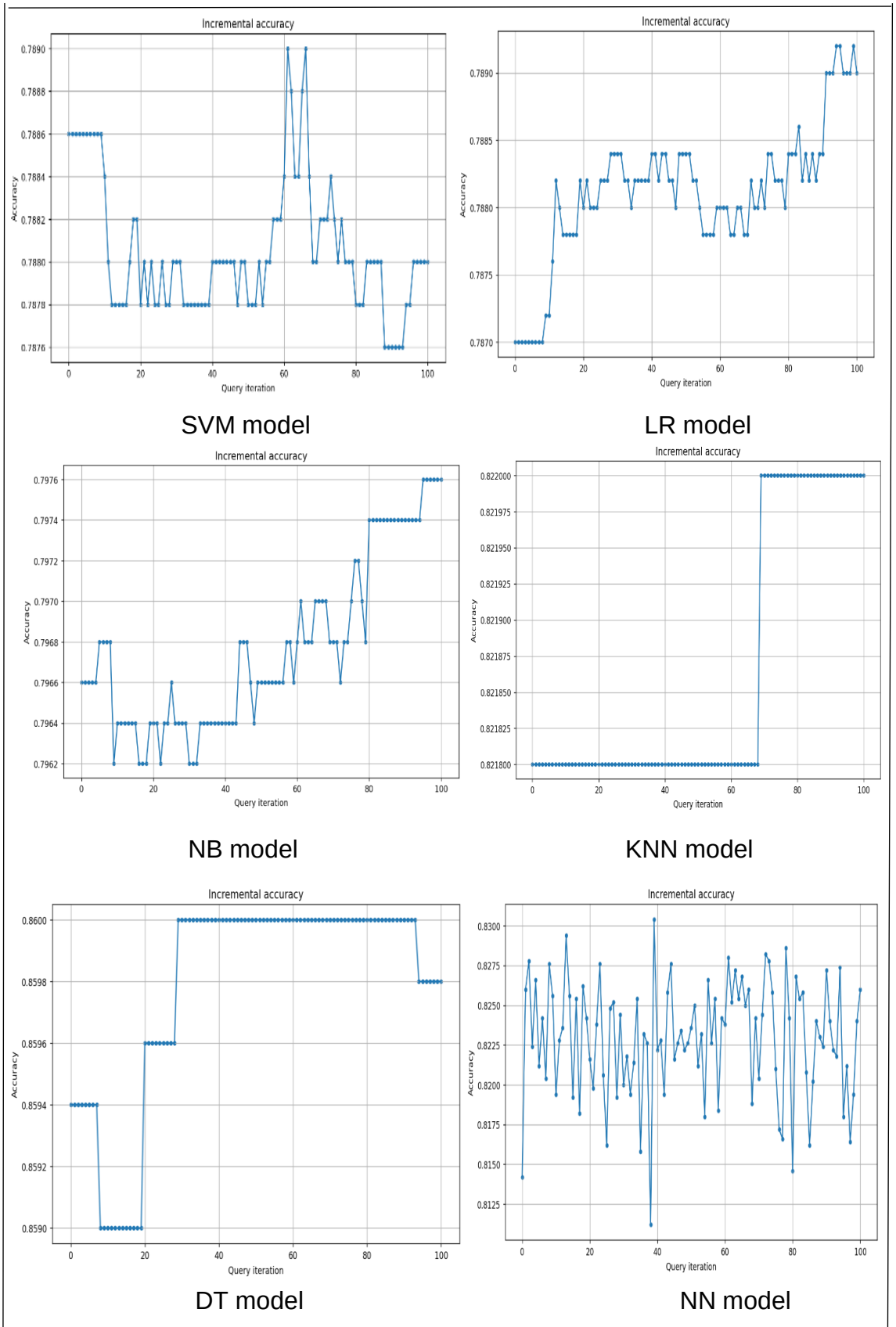

Figure 5.8: Accuracy curve of credit scoring models when reducing sample bias using classification margin

Models	Training set	
	Accuracy	AUC
LR	0.789	0.775
SVM	0.788	0.7727
DT	0.8598	0.833
KNN	0.822	0.7807
NB	0.7976	0.7707
NN	0.826	0.7943

Table 5.10: Performance of credit scoring models when reducing sample bias using classification margin.

Table 5.10 shows a slight improvement in the predictive performance and the AUC of all credit scoring models (compared to table 5.1) when reducing sample bias by adding the most informative rejected applications into the training set based on classification margin. As with the previous uncertainty query strategies, performance improvement is higher in the NN model than in the other scoring models. Increasing predictive performance during the training improves the performance of credit scoring models on the test set as shown in the table below:

Model	Test set	
	Accuracy	AUC
LR	0.79044	0.7752
SVM	0.793	0.7758
DT	0.8035	0.7665
KNN	0.8179	0.7723
NB	0.8008	0.7751
NN	0.8235	0.7883

Table 5.11:Performance of credit scoring models on the test set when reducing sample bias using classification margin

Compared to Table 5.2, Table 5.11 shows that reducing sample bias by adding the most informative rejected applications to the training using classification margin slightly improves the accuracy and AUC of credit scoring models on the test set. Furthermore, the performance of credit scoring models when reducing sample bias can also be assessed using the confusion matrix presented in the following table:

Models	Confusion matrix			
	TP	FP	TN	FN
LR	5697	1309	2205	786
KNN	6205	801	1971	1020
SVC	5736	1270	2192	799

DT	6016	990	2017	974
NB	5879	1127	2127	864
NN	6188	818	1813	1178

Table 5.12: Confusion matrix of credit scoring models when reducing sample bias using classification margin

when the sampling bias is reduced, the performance of credit scoring models increases, leading to improve the number of correctly predicted applications and decrease the number of wrongly predicted applications, as shown in table above. Compared to Table 5.3, Table 5.12 shows an increase in the number of TP and a decrease in the number of FP for the credit scoring models.

- **Comparison of the Uncertainty query strategies**

The following table summarizes the predictive performance of credit scoring models based on accepted applications only and their predictive performance when reducing sample bias using active learning process in which uncertainty query strategies including classification entropy, classification uncertainty, and classification margin were used to select the most informative rejected applications to be added into the training set. From this table, we can determine which strategy improves the scoring model best by comparing the query strategies with each other:

Model	Accuracy (accepted only)	Classification entropy	Classification uncertainty	Classification margin
LR	0.7870	0.7882	0.7886	0.789
SVM	0.7886	0.7914	0.7914	0.788
DT	0.8594	0.8616	0.8626	0.8598
KNN	0.8218	0.8236	0.8224	0.822
NB	0.7966	0.7978	0.7972	0.7976
NN	0.8142	0.822	0.8228	0.826

Table 5.13: Accuracy comparison of credit scoring models when reducing sample bias using stream-based uncertainty query strategies

We can see from Table 5.13 that all query strategies slightly improve the predictive performance of credit scoring models. The LR model and the NN model increase better with classification margin strategy than with other uncertainty query strategies. Classification uncertainty and classification entropy improve the SVM model. The DT model gets better by using classification uncertainty. Moreover, the predictive performance of the KNN model and NB model increase better when classification entropy is used. However, the gap of the model accuracy between the different uncertainty query strategies is very small. Thus, we see that classification entropy and classification uncertainty are uncertainty query strategies that improve the scoring models more. Finally, we can say that the stream-based uncertainty sampling approach from the active learning

technique increases the performance of credit scoring models by adding the most informative rejected applications, thus reducing the sample bias.

- **Query strategy: QBC**

The QBC approach is based on the measure of the disagreement among committee members over the attribution of a label to an instance. For this, the build of the committee is required. In the application, committee members are built from credit scoring models (including the LR model, the SVM model, the NB model, the DT model, the KNN model and the NN model) and a query strategy function. In the stream based QBC strategy, there are two query strategies for measuring the disagreement of the committee members, namely vote entropy and KL divergence. These strategies are located in the *"modAL.disagreement"* package. They are used to query the pool of rejected applications. The higher the score of the application, the greater the disagreement of committee members about a query. The aim of this QBC approach is to query the rejected applications with the highest degree of disagreement. These represent the most informative observations and are added one by one to the training in order to improve the performance of the prediction.

- **Vote Entropy**

The committee is built from the *"committee"* function, which includes the list of learner models (built from the "*ActiveLearner"* function) and vote entropy as the query strategy. Thus, the initial learner score is computed from the set of accepted applications using the *"committee.score()"* function. This score will be updated in the QBC process. After that, the pool of rejected applications is queried to determine the degree of disagreement of committee members for each application. This is done by using the *"vote_entropy"* function including the committee previously built. The higher the degree of disagreement, the more the committee members disagree. The rejected applications with a high degree of disagreement are selected and added one by one to the training. By querying the pool of rejected applications, the degree of disagreement of credit scoring model committee is between 0 and 0.6. In the QBC process, the 100 most informative rejected applications, which have a vote entropy score higher than the minimum threshold equal to 0.6, are randomly selected from the pool of rejected applications and added to the training. Then, the ".teach(X,y)" function of the committee class is used to teach the new applications in the committee for performance improvement. With each addition of the most informative rejected application, the committee score is updated as shown in Figure 5.9 below. The added applications are then removed from the pool of rejected applications. The process is repeated until all the 100 most informative rejected applications are added to the process. Tables 5.14 and 5.15 summarize the performance metric values of the committee of credit scoring models on the training set and the test set before and after sample bias reduction, respectively.

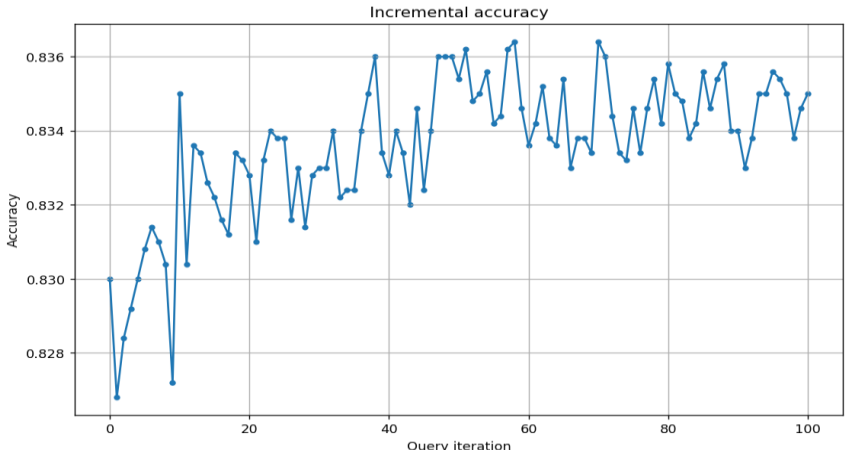

Figure 5.9: Predictive performance curve of the credit scoring model committee when reducing sample bias using the vote entropy strategy.

- **KL divergence**

The process of using KL divergence is the same as that of vote entropy. The committee is first built using the list of committee members and "*kl max disagreement"* as a query strategy. It helps then to calculate the initial score of the committee, which will be updated during the QBC process. The pool of rejected applications is then queried using the *"KL_max_disagreement"* function including the committee built to determine the degree of disagreement of committee members for each application. The degree of disagreement of the set of rejected applications is between 0.1 and 0.9. As a result, rejected applications with a higher degree of disagreement are added one by one to the QBC process. As in vote entropy process, we randomly selected the 100 most informative rejected applications with a degree of disagreement greater than the minimum threshold, set at 0.8, and added them to the process. With each addition, the score is updated as shown in Figure 5.10. Then, the added application is deleted from the pool of rejected applications. The performance metric values of the credit scoring model committee on the training and test set when reducing sample bias using KL divergence are respectively summarized in Tables 5.14 and 5.15.

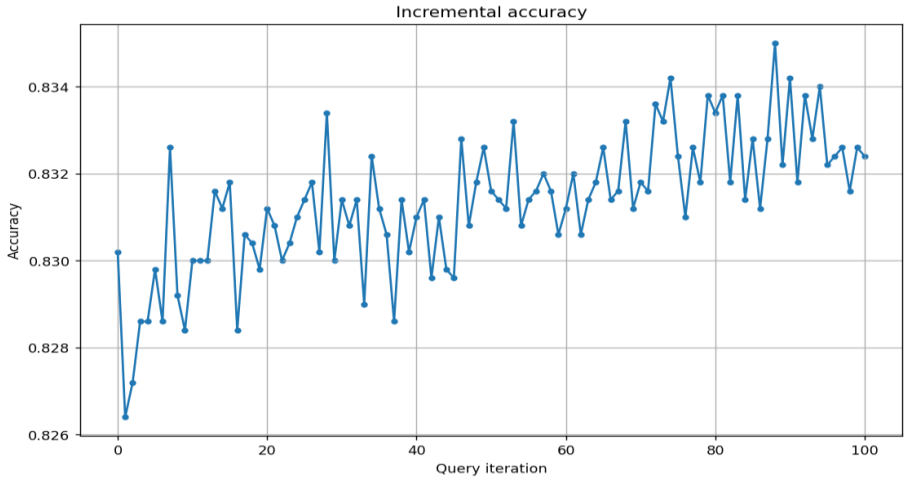

Figure 5.10: Predictive performance curve of the credit scoring model committee when reducing sample bias using the KL Divergence strategy.

Approach	Training set	
	Accuracy	AUC
Committee (accepted applications only)	0.8302	0.7884
Vote Entropy	0.835	0.7892
KL Divergence	0.8324	0.7903

Table 5.14: Performance of the credit scoring model committee when reducing sample bias using stream based QBC query strategies.

The table above compares the performance of the credit scoring model committee based solely on accepted applications with its performance when reducing sample bias using vote entropy and KL divergence. It shows that the performance of the credit scoring model committee slightly increases when sample bias is reduced by adding the most rejected applications based on QBC measures. Moreover, vote entropy improves the predictive performance than KL divergence. This is illustrated in Figure 5.11 below:

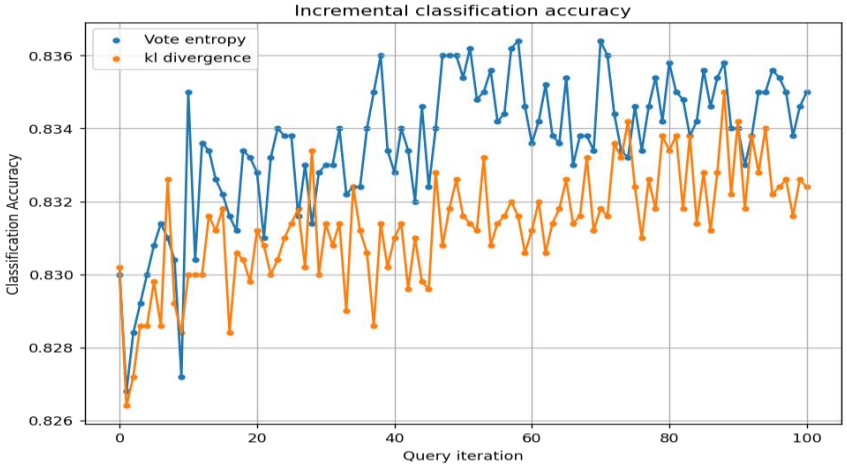

Figure 5.11: Predictive performance comparison of the credit scoring model committee when reducing sample bias using stream based QBC strategies.

This figure shows that the committee score throughout the QBC process increases most when vote entropy strategy is used to select the most informative rejected applications to be added into the training set.

Approach	Test set	
	Accuracy	AUC
Committee (accepted applications only)	0.8181	0.7792
Vote Entropy	0.8204	0.7755
KL Divergence	0.8223	0.7828

Table 5.15: Performance of the credit scoring model committee on the test set when reducing sample bias using stream based QBC query strategies

As we can see from Table 5.15 above, the predictive performance and AUC of the credit scoring model committee on the test set improve slightly when sample bias is reduced by using vote entropy and KL divergence. Finally, we can say that

the stream based QBC approach improves the performance of credit scoring models when reducing sample bias. Table 5.16 below shows the confusion matrix of QBC query strategies.

Approach	Confusion matrix			
	TP	FP	TN	FN
Committee	6138	868	2041	950
Vote Entropy	6217	789	1985	1006
KL Divergence	6174	832	2047	944

Table 5.16: Confusion matrix of the credit scoring model committee when reducing sample bias using stream based QBC strategies

The table above shows an improvement in the prediction of the credit scoring model committee when sample bias is reduced using QBC query strategies. They increase the number of correctly predicted values and decreases the number of wrongly predicted values by the committee. In KL divergence, the number of TP and TN increases and that of FP and FN decreases. In vote entropy strategy, the number of TP increases and the number of FP decreases. These QBC query strategies hence reduce the misclassification error.

❖ **Scenario 2: Pool-based active learning**

In the pool-based active learning scenario, the set of rejected applications is queried by using query strategies and the most informative one is selected. This most informative rejected application is added to the training process in order to improve the predictive performance of the credit scoring model. In this scenario, we have implemented three query strategies, namely uncertainty sampling, QBC and density-weighted:

➢ **Query strategy: Uncertainty sampling**

In pool-based uncertainty sampling, three measures are used to query the most informative applications from a set of rejected applications. These are entropic sampling, least confident sampling, and margin sampling. These measures are imported from the *"modAL.uncertainty"* package.

- **Entropy sampling**

The process starts first with determining the score of the credit scoring model. This is calculated using the set of accepted applications from the learner model, which is built using *the"ActiveLearner"* function including the core estimator of the credit scoring model, the *"entropy_sampling"* as the query strategy and the set of accepted applications. Then, this score is updated as the most informative rejected applications are added in the learning process. This is performed by allowing the learner's model to query the set of rejected applications and then to select the most informative application, which will then be added to the training. Thus, based on the number of queries set, the pool of rejected applications is

queried using the *".query(X)".* Then, the *".teach(X,y)"* function is used to teach the learner model the instance it has queried. After that, the most informative rejected application is selected and added to the learning process. The added application is removed from the pool of rejected applications afterwards. Finally, the new score is recorded. The process is repeated until the most informative rejected applications (the number of queries) are added to the training. In the active learning process of credit scoring models, we added the 100 most informative applications from the set of rejected applications into the training. Each addition improves the score of the credit scoring model as shown in Figure 5.12.

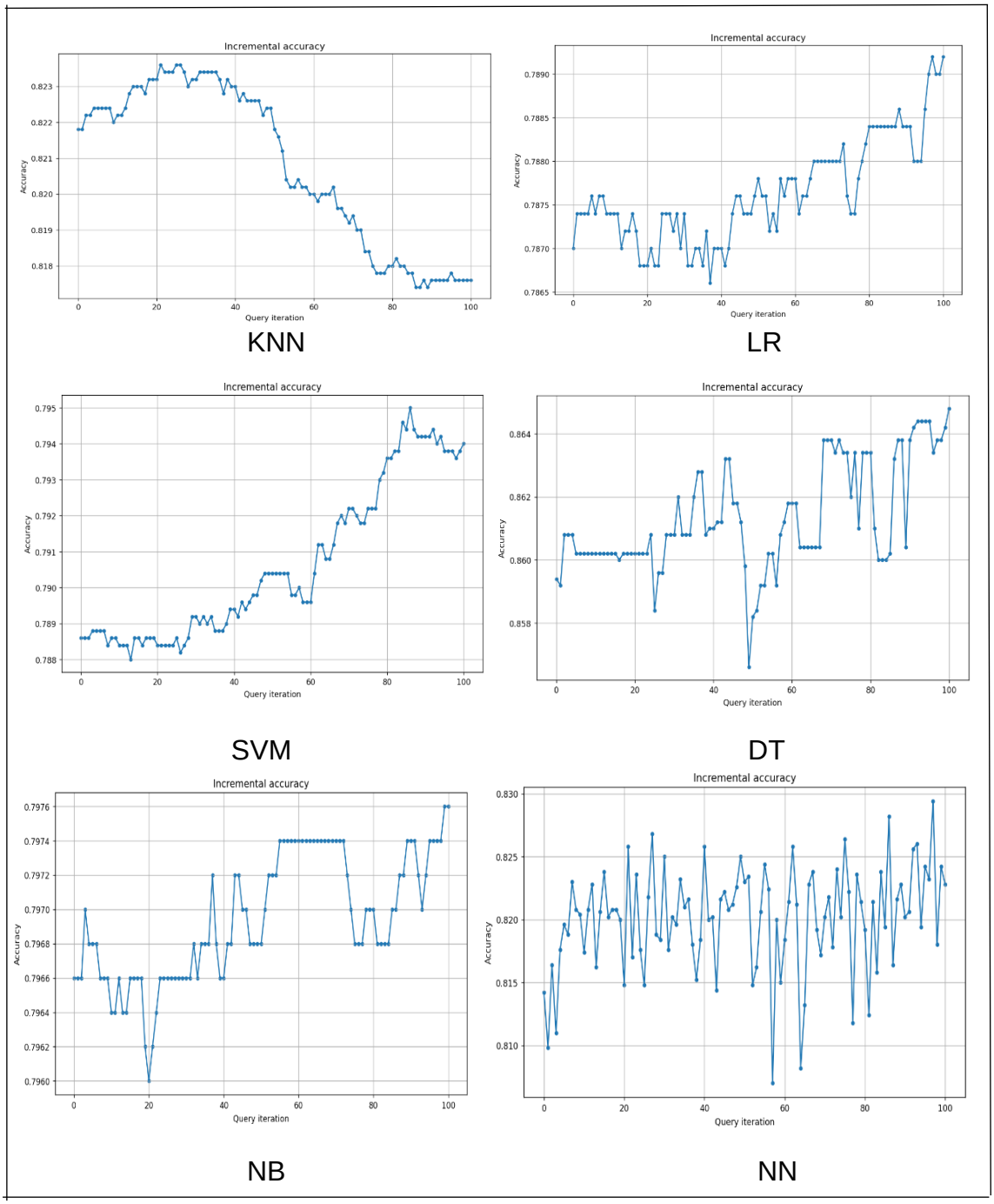

Figure 5.12: Accuracy curve of credit scoring models when reducing sample bias using entropy sampling

This figure shows that only the predictive performance of the KNN model decreases with each addition of the most rejected application.

The tables below summarize performance metric values of credit scoring models on the training set and the test set when reducing sample bias using entropy sampling.

Models	Training set	
	Accuracy	AUC
LR	0.7892	0.7739
SVM	0.794	0.7743
DT	0.8648	0.8218
KNN	0.8176	0.7643
NB	0.798	0.77
NN	0.8228	0.7946

Table 5.17: Performance of credit scoring models on the training set when reducing sample bias using entropy sampling

Table 5.17 shows that the performance of credit scoring models, with the exception of the KNN model, improves slightly (compared to table 5.1) when sample bias is reduced by using entropy sampling. Only the predictive performance of the KNN model decreases slightly. Moreover, the increase in the performance is higher in the NN model than in the other credit scoring models. Thus, the more informative rejected applications are added to the training, the better is the predictive performance of credit scoring models. The increase in the predictive performance improves the performance of credit scoring models on the test set as shown in the following table:

Model	Test set	
	Accuracy	AUC
LR	0.7905	0.7731
SVM	0.7973	0.7753
DT	0.8067	0.765
KNN	0.8122	0.7539
NB	0.8010	0.7718
NN	0.8236	0.7914

Table 5.18: Performance of credit scoring models on the test set when reducing sample bias using entropy sampling

As we can see from Table 5.18, the accuracy and AUC of credit scoring models on the test set, with the exception of KNN, improve slightly (compared to Table 5.2) when sample bias is reduced. The decrease in the performance of the KNN model is due to the fact that its predictive performance has decreased during the learning process. Table 5.19 illustrates the confusion matrix of the credit scoring models.

Models	Confusion matrix			
	TP	FP	TN	FN
LR	5721	1285	2182	836
KNN	6300	706	1820	1171
SVM	5816	1190	2155	819
DT	6188	818	1813	1178
GM	5917	1089	2091	900
NN	6105	901	2128	863

Table 5.19: Confusion matrix of credit scoring models when reducing sample bias using entropy sampling strategy.

Increasing the predictive performance by adding the most informative rejected application into the training set based on entropy sampling improves the quality of the prediction of credit scoring models as illustrated in Table 5.19. Compared to Table 5.3, this table shows an increase of TP and a decrease of TP in the credit scoring models.

- **Least Confident sampling**

As with entropy sampling, the learner model of the credit scoring model is built from the *"ActiveLearner"* function, which includes the core estimator of the credit scoring model, the *"uncertainty_sampling"* attribute as the query strategy and the set of accepted applications from which it learns. The process starts with determining the score of the credit scoring model, which is calculated from the learner model on the set of accepted applications only. This score is then updated as the most informative applications are added during the active learning process. After that, the learner model requests a set of rejected applications using the *".query(X)"* function and teach the new applications using the *".teach(X,y)"* function. Thus, based on uncertainty sampling, the most informative rejected application is selected, and added to the training. Finally, the added application is removed from the pool of rejected applications and the new score of the credit scoring model is recorded. Using least confident sampling, we added the 100 most informative rejected applications to the training during the active learning process of credit scoring models. Figure 5.13 graphically presents the score of each credit scoring model with each addition of the most informative rejected application. This figure shows an increase in the predictive performance of credit scoring models, with the exception of the KNN model, when reducing sample bias by adding the most informative rejected applications into the training set:

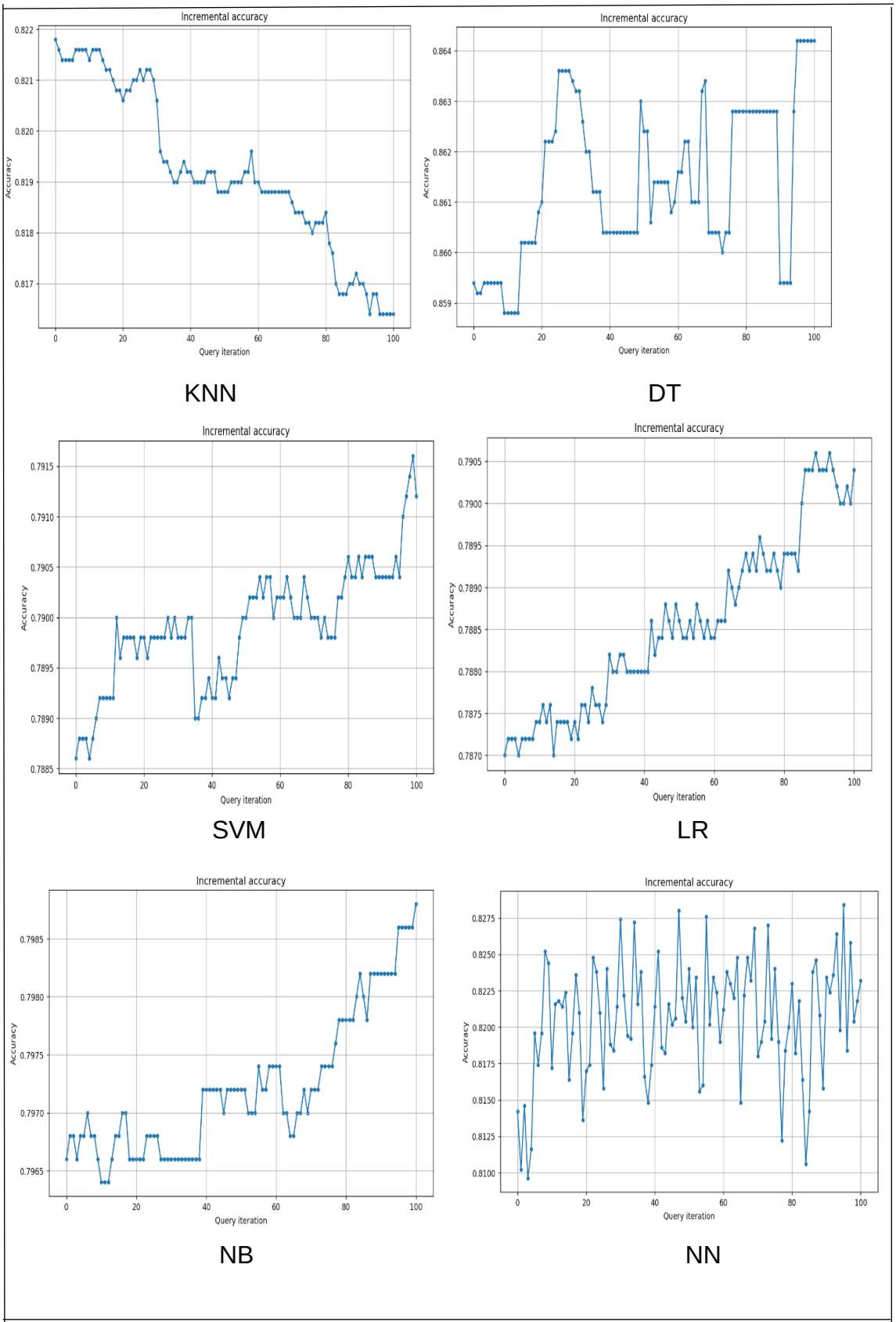

Figure 5.13: Accuracy curve of credit scoring models when reducing sample bias using least confident sampling

The following table shows the values of the performance metrics of credit scoring models on the training set when sample bias is reduced using least confident sampling:

Model	Training set	
	Accuracy	AUC
LR	0.7904	0.7752
SVM	0.7912	0.7769
DT	0.8656	0.8335
KNN	0.8164	0.7640
NB	0.7988	0.77
NN	0.8232	0.7723

Table 5.20: Performance of credit scoring models when reducing sample bias using least confident sampling.

Compared to Table 5.1, the table above shows that, the predictive performance and AUC of credit scoring models except the KNN model increase when sample bias is reduced using least confident sampling. However, the decrease in the performance of the KNN model is small. As with entropy sampling, the performance improvement is higher in the NN model than in the other models. This performance improvement increases the performance of credit scoring models on the test as shown in Table 5.21:

Model	Test set	
	Accuracy	AUC
LR	0.7929	0.7762
SVM	0.7952	0.7751
DT	0.8052	0.7516
KNN	0.8115	0.7536
NB	0.8013	0.7720
NN	0.8227	0.7683

Table 5.21: Performance of credit scoring models on the test set when reducing sample bias using least confident sampling

Compared to Table 5.2, we can see from Table 5.21 that the performance metrics of the credit scoring models increase slightly when reducing sample bias from the use of the least confident sampling. However, the decrease in the predictive performance of the KNN model deteriorates its performance on the test set as shown in the table. The performance of credit scoring models when reducing

sample bias can also be evaluated from the confusion matrix presented in Table 5.22:

Models	Confusion matrix			
	TP	FP	TN	FN
LR	5730	1276	2197	794
KNN	6290	716	1823	1168
SVM	5781	1225	2169	822
DT	6201	805	1849	1142
NB	5920	1086	2091	900
NN	6331	675	1893	1098

Table 5.22: Confusion matrix of credit scoring models on the test set when reducing sample bias using least confident sampling.

From this table, we see an improvement of the predicted values by the credit scoring models. It shows an increase in the number of TP and a decrease in the number of FP for credit scoring models (Compared to Table 5.3). Moreover, we can see from this table that improving the predictive performance of credit scoring models increases the number of correctly predicted applications and decreases the number of wrongly predicted applications.

- **Margin sampling**

Using margin sampling, the learner model is thus built from the *"ActiveLearner"* function including the estimator of the credit scoring model, the *"margin_sampling"* as the query strategy and a set of accepted applications as training sample. The active learner process begins by determining the score of the credit scoring model, which is calculated on the set of accepted applications only using the learner model that has been built. This score will be updated during the active learning process. Then, a set of rejected applications is queried in order to select the most informative rejected application that will be added into the training to improve the model score. This is done using the *"query(X)* and *"teach(X,y)* functions. Finally, the added application is deleted from the pool of rejected applications. This process is repeated until the number of queries is reached. In the process, we added the 100 most informative rejected applications to the training during the active learning process of credit scoring models. Figure 5.14 illustrates the change in the score of each credit scoring model with each addition of the most informative rejected application. This figure shows that adding the most informative rejected applications based on margin sampling improves slightly the predictive performance of credit scoring models, with the exception of the KNN model. Tables 5.23 and 5.24 show the performance metric

values of credit scoring models on the training set and the test set when reducing sample bias, respectively.

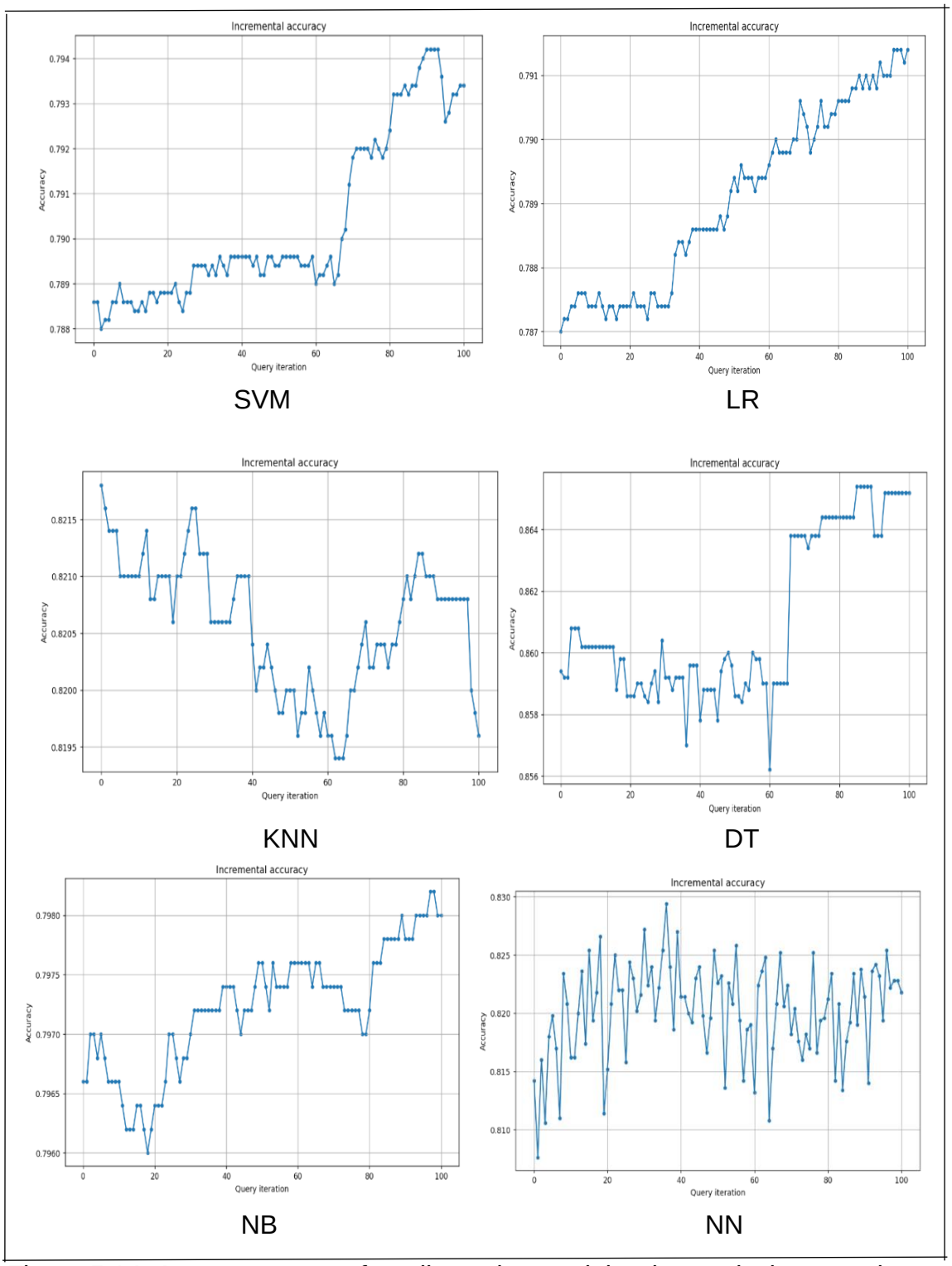

Figure 5.14: Accuracy curve of credit scoring models when reducing sample bias using margin sampling.

Model	Training set	
	Accuracy	AUC

LR	0.7914	0.7756
SVM	0.7934	0.7741
DT	0.8654	0.8365
KNN	0.8196	0.7693
NB	0.798	0.77
NN	0.8218	0.7745

Table 5.23: Performance of credit scoring models when reducing sample bias using margin sampling.

Compared to Table 5.1, the table above shows a slight improvement in the predictive performance and AUC of credit scoring models, with the exception of KNN, when reducing sample bias using margin sampling. It shows a slightly decrease in the performance of the KNN model. As with the other uncertainty query strategies, the performance improvement is higher in the NN model increase more than in the other credit scoring models.

Model	Test set	
	Accuracy	AUC
LR	0.7929	0.7761
SVM	0.7972	0.7760
DT	0.8115	0.7746
KNN	0.8107	0.7536
NB	0.8012	0.7724
NN	0.8219	0.7691

Table 5.24: Performance of credit scoring models on the test set when reducing sample bias using margin sampling.

The table above shows the performance of credit scoring models on the test set when sample bias is reduced. It shows a slight improvement in the accuracy and AUC of credit scoring models except KNN (compared to Table 5.2). The performance of credit scoring models can also be assessed from the confusion matrix presented in the following table:

Models	Confusion matrix			
	TP	FP	TN	FN
LR	5731	1275	2196	795
KNN	6277	729	1828	1163
SVM	5807	1199	2163	828
DT	6174	832	1849	1142
NB	5915	1091	2095	896

NN	6310	696	1907	1084

Table 5.25: Confusion matrix of credit scoring models when reducing sample bias using margin sampling.

Table 5.25 shows an improvement of the number of TP and a decrease of the number of FP in the credit scoring models when sample bias is reduced (compared to Table 5.3). Therefore, the improvement in the performance of the credit scoring model increases the number of correctly predicted applications and decreases the number of wrongly predicted applications.

- **Comparison of the Uncertainty query strategies**

In the following table, we compare the predictive performance of credit scoring models when minimizing sample bias using pool-based uncertainty query strategies including entropy sampling, least confident sampling, and margin sampling in order to determine which query strategy best increases the predictive performance of the models.

Model	Model score (accepted only)	Entropy sampling	Least confident sampling	Margin sampling
LR	0.7870	0.7892	0.7904	0.7914
SVM	0.7886	0.794	0.7912	0.7934
DT	0.8594	0.8648	0.8656	0.8654
KNN	0.8218	0.8176	0.8164	0.8196
NB	0.7966	0.798	0.7988	0.798
NN	0.8142	0.8228	0.8232	0.8218

Table 5.26: Accuracy comparison of credit scoring models when reducing sample bias using pool-based uncertainty query strategies

We can see from Table 5.26 that the use of uncertainty query strategies slightly increases the predictive performance of credit scoring models with the exception of the KNN model. It shows that the predictive performance of LR, NB using margin sampling is higher than that using least confident sampling and margin sampling. Least confident sampling improves DT and NN more than the other uncertainty query strategies. Moreover, the improvement in the predictive performance of SVM using entropy sampling is stronger than that of the other uncertainty query strategies. In conclusion, we can say that the pool-based uncertainty sampling approach improves the performance of credit scoring models when sample bias decreases.

- **Query strategy: QBC**

In the QBC approach, queries are selected by measuring the credit scoring model committee disagreement. The credit scoring model committee is built from the *"committee"* function, which includes the list of learner credit scoring models (built using *"ActiveLearner"* function) and a query strategy function. In the pool-based QBC approach, vote entropy sampling and KL divergence sampling are used as query strategies to measure the disagreement of the credit scoring model committee. These query strategies are provided from the python package *"modAL.disagreement"*. The more the committee disagrees on a query, the more informative it is. The most informative application from the set of rejected applications is selected and added to the training.

- **Vote entropy sampling**

The QBC process begins by determining the score of the credit scoring model committee. For this, the committee is built from the *"committee"* function including the list of learning credit scoring models and the *"vote_entropy_sampling"* as the query strategy. This score is calculated only on the basis of the set of accepted applications. Then, the set of rejected applications is queried, in order to select the most informative rejected application that will be added to the training in order to improve the committee' accuracy. This is done by using the *".query(X,y)"* function and then the *".teach (X,y)"* function to teach the new applications. Finally, the added application is removed from the pool of rejected applications. The process is repeated until the most informative rejected applications (number of queries) are added into the training. In this case, we queried the 100 most informative applications from the pool of rejected applications. With each addition of the most informative rejected application, the committee score is updated as shown in Figure 5.15. Moreover, Performance metric values of the credit scoring model committee on the training set and the test set before and after sample bias reduction using vote entropy are shown in Tables 5.27 and 5.28, respectively.

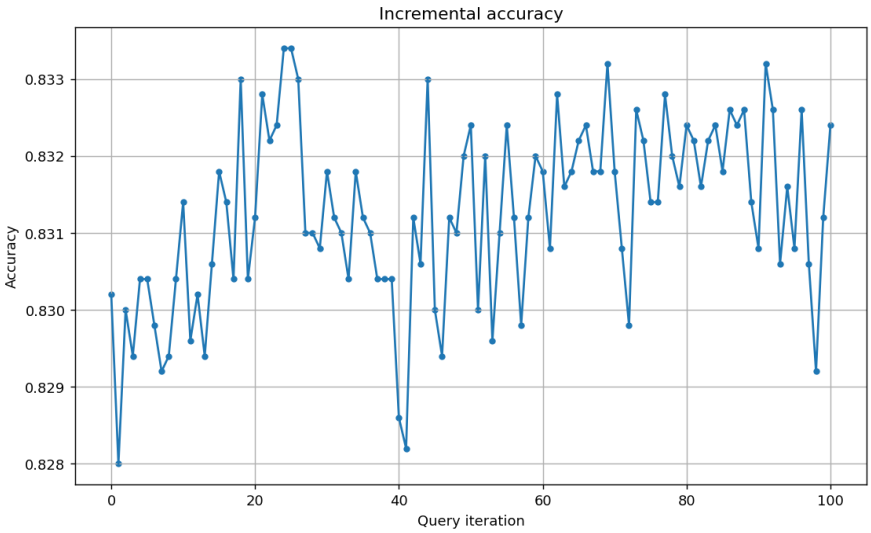

Figure 5.15: Predictive performance curve of the credit scoring model committee when reducing sample bias using the Vote Entropy sampling

- **KL divergence sampling**

As with vote entropy sampling, the process starts by determining the committee score, which is computed on the set of accepted applications from the credit scoring model committee built using the *"committee"* function including the list of learning credit scoring models and the *"max_disagreement_sampling"* as the query strategy. Then, the set of rejected applications is queried using the *".query(X,y)"* function to select the most informative application that will then be added to the process. Finally, the new committee accuracy is recorded, and the added application is removed from the pool of rejected applications. The process is then repeated until the stopping criteria reached (the number of queries). During the QBC process, we added the 100 most informative rejected applications to the process. Figure 5.16 below illustrates the committee score recorded with each addition of the most informative rejected application. Performance metric values of the credit scoring committee on the training and test set before and after sample bias reduction using KL divergence are summarized in Tables 5.27 and 5.28, respectively.

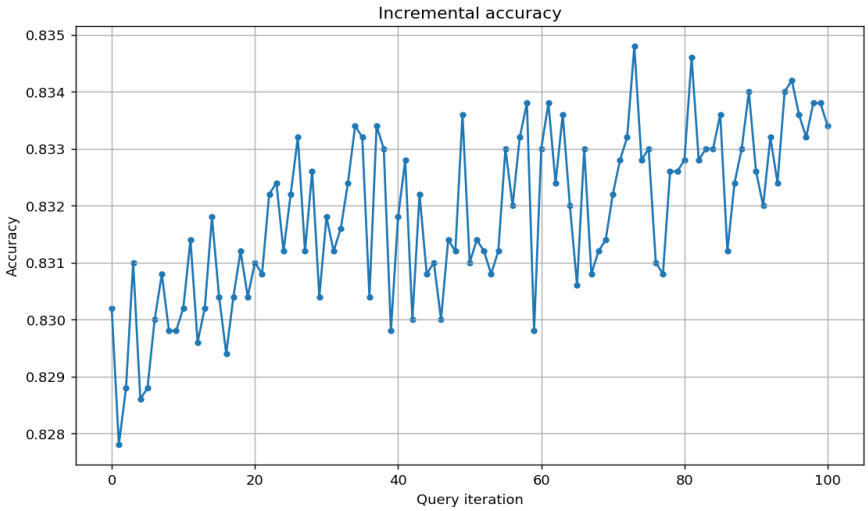

Figure 5.16: Predictive performance curve of the credit scoring model committee when reducing sample bias using the KL divergence sampling.

Approach	Training set	
	Accuracy	AUC
Committee (accepted applications only)	0.8302	0.7884
Vote Entropy sampling	0.8324	0.7895
KL Divergence sampling	0.8334	0.7924

Table 5.27: Performance of the credit scoring model committee when reducing sample bias using pool based QBC strategies.

The table above shows the performance of the credit scoring model committee before (based on accepted applications only) and after (using QBC query strategies) sample bias reduction. In addition, It shows that the predictive performance and AUC of the credit scoring model committee have increase when

sample bias is reduced by adding the most informative rejected applications to the training based on vote entropy sampling and KL divergence sampling. Moreover, the improvement in the performance metrics is stronger when using KL divergence sampling measure. This is illustrated in Figure 5.17 below

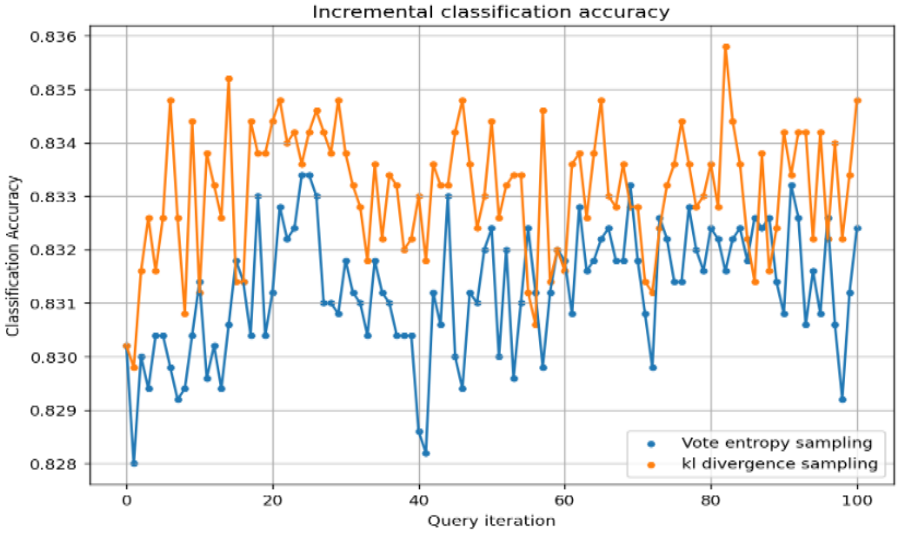

Figure 5.17: Predictive performance comparison of the credit scoring model committee when reducing sample bias using pool based QBC strategies.

The above figure shows that the predictive performance of the committee throughout the QBC process is higher with each addition of the most informative rejected application using KL divergence sampling than vote entropy sampling.

Approach	Test set	
	Accuracy	AUC
Committee (accepted applications only)	0.8181	0.7792
Vote Entropy sampling	0.8233	0.7805
KL Divergence sampling	0.8207	0.7809

Table 5.28: Performance of the credit scoring model committee on the test set when reducing sample bias using pool based QBC strategies

Table 5.28 shows that the performance of the credit scoring model committee on the test set increases when sample bias is reduced by using QBC query strategies. In conclusion, we can say that the pool-based QBC strategy reduces sample bias by adding the most informative rejected application, which improves the predictive performance of credit scoring models. Furthermore, the performance of credit scoring models can also be assessed from the confusion matrix. Table 5.29 shows the confusion matrix of the scoring model committee:

Approach	Confusion matrix			
	TP	FP	TN	FN

Committee	6138	868	2041	950
Vote Entropy sampling	6216	790	2015	976
KL Divergence sampling	6166	840	2039	952

Table 5.29: Confusion matrix of the credit scoring model committee using pool based QBC strategies.

Table 5.29 also shows an improvement in the prediction of the credit scoring model committee when reducing sample bias using the QBC query strategies. These strategies increase the number of correctly predicted values and decrease the number of wrongly predicted values by the committee. Thus, the increase in the predictive performance of the committee of credit scoring models decreases the rate of misclassification.

- **Query strategy: Density-Weighted method**

As with previous query strategies, the density-weighted strategy is used to query the most informative applications from the pool of rejected applications that will be then added into the training set. For this, the cluster-based sampling is used. First, the pool of rejected applications is divided into a large number of clusters (100 clusters) and sampled evenly from each cluster. To do this, the k-means clustering algorithm from the package *"sklearn.cluster"* will be used. This algorithm measures the distance from the center of a cluster. From there, the rejected application that is closest to the center of a cluster (known as the centroid) is thus added to the training set. Therefore, the applications that are farthest from the center of any cluster are outliers and are not considered in the active learning process. In the active learning process of each credit scoring model, the score is first calculated based on the set of accepted applications and then updated by adding the most informative rejected applications. It is calculated from the learner model, which is built using the "*ActiveLearner"* function including the estimator of the credit scoring model, the *"uncertainty_sampling*" as query strategy, and the set of accepted applications. Then, the learner model is used to add applications closest to centroids into the training set. The active learning process begins after defining the number of queries to be added into the training set, in order to update the score of the credit scoring model. Then, the *".teach(X,y)"* function (X represents the cluster center of the instance and y the label of this instance) is used to teach the learner model the instance it has queried. With each addition of the most informative rejected application, the model score is updated. We added the 100 most informative applications from the set of rejected applications in the active learning process of each credit scoring model. Figure 5.18 below illustrates the change in model accuracy at each addition. This figure shows that the predictive performance of all credit scoring models, except the NN and DT model, decreases with each addition of the most informative rejected application into the training set. Table 5.30 and 5.31

summarize the performance metric values of credit scoring models on the training set and test set when reducing sample bias using density-weighted:

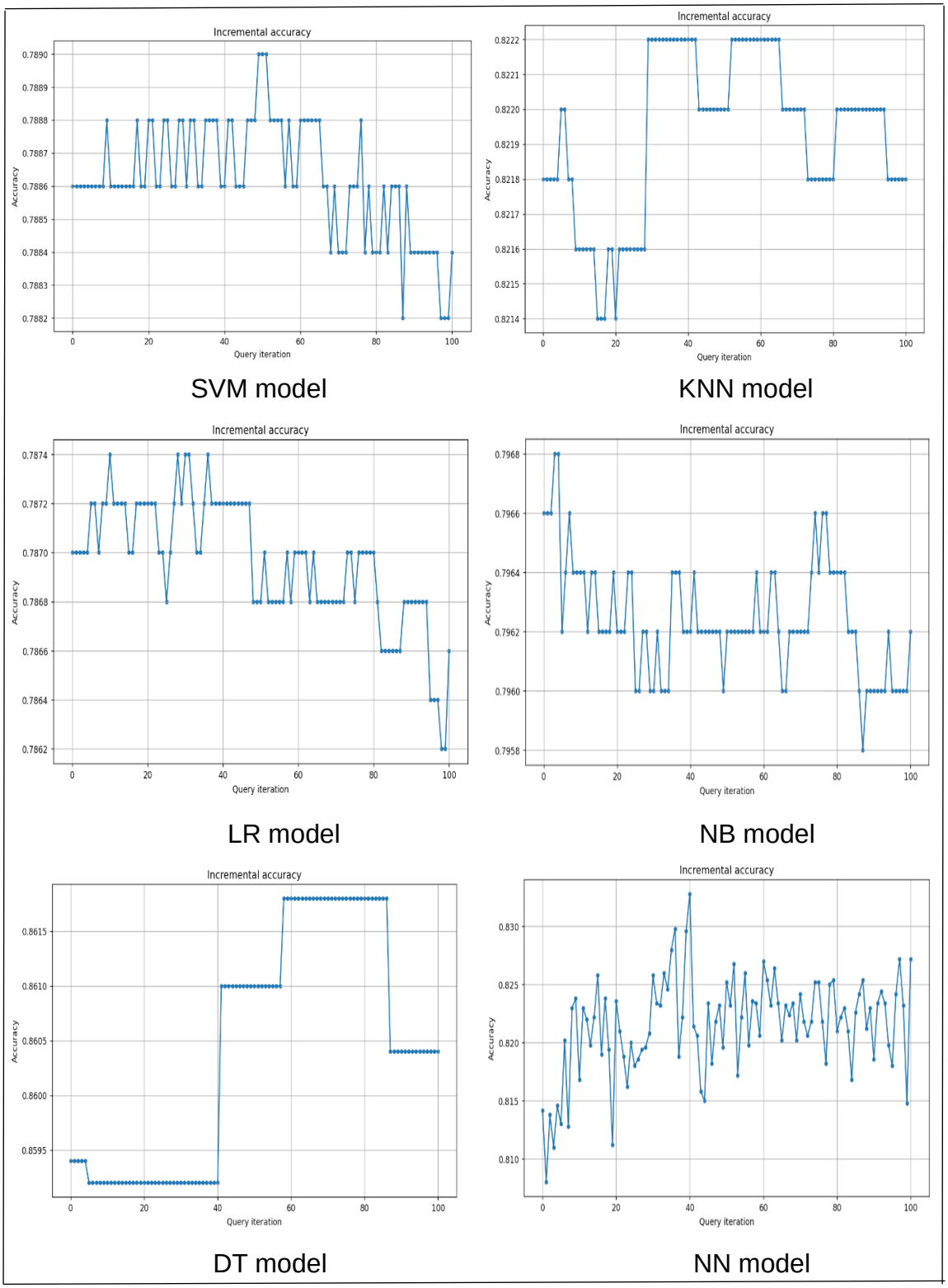

Figure 5.18: Accuracy curve of credit scoring models when reducing sample bias using density weighted method.

Model	Training set	
	Accuracy	AUC

LR	0.7866	0.7731
SVM	0.7884	0.7720
DT	0.8604	0.8355
KNN	0.8218	0.7805
NB	0.7962	0.7698
NN	0.8272	0.7989

Table 5.30: Performance of credit scoring models when reducing sample bias using the density-weighted approach.

Compared to Table 5.1, Table 5.30 shows a slight improvement in the predictive performance of some credit scoring models such as DT and NN when sample bias is reduced. However, reducing sample bias using the density-weighted strategy deteriorates the performance of the LR; SVM, KNN and NN models. This leads to decrease the performance of the credit scoring models on the set as shown in Table 5.31:

Model	Test set	
	Accuracy	AUC
LR	0.7892	0.7741
SVM	0.7932	0.7751
DT	0.8003	0.7674
KNN	0.8154	0.7697
NB	0.7998	0.7743
NN	0.8232	0.7911

Table 5.31: Performance of credit scoring models on the test set when reducing sample bias using the density-weighted approach.

Increasing the predictive performance of NN and DT when reducing sample bias by adding the most informative rejected applications using density-weighted improves their performance on the test set as illustrated in Table 5.31 (Compared to Table 5.2). The performance of the other credit scoring models decreases slightly as their predictive performance decreased during training. Furthermore, the performance of the credit scoring model can also be evaluated from the confusion matrix presented in Table 5.32:

Models	Confusion matrix			
	TP	FP	TN	FN
LR	5687	1319	2203	788
KNN	6190	816	1029	1962
SVC	5747	1259	2183	808
DT	5951	1055	2050	941
NB	5870	1136	2126	865
NN	6101	905	2128	863

Table 5.32: Confusion matrix of credit scoring models when reducing sample bias using density-weighted approach

Compared to table 5.3, the table above shows a decreasing number of TP and TN, and an increasing number of FP and FN in the LR, SVC and NB models when sample bias is reduced. This is due to the decrease in their predictive performance as shown in Table 5.30. On the other hand, it shows an improvement of these parameters in the DT and NN models.

6 Discussion

Sample bias is one of the most important problems in credit scoring models. This occurs when models are trained only on a set of accepted applications, while data from rejected applications are discarded. There are some techniques that can be used to solve this problem. Active learning is conceived as a suitable method, which was implemented in this thesis. This technique explores the set of rejected applications in order to select the best ones able to increase the performance of credit scoring models. Using query strategies, the most informative applications from a set of rejected applications are queried and added into the training set to reduce sample bias, thus leading to improve the predictive performance of credit scoring models. Since the repayment behavior of rejected applications cannot be observed, a rejected application, which should nevertheless be granted, is added to the training in order to observe the repayment behavior that increases the accuracy of the models. When the repayment behavior of the rejected application is considered to be non-default, the predictive performance of credit scoring models increases while it decreases when it is considered to be default. Therefore, the repayment behavior of rejected applications added into the training was considered non-default. We first calculated the predictive performance of credit scoring models based on the set on accepted applications only. Then, we updated it by adding the most informative rejected applications. The selection of the most informative rejected applications is performed from query strategies, such as Uncertainty sampling, QBC, which use various metrics to query an application. These query strategies allow the credit scoring models to learn from the features of accepted applications, identify rejected applications to be included into the training set, so that their predictive performance can improve. By implementing active learning strategies, it becomes clear that reducing sample bias by adding the most informative rejected applications slightly improves the predictive performance of the credit scoring models. This was also the same conclusion from other studies using other approaches for solving sample bias in credit scoring models. For example, Banasik & Crook (2007) have implemented different reject inference methods on different data sets in their study to solve the sample bias problem and came to the conclusion that these methods slightly increase the performance of credit scoring models. According to the results query strategies in active learning, we cannot say that one query strategy is better than the other because the gaps between them are very small. However, the density-weighted approach is the strategy that does not improve the credit scoring models significantly. This query strategy improves the performance of a limited number

of models but deteriorates that of a large number of models (as shown in Table 5.31 and 5.32). Nevertheless, the improvement in the performance of credit scoring models is higher in the pool-based active learning than in the stream based active learning. Moreover, we predicted the creditworthiness of applicants in the test set using credit scoring models before (trained from the set of accepted applications only) and after (trained from the set of applications including accepted and best ones rejected applications using active learning) sample bias reduction and compared them using performance metrics. So, we came to the conclusion that the performance of credit scoring models is slightly higher when sample bias is reduced. Therefore, we can say that using active learning strategies to include the most informative rejected applications in order to reduce sample bias improves the predictive performance of credit scoring models. One of the main benefits of active learning is that it reduces the treatment cost of the applications because it works with a small amount of data from which it learns. Furthermore, it reduces the risk of granting loans to risky applicants as it is based on selecting and adding the best rejected applications to the training, thus improving the prediction of the creditworthiness of the applicants. However, active learning is time-consuming because of the number of queries to be added in the training and takes more computer power (computational budget). The more the rejected applications are added to the training, the longer active learning takes. In addition, it will not completely eliminate sample bias because rejected applications with less or no information are not included in the process. This was also the same drawback of the external performance approach that was applied by Barakova, Glennon and Palvia (2013) to solve the sample bias problem in credit scoring models.

As illustrated in the related work section, there are various other techniques used to solve the sample bias problem such as the BVP (Banasic, Crook & Thomas, 2003), and the reject inference method (Banasik & Crook, 2007, Verstraeten & Poel, 2005). Like active learning, these techniques are also based on the analysis of rejected observations and the use of some sample selection techniques to highlight the most significant observations. Unlike active learning, in which a learning process is used to observe how the repayment behavior of selected rejected applications may influence the predictive performance, the BVP method mainly investigates the influence of missing variables that occurs when rejected applications are not taken into account in the building of scoring models (Banasic, Crook & Thomas, 2003). Furthermore, it is based on the build of the single model which is implemented on the set of all applicants in order to differentiate good applicants from bad. Active learning, on the other hand, queries the set of rejected applications in order to select the most informative ones that will then be added into the training set to reduce sample bias. However, the predictive performance of the models increases slightly using the BVP approach. The same is true for the active learning method. In addition, with a small training dataset, which reduces working time, the active learning improves the performance of credit scoring models, while the BVP is applied on the whole dataset, which requires a

lot of training time. Regarding the reject inference, there are many procedures such as reweighting, extrapolation, augmentation, sample selection using to solving sample bias in credit scoring models. In their study, Banasik and Crook (2007) used the sample selection model and the sampling weights technique (augmentation) together, to construct the reject inference method for improving the predictive performance of the models, while the model in active learning is built from the estimation of credit scoring model and the query strategy. Furthermore, the difference between reject inference and active learning is that active learning explore the set of rejected applications to query the best ones, whereas reject inference techniques try to incorporate the characteristics of rejected applicants into the model based on the repayment behavior of accepted applicants, so that the final model can be estimated using the set of all applicants (Crook & Banasic, 2004). However, one came to the same conclusion that both methods slightly improve the performance of credit scoring. Just like active learning, the study by Barakova, Glennon and Palvia (2013) on the use of the proxy payment performance for rejected applications to reduce the sample bias is based on selecting of the most informative rejected applications and also cannot fully eliminate the problem of sampling bias. Nevertheless, the active learning approach is less expensive because of the size of the data needed, while the proxy performance approach is very costly because of obtaining the bureau data. In the study by Kiefer and Larsen (2006), based on the use of the reject inference technique which consists of assigning the default class to rejected applications, it was shown that the performance of the credit scoring model decreases. In the active learning process, the non-default class was assigned to the rejected applications, which improves the performance of credit scoring models.

7 Conclusion

In recent years, the credit scoring method has been widely used by banking institutions for credit applications in order to identify applicants with high credit risks. The decision of banks to approve or reject the loan application is mainly based on the credit score, which can be determined from several statistical models. The credit score is a numerical expression that represents the creditworthiness of the borrower. In this study, we implemented some credit scoring models including the RF model, the KNN model, the DT model, the GM model, the LR model, the NN model, and the SVM Model. These models are trained solely on the set of accepted applications, thus creating the sample bias problem, which deteriorates their performance. The sample of rejected applications is not considered. Therefore, the sample for training credit scoring models is not representative of the whole population of applicants. There are many works proposed by different authors to solve the sample bias in credit scoring models. This thesis implemented the active learning technique. This technique relies on the selection of the most informative applications from the

pool of rejected applications, which can improve the predictive performance of credit scoring models. In this approach, two scenarios were considered, namely the pool-based active learning and the stream-based active learning scenario. In each scenario, various query strategies, including uncertainty sampling (with measures namely entropy sampling, least confident sampling and margin sampling in pool-based active learning scenario; classifier entropy, classifier uncertainty and classifier margin in stream-based active earning scenario),the Query By Committee (with vote entropy and KL divergence in stream based active learning; vote entropy sampling and divergence sampling in pool based active learning), the density weighted method, have been implemented. The dataset used in this paper is the Mortgage dataset provided by IFR. The first step was to explore and analyze the dataset using some feature selection techniques. These techniques can clean up the dataset and select the most important variables, which were then used for modelling of credit scoring models. Then, the dataset was divided into a training and a test data. The training data consists of two sets namely the set of accepted applications and the set of rejected applications. After that, the credit scoring models were trained only on a set of accepted applications. Finally, active learning was implemented to explore the pool of rejected applications in order to select the most informative rejected applications based on query strategies, which will then be added to the process to reduce sample bias in the training set. The results of this approach showed a slight improvement in the performance of credit scoring models. Therefore, granting loans to risky applicants is costly, but ensures performance improvements of credit scoring models due to gathering more representative training data. The credit scoring model when reducing sample bias is used to predict the creditworthiness (default or not) of applicants in the test set. Model evaluation showed an increase in the performance of credit scoring models on the test data when sample is reduced. Therefore, adding the most informative rejected applications into the training enhances the decision on whether to grant credit or not.

This study has shown that active learning can perform as a method for solving sample bias that arises using the credit scoring models quite well, despite it was designed to label a large set of unlabeled instances. Also, we showed that this method can improve the predictive performance of credit scoring models on the basis of deploying information about rejected applicants. However, the density weighted strategy does not bring an improvement for all credit scoring models. One reason may be the selection of the training data. That can be overcome with another study about clustering the data. Further work can focus on the use of some clustering methods for initial sample selection. One could also expand the study to include the QUIRE (QUerying Informative and Representative Examples) approach, which is based on the min-max view of active learning (Huang, Jin & Zhou, 2014). This approach focuses on the selection of unlabeled instances that are not only informative but also representative. Furthermore, the active learning approach could also be extended to the instance correlation.

Active learning-based instance correlation is used to address information redundancy and consists of many approaches such as the exploration on feature correlation, the exploration on feature and label correlation, the exploration on label correlation, the exploration on graph structure correlation (Fu, Zhu & Li, 2012).

References

Abdou, H.A. (2009). Genetic programming for credit scoring: The case of Egyptian public sector banks. *Expert Systems with Applications,* 1(Nov), 11402-11417.

Abdou, H., & Pointon, J. (2011). Credit scoring, statistical techniques, and evaluation criteria: A review of the literature. *Intelligent Systems in Accounting Finance & Management*, 22(Jun), 59-88.

Akindaini, B. (2017). Machine learning applications in mortgage default prediction. *University of Tampere*, 24(Nov), 1-55.

Al Amari, A. (2002). The credit evaluation process and the role of credit scoring: A case study of Qatar. *Doctoral dissertation, University College Dublin*.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. The *American Statistician, 1(Aug), 175-185.*

Argamon-Engleson, S., & Dagan, I. (1999). Committee-based Sample Selection for Probabilistic Classifiers. *Journal of Artificial Intelligence Research*, 1(Jul), 335-360.

Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J., & Vanthienen, J. (2003). Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the operational research society*, 9(Jun), 627-635.

Baesens, B., Roesch, D., & Scheule, H. (2016). Credit Risk Analytics: Measurement Techniques. *Applications and Examples in SAS*, *Wile*y.

Bagherpour, A.R. (2017). Predicting Mortgage Loan Default with Machine Learning Methods. *University of California/Riverside,* 1(Jan), 1-29*.*

Banasik, J., & Crook, J. (2007). Reject Inference, Augmentation, and Sample Selection. *European Journal of Operational Research*, 16(Dec), 1582-1594.

Banasik, J., Crook, J., & Thomas, L. C. (2003). Sample selection bias in credit scoring models. *Journal of the Operational Research Society,* 1(Mar), 822-832.

Barakova, I., Glennon, D., & Palvia, A. A (2013). Sample Selection Bias in Acquisition Credit Scoring Models: An Evaluation of the Supplemental-Data Approach. *Journal of Credit Risk*, 1(Apr), 77-117*.*

Beygelzimer, A., Dasgupa, S., & Langford, J. (2009). Important weighted active learning. In *Proceedings of the 26th international conference on machine learning (ICML)*, 14(Jun), 49–56.

Blöchlinger, A., & Leippold, M. (2011). A New Goodness-of-Fit Test for Event Foracasting and Its Application to Credit Defaults. *Management Science*, 1(Mar), 487-505.

Boyes, W. J., Hoffman D. L., & Low S. A. (1989). An econometric analysis of the bank credit scoring problem. *Journal of Econometric perspectives,* 1(Jan), 3–14.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and Regression Trees. *CRC press*, 1(Sep), 1-8.

Breiman, L. (2001). Random forests*. Machine Learning,* 1(Oct), 5-32.

Burbidge, R., Rowland, J. J., & King, R. D. (2007). Active learning for regression based on query by committee. In *Proceedings of Intelligent Data Engineering and Automated Learning (IDEAL)*, 16(Dec), 209-218.

Cieslak, D. A., & Chawla, N. V. (2008). Learning decision trees for unbalanced data. *Machine Learning and Knowledge in Databases. ECML/PKDD,* 14(Sep), 241-256.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, 15(May):201-221.

Cohn, D., Ghahramani, Z., & Jordan, M. I. (1996). Active learning with statistical models*. Journal of Artificial Intelligence Research*, 1(Mar), 129-145.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(Feb), 273-297.

Cox, D. R. (1958). The Regression Analysis of Binary Sequences. *Journal of the Royal Statistical Society*, Series B (Methodological), 1(Jul), 215-232.

Crook, J., & Banasik, J. (2004). Does reject inference really improve the performance of application scoring models? *Journal of Banking and Finance*, 1(Apr), 857-874.

Crook, J. N., Edelman, D. B., & Thomas, L. C. (2007): Recent developments in consumer credit risk assessment. *European Journal of Operational Research*, 16(Dec), 1447-1465.

Culotta, A., & McCallum, A. (2005). Reducing labeling effort for structured prediction tasks. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 9(Jul), 746-751.

Dagan, I., & Engelson, S. P. (1995). Committee-based sampling for training probabilistic classifiers. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML)*, 9-12(Jul), 150-157.

Domingos, P., & Michael, P. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 1(Nov), 103-130.

Einav, L., Jenkins, M., & Levin, J. (2013). The impact of credit scoring on consumer lending. *The RAND Journal of Economics*, 18(Jun), 249-274.

Fan, M., Yang, S., & Zhang, J. (2008). Credit Rating Models Considering Sample Selection Biases. *International Seminar on Business and Information Management*, 1(Dec), 433-436.

Fawcett, T. (2006). An introduction to ROC analysis .*Pattern Recognition Letters,* 1(Jun), 861-874.

Fix, E., & Hodges, J.L. (1952). Discriminatory Analysis. Nonparametric Discrimination: small sample performance. *USAF School of Aviation Medicine*, 1(Aug), 280-322.

Fix, E., & Hodges Jr, J. L. (1989). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties. *International Statistical Review,* 1(Dec), 238-247 .

Freund, R. J., & William, W. J. (1998). Regression analysis: Statistical modeling of a response variable. *San Diego: Academic Press*.

Fröhlich, H., & Chapelle, O. (2003). Feature selection for support vector machines by means of genetic algorithms. In *Proceedings of the 15th IEEE international conference on tools with artificial intelligence*, 5(Nov), 142–148.

Fu, Y., Zhu, X., & Li, B. (2012). A survey on instance selection for active learning. *Knowledge and Information Systems,* 6(Jun), 249-283.

Ghasemi, A., Rabiee, H. R., Fadaee, M., Manzuri, M. T., & Rohban, M. H. (2011). Active Learning from Positive and Unlabeled Data. *IEEE 11th International Conference on Data Mining Workshops,* 11(Dec), 244-250.

Gong, R., & Huang, S. H. (2012). A Kolmogorov-Smirnov statistic-based segmentation approach to learning from imbalanced datasets: With application in property refinance prediction. *Expert Systems with Applications*, 1(May), 6192-6200.

Greene, W. (1998). Sample selection in credit-scoring models. *Japan and the World Economy*, 1(Jul), 299-316.

Guyon, I., & Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research (JMLR),* 3(Mar), 1157-1182.

Hand, D. J. (2005). Good practice in retail credit scorecard assessment. *Journal of the Operational Research Society*, 2(Feb), 1109-1117.

Hand, D.J., & Henley, W.E., (1993/4). Can reject inference ever work? *IMA Journal of Management Mathematics*, 1(Jan), 45–55.

Hand, D.J., & Henley, W.E., (1994). Inference about rejected cases in discriminant analysis. *New Approaches in Classification and Data Analysis: Springer-Verlag, Berlin*, 1(Jul), 292–299.

Hand, D. J., Henley, & W. E. (1997). Statistical Classification Methods in Consumer Credit Scoring: A Review. *Journal of the Royal Statistical Society*: *Series A (Statistics in Society),* 1(Jan), 523-541.

Hand, D. J., & Jacka, S. D. (1998). Statistics in Finance (Arnold Applications of Statistics Series), *Wiley*, 4(Mar).

Heckman, J. (1979). Sample selection bias as a specification error. *Econometrica,* 1(Jan), 153–161.

Hernandez-Orallo, J., Flach, P., & Ferri, C. (2011). Brier curves: a new cost-based visualisation of classifier performance. In *Proceedings of the 28th International Conference on Machine Learning (ICML), 28(Jun), 585-592.*

Henley, W. E. (1995). Statistical aspects of credit scoring. *Ph.D. Thesis: The Open University, Milton Keynes*.

Hsai D. C. (1978). Credit scoring and the Equal Credit Opportunity Act. *Hastings Law Journal,* 1(Jan), 371–448.

Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2010). A practical guide to support vector classification. *BJU International*, 15(Apr), 396-400.

Huang, C., Chen, M., & Wang, C. (2007). Credit scoring with a data mining approach based on support vector machines, *Expert Systems with Applications,* 1(Nov), 847-856.

Huang, S. -J., Jin, R., & Zhou, Z. -H. (2014). Active Learning by Querying Informative and Representative Examples. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(Oct), 1936-1949.

Iihan, H. O., & Amasyali, M. F. (2014). Active learning as a way of increasing accuracy. *International Journal of Computer Theory and Engineering*, 6(Dec), 460-465.

Jacobson, T., & Roszbach, K. (2003). Bank lending policy, credit scoring and value-at-risk. *Journal of Banking & Finance*, 1(Apr), 615-633.

Joanes, D. N. (1993/4). Reject inference applied to logistic regression for credit scoring. *IMA Journal of Management Mathematics*, 1(Jan), 35– 43.

Kiefer, N. M., & Larson, C. E. (2006). Specification and Informational Issues in Credit Scoring. *Available at SSRN 956628*, 1(Oct), 2-29.

Kim, Y. S., & Sohn, S. Y. (2004). Managing Loan Customers Using Misclassification Patterns of Credit Scoring Model. *Expert Systems with Applications,* 1(May), 567-573.

King, G., & Zeng, L. (2001). Logistic Regression in Rare Events Data. *Political Analysis*, 16(Feb), 137-163.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. The *Annals of Mathematical Statistics*, 1(Mar), 79-86.

Lessmann, S., Baesens, B., Seow, H. V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, 16(Nov), 124-136.

Lewis, D. D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 10(Jul), 148-156.

Lewis, D. D., & Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *SIGIR'94,* 24(Jul), 3–12.

Lim, M. K., & Sohn, S. Y. (2007). Cluster-Based Dynamic Scoring Model. *Expert Systems with Applications,* 1(Feb), 427-431.

Loh, W. Y. (2014). Fifty years of classification and regression trees. *International Statistical Review*, 30(Jun), 329-348.

McCallum, A., & Nigam, K. (1998). Employing EM in pool-based active learning for text classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, 24(Jul), 359-367.

Meng, C.L., & Schmidt, P. (1985). On the cost of partial observability in the bivariate probit model. *International Economic Review*, 1(Feb), 71–85.

Mislevy, R. J. (1991). Book Reviews: Statistical Analysis With Missing Data: Roderick JA Little and Donald B. Rubin New York: John Wiley & Sons, 1987. xiv+ 278 pp. *Journal of Educational Statistics*, 1(Jun), 150-155.

Okasha, M. K. (2014). Using Support Vector Machines in Financial Time Series. *International Journal of Statistics and Applications*, 1(Avr), 28-39.

Parnitzke, T. (2005). Credit Scoring and the sample selection bias. *Institute of Insurance Economics, University of St. Gallen (mimeo)*, 31(May), 1-20.

Pundir, S., & Seshadri, R. (2012). A Novel Concept of Partial Lorenz Curve and Partial Gini Index. *International Journal of Engineering, Science, and Innovative Technology,* 1(Jan), 296-301.

Rish, I. (2001). An empirical study of the naïve Bayes classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence,* 1(Jan)*,* 41-46.

Rokach, L., & Maimon, O. (2008). Data Mining with Decision Trees: Theory and Applications. *World Scientific Publishing Company*, 1(Jan), 264.

Salzberg, S. L. (1994). C4.5: Programs for machine learning by J:Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993. *Machine Learning*, 1(Sep), 235-240.

Scheffer, T., Decomain, C., & Wrobel, S. (2001). Active hidden Markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, 3(Sep), 309-318.

Settles, B. (2010). Active Learning Literature Survey. *University of Wisconsin, Madison*, 26(Jan), 127-131.

Settles, B. (2008). Curious Machines: Active Learning with Structured Instances. *Doctoral dissertation, University of Wisconsin-Madison.*

Settles, B., & Craven, M. (1998). An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Settles, B., & Craven, M. (2008). An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 25(Oct), 1070-1079.

Seung, H. S., Opper, M., & Sompolinsky, H. (1992). Query By Committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, 1(Jul), 287-294.

Schreiner, M. (2003). Scoring: The Next Breakthrough in Microcredit? *Occasional Paper*, 1(Jan), 8-75.

Shannon, C.E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 1(Jul), 379-423.

Sun, L. L., & Wang, X. Z. (2010). A survey on Active Learning Strategy. *In Proceedings of the Ninth International Conference on Machine Learning and Cybernetics (ICMLC)*, 11-14(Jul), 161-166.

Tan, Z., Yan, Z.; & Zhu, G. (2019). Stock Selection with Random Forest: An exploitation of excess return in the Chinese stock market. *Heliyon*, 12(Aug), e02310.

Tang, M., Luo, X., & Roukos, S. (2002). Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 6(Jul);120-127.

Thomas, L, Crook, J., & Edelman, D. (2017). Credit Scoring and Its Applications. *Society for industrial and Applied Mathematics (SIAM), 17(Aug)*.

Thomson, C. A., Califf, M. E., & Mooney, R. J. (1999). Active learning for natural language parsing and information extraction. *In proceedings of the International Conference on Machine Learning (ICML)*, 5(Feb), 406-414.

Tong, S., & Koller, D. (2002). Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2(Nov), 45-66.

Verstraeten, G., & Van den Poel, D. (2005). The impact of sample bias on consumer credit scoring performance and profitability. *Journal of the Operational Research Society*, 22(Dec), 981-992.

West, D. (2000). Neural Network Credit Scoring Models. *Computers & Operations Research,* 1(Oct), 1131-1152.

Zekic-Susac, M., Sarlija, N., & Bensic, M. (2004). Small Business Credit Scoring: A Comparison of Logistic Regression, Neural Networks, and Decision Tree Models. *26th International Conference on Information Technology Interfaces, 7-10(Jun)*, 265-270.

Zhao, Y., Cao, Y. C., Pan, X. Q., Lu, Y.,& Xu, X. N. (2008). A telecom clients' credit risk rating model based on active learning. *In Proceedings of IEEE international Conference on Automation and Logistics*, 30(Sep), 2590-2593.

Zhang, G, Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks: the state of the art. *International Journal of Forecasting,* 1(Mar), 35-62.

Zhang, Y., & Bhattacharyya, S. (2004). Genetic Programming in classifying large-scale data: an ensemble method. *Information Sciences,* 14(Jun), 85-101.

Zhou, Z.-H. (2012). Ensemble Methods: Foundations and Algorithms. *Chapman and Hall/CRC* , 6(Jun), 459-485.

Zhu, J., Wang, H., Tsou, B. K., & Ma, M. (2010). Active Learning with sampling by uncertainty and density for data annotations. *IEEE Transactions on audio, speech, and language processing*, 18(Jun), 1323-1331.

Zhu, X., & Wu, X. (2006). Scalable representative instance selection and ranking. *In Proceedings of the 18th international conference on pattern recognition*, 20(Aug), 352–355.

Appendix

A1 Feature Description

	Variable	Description
1	Id	It is an integer variable which takes the value from 0 to 50000 and it refers to the id of the borrower.
2	time	It is an integer variable which takes the value from 0 to 50000 and it refers to the id of the borrower.
3	orig_time	It is an integer variable that indicates the time of origination of the loan, i.e., the time period at which the loan was taken. It takes values from -40 to 60.
4	first_time	It is an integer variable that shows the first observation period, i.e., it is the time period at which the loan was observed for the first time. It takes values from 0 to 60.
5	mat_time	It corresponds to the time stamp for maturity. It is an integer variable ranging from 18 to 229 and gives information about the time period in which the loan will mature
6	balance_time	It is a numerical variable which shows the amount of the loan that still needs to be repaid at the observation time, i.e., it shows the repayment amount due at the observation period. Its value ranges from 0 to 2730000.
7	LTV_time	It is a numerical variable bearing a value from 0% to 166.29%. It represents the loan-to-value ratio at the observation time, i.e., it expresses the loan amount as a percentage of the value of the house at the observation period.
8	interest_rate_time	It is a numerical variable ranging from 0% to 37.5%. It shows the proportion of the residential loan that has been charged as interest to the borrower at the observation period.
9	hpi_time	It is the House price index at observation time. It is a numerical variable bearing a value from 107.83 to 226.29. This index measures the price changes

		in residential housing, here, at the observation period.
10	gdp_time	Represents Gross domestic product growth at observation time. It is a numerical variable which illustrates how far the economy is expanding at the observation time. Generally, GDP growth is used to determine the health of a country's economy. Typically, it is calculated annually or quarterly and in some countries such as the US, even monthly. In our study, this GDP growth is a monthly calculation. It ranges between -4.15% to 5.13%.
11	uer_time	It is a numerical variable indicating the unemployment rate of the economy at the observation period. In other words, it reflects the percentage of unemployed persons in the total labour force. The rates of unemployment in this study ranges between 3.8% to 10%.
12	REtype_CO_orig_time	It is an integer binary variable considered at the time of origination. It takes the value of 1 if it is a complex with individually owned houses and the value of 0 otherwise
13	REtype_PU_orig_time	Corresponds to the real estate type planed urban. It is an integer binary variable considered at the origination time, which takes the value of 1 if the real estate is one which contains residential and commercial buildings along with an open space and the value of 0 otherwise.
14	REtype_SF_orig_time	It represents the real estate type single-family home. It is an integer binary variable considered at the origination time, taking the values of 1 if it is a stand-alone house designed for solely one family and 0 otherwise, respectively.
15	investor_orig_time	It is an integer binary variable considered at origination time, which takes the value of 1 if the investor is a first-time borrower and the value of 0 otherwise.
16	balance_orig_time	It is a numerical variable showing the loan amount at the origination time. In other words, it corresponds to the initial amount borrowed. It varies between the amounts 0 to 6500000

17	FICO_orig_time	It is an integer variable ranging from 400 to 840. It shows the FICO score of the borrower at the time the loan was originated. The FICO score is one which has been created by Fair Isaac Corporation and typically, it often ranged between 300 to 850. It is a commonly used score in the assessment of credit risk and in the decision process of whether to extend further credit to an individual. This score in itself is essentially made up from 5 data sources of the borrower namely the payment history, the amount of current debt, the length of credit history, the types of credit in use and the pursuit of new credit.
18	LTV_orig_time	It represents the loan-to-value ratio at the origination time, i.e., the loan amount expressed as a percentage of the value of the house at origination time. It is a numerical variable ranging from 50.1% to 218.5%.
19	Interest_Rate_orig_time	It is a numerical variable which shows the proportion of the interest charged from the loan amount, at the time of origination. It varies between 0% to 19.75%.
20	hpi_orig_time	It is a numerical variable ranging from 75.71 to 226.29. It shows the price changes in residential housing, at the origination time.
21	payoff_time	It is an integer binary variable which takes the value of 1 to indicate that the borrower has cleared the loan repayments at the observation period and the value of 0 otherwise.
22	default_time	It is an integer binary variable which takes the value of 1 to indicates that the borrower has defaulted at the observation period and the value of 0 otherwise
23	status_time	It is an integer variable which takes the value of 0 to indicate that the borrower has neither defaulted nor paid off the loan at the observation period, the value of 1 to indicated that he has defaulted at the observation period and the value of 2 to indicate that he has paid off the loan at the observation period.

Table A-1: Data description

Source: Measurement Techniques, Applications and Examples in SAS, Wiley (Baesens, Roesch & Scheule, 2016)

A2 Feature Selection

A2.1 Missing observations

Variables	Missing values
id	0
time	0
orig_time	0
first_time	0
mat_time	0
balance_time	0
LTV_time	18
interest_rate_time	0
hpi_time	0
gdp_time	0
uer_time	0
REtype_CO_orig_time	0
REtype_PU_orig_time	0
REtype_SF_orig_time	0
investor _orig_time	0
balance_orig_time	0
FICO_orig_time	0
LTV_orig_time	0
Interest_Rate_orig_time	0
hpi_orig_time	0
default_time	0
payoff_time	0
status_time	0

Table A-2: Missing observations in the variables

A2.2 Correlation

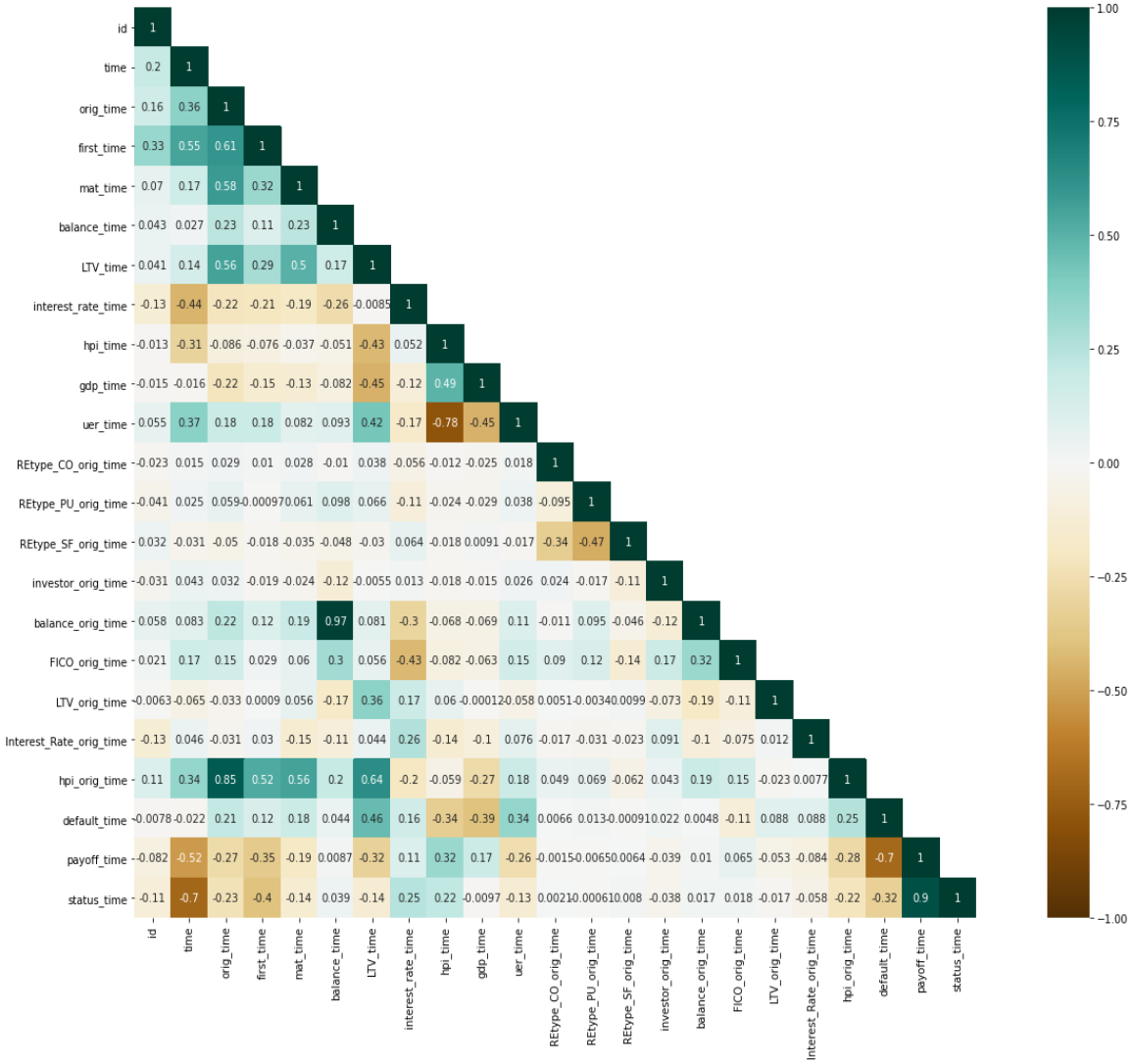

Figure A-1: Correlation between the variables

Variables	Pearson correlation	P-value
id	-0.01	0.08
time	-0.02	0
orig_time	0.21	0
first_time	0.12	0
mat_time	0.18	0
balance_time	0.04	0
LTV_time	0.46	0
interest_rate_time	0.16	0
hpi_time	-0.34	0
gdp_time	-0.39	0
uer_time	0.34	0
REtype_CO_orig_time	0.01	0.14

E

REtype_PU_orig_time	0.01	0
REtype_SF_orig_time	0	0.85
investor _orig_time	0.02	0
balance_orig_time	0	0.29
FICO_orig_time	-0.11	0
LTV_orig_time	0.09	0
Interest_Rate_orig_time	0.09	0
hpi_orig_time	0.25	0

Table A-3: P-Value table.

A2.3 Outlier detection

Variable	Outliers rate
time	0%
orig_time	7.5%
first_time	7.7%
mat_time	13.4%
balance_time	8.6%
LTV_time	5.8%
interest_rate_time	13.1%
hpi_time	0%
gdp_time	13.9%
uer_time	15.6%
REtype_PU_orig_time	11.6%
Investor_orig_time	11.8%
FICO_orig_time	1.5%
LTV_orig_time	13.1%
Interest_Rate_orig_time	0.2%
hpi_orig_time	10.2%

Table A-4: Outlier rate in the variables

A3 Statistical description of variables

A3.1 Description of variables

	count	mean	std	min	25%	50%	75%	max
time	49982	36.17	14	1	26	32	46	60
Orig_time	49982	20.4	7.877	-40	17	22	25	60
First_time	49982	24.7	7.9	1	20	25	28	60
Mat_time	49982	137.9	17.9	18	136	141	145	229
Balance_time	49982	2.4e5	2e5	0	1e5	2e5	3e5	8e6
LTV_time	49982	79.31	25.13	0	64.1	78.7	95.7	61.7

Interest_rate_time	49982	7.01	2.21	0	5.88	7	8.35	37
Hpi_time	49982	190	26	107	162	190	217	226
Gdp_time	49982	1.7	1.92	-4	1.23	2.27	2.84	5.13
Uer_time	49982	6	1.7	3.8	4.7	5.5	6.5	10
REtype_PU_orig_time	49982	0.12	0.320	0	0	0	0	1
Investor_orig_time	49982	0.12	0.323	0	0	0	0	1
FICO_orig_time	49982	661.3	72.7	400	612	663	715	84
LTV_orig_time	49982	79.7	9.9	50	75	80	85	21.9
Interest_Rate_orig_time	49982	5.44	3.29	0	3	6.38	7.62	19.8
Hpi_orig_time	49982	196.3	34.33	75	179	213	222	227
Default_time	49982	0.303	0.46	0	0	0	1	1

Table A-5: Descriptive statistics for variables:

A3.2 Correlation between the significant variables

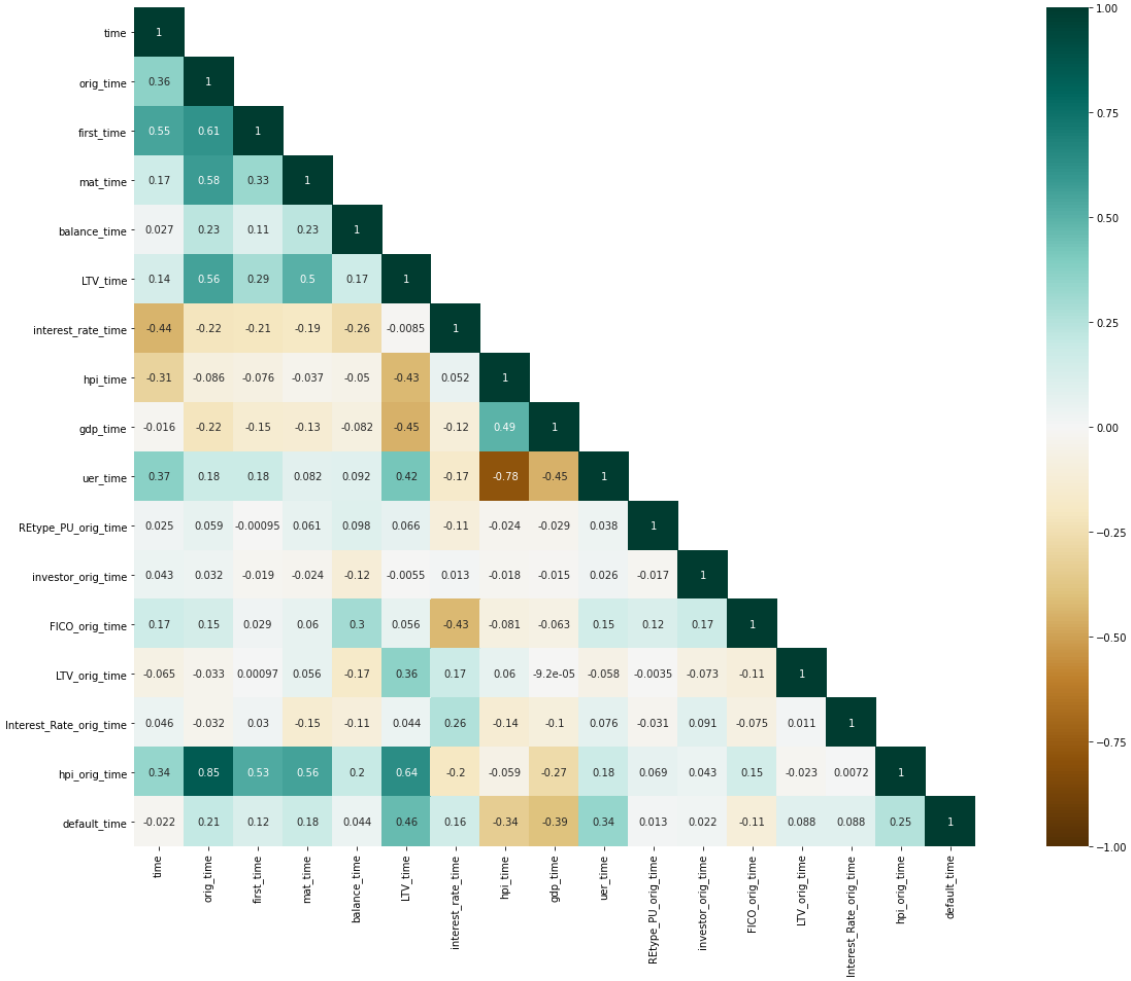

Figure A-2: Correlation between the most significant variables