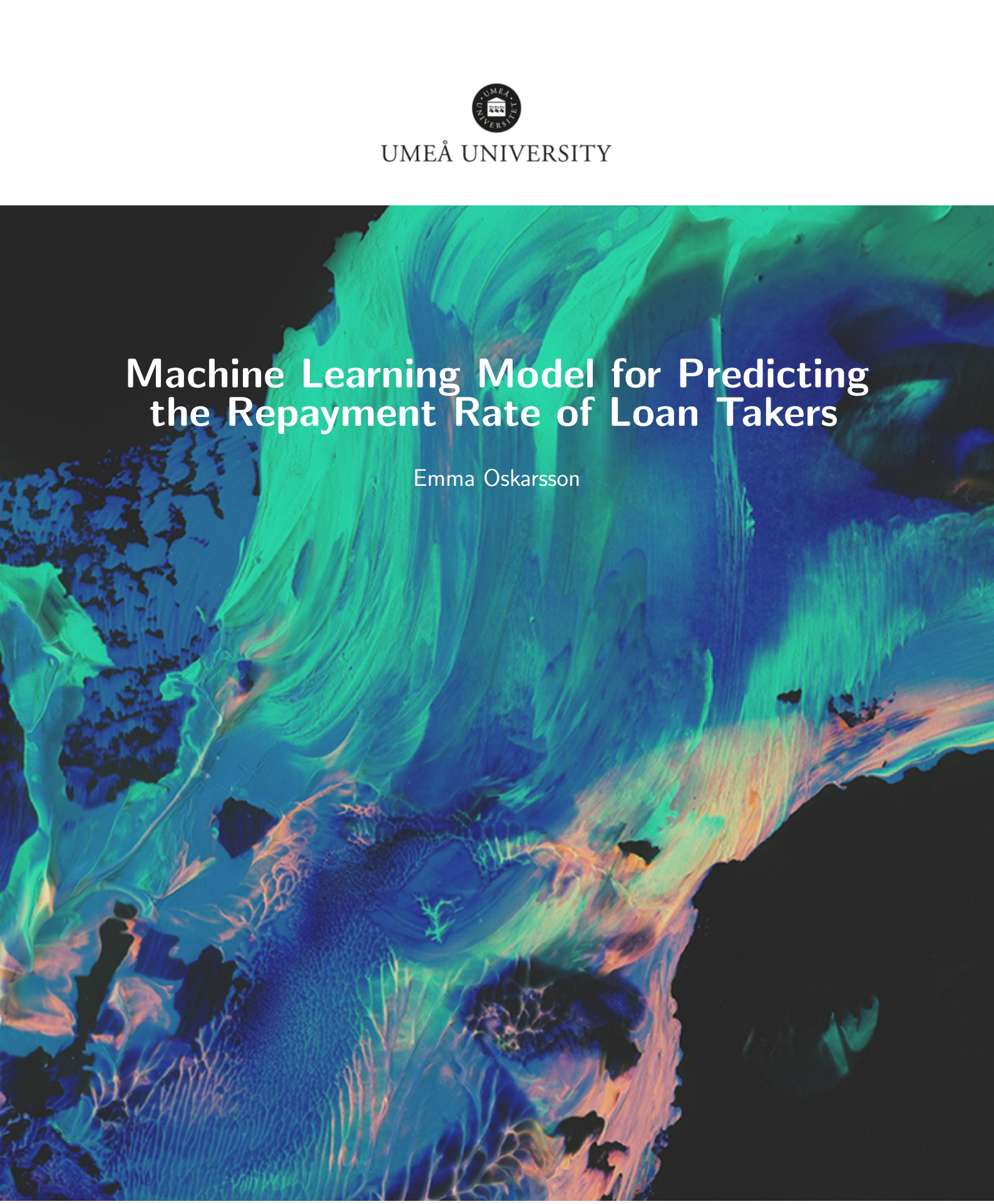


UMEÅ UNIVERSITY

Machine Learning Model for Predicting the Repayment Rate of Loan Takers

Emma Oskarsson

Master of Science in Engineering Physics 300 ECTS

Master's Thesis in Engineering Physics, 30 ECTS, Spring term 2021

Master's Thesis In Engineering Physics

Machine Learning Model for Predicting the Repayment Rate of Loan Takers

Emma Oskarsson

Department Of Physics

Umeå University

June 2021

Machine Learning Model for Predicting the Repayment Rate of Loan Takers

Examiner: Leif Nilsson, Department of Mathematics and Mathematical Statistics.

Supervisors: Xijia Liu, Department of Statistics.

Fredrik Lundquist, CSN.

Master's Thesis in Engineering Physics

Department of Physics

Umeå University

SE-901 87 Umeå, Sweden.

Abstract

Machine Learning (ML) uses statistics to find patterns in high dimensional data. The Swedish Board of Student Finance (CSN) wants to improve the way they classify new loan takers. Using Machine Learning (ML) on data from previous loan takers can establish patterns to use on new loan takers. The aim of this study is to investigate if CSN can improve the way they classify loan takers by their ability to pay back their loan. In this study, different ML models are applied to a data set from CSN, their performance are compared and investigated by the most related factors affecting an individuals repayment rate. A data set of a total of 2032095 individuals were analysed and used in the different models. Using Random Forest (RF) for binary classification produced the best result with a sensitivity of 0.9695 and a specificity of 0.8058.

Contents

Chapter 1

Introduction

This study will apply different Machine Learning (ML) models on a data set from CSN with the ambition to predict the repayment rate of future loan takers. The background to why this is of interest and how the problem can be approached will be presented in this *introduction*.

1.1 *Background*

ML uses statistics to find patterns in high dimensional data. The data encompasses a lot of things - numbers, words, images, clicks etc [1]. ML powers many services we use today - recommendation systems like those on Netflix or Youtube where they use user data to construct as relevant recommendations as possible.

The Swedish Board of Student Finance (CSN) is under development to become more digitised. CSN is responsible for the Swedish student loans, home equipment loans and drivers license loans. It is important that the money being lent out is payed back after an individual has completed their studies. CSN wants to be able to predict how well an individual is able to repay their loans.

In a study from 2013 they analysed the amount of money being paid back to CSN by focusing on earlier data [2]. From data analysis they drew a conclusion that the predominant part of the people having annuity loans also have a low income.

In a study from 2018 they analysed the amount of money being payed back through predictive analysis [3]. A predictive analysis implies doing statistical assumptions based on a big amount of data in order to do future predictions [4]. In this analysis they had to

change their study since a complete ML model was not possible. This was due to lack of quality in the data and the task being too complex to solve. They speculated about the ability to develop this study more in the future by using more complex data analysis.

The lack of previous studies using ML to find patterns to why individuals do not pay back their loans made this study even more important but also more challenging.

1.2 *Aim and Goal*

The aim of this study is to investigate if CSN can improve the way they classify loan takers by their ability to pay back their loan. It will also investigate which factors that are of interest in the repayment rate of the loan takers. This will be accomplished by exploring some of the most known models in the field of ML. Using ML on data from previous loan takers at CSN can help find patterns to use on new loan takers.

The goal of this study is to compare the implemented ML models to identify the preferred models for the used data and also to simplify the work of future researchers.

1.3 *Limitations*

There are three different loans that CSN administrates, the ones taken before 1989 called student aids, the ones taken between 1989 and 2001 called student loans and the ones taken after 2001 called annuity loans. Since the current loans distributed today are the annuity loans this work will be focusing on only those.

1.4 *Brief summary*

The data set received from CSN will need to be preprocessed before used in any of the ML models. The data set needs to be sanitised from values that do not make sense or are not allowed in the models. After this feature extraction can be conducted depending on the model, and the response variable will be created. The models that will be used to investigate performance are three classification models - Linear Discriminant Analysis (LDA), Logistic Regression (LR) and Random Forest (RF). The results will show that multiclass classification will not perform as well as binary classification. RF for binary classification will have the best performance.

1.5 *Layout*

In this report there are 5 main chapters, of which the first is this *introduction*. Chapter 2 will present the *theory* of the adopted ML models and their performance metrics. Chapter 3 contains the *method* section, which presents what pre-processing steps were applied to the data set and how that data was used in the models. The *results* are presented in chapter 4 and are then *discussed* in chapter 5 with a finishing *conclusion*.

Chapter 2

Theory

There are many different techniques to use in ML. In this study we have a known response variable, therefore supervised learning will be utilised. The response variable will also be divided into classes and then presented in both a binary and a multiclass problem. A classification model can be both linear and non linear. A linear classifier uses a linear combination of the characteristics and divides them into classes. Generally, it is preferred to keep the model as simple as possible by using a linear model, however, a non-linear model should be considered if the linear model can not handle the problem.

2.1 *Machine Learning Models*

In this section the applied models used in this study will be presented.

2.1.1 Linear Discriminant Analysis

Linear discriminant analysis (LDA) is used for linear classification but also data visualization and dimension reduction. LDA uses probability to classify samples into different populations. Even though LDA is a simple model it produces interpretable classification results [5]. LDA is based on *Naive Bayes* classification algorithm. With Bayes theorem we can calculate the probability that an event, C_k, is true given a feature vector $X = (x_1, x_2, ..., x_n)$. *Bayes theorem* then states that:

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)}, k = 1, 2, ..., K. \tag{2.1}$$

$P(C_k|X)$ is the posterior probability, $P(X|C_k)$ the likelihood, $P(C_k)$ the prior probability of class and $P(X)$ is the prior probability of the feature vector. The posterior probability can be calculated from the likelihood and prior probabilities. The *Naive independence assumption* states that $P(X|C_k) = P(x_1, ..., x_n|C_k) = \prod_{i=1}^{n} P(x_i|C_k)$. Using this, Equation 2.1 can be written as

$$P(C_k|X) = \frac{P(C_k)\prod_{i=1}^{n} P(x_i|C_k)}{P(X)}. \tag{2.2}$$

Using Equation 2.2 *Naive Bayes* classification problem can be stated as: for different values of C_k, find the maximum of $P(C_k)\prod_{i=1}^{n} P(x_i|C_k)$. More formally stated as

$$\hat{C} = arg\max_{C_k} P(C_k)\prod_{i=1}^{n} P(x_i|C_k). \tag{2.3}$$

Consider in population π_i there is a probability density function, x, which is a multivariate normal with mean vector μ and variance-covariance matrix Σ, that is the same for all populations. This normal probability density function is defined as

$$P(X|\pi_i) = \frac{1}{(2\pi)^{(\frac{p}{2})}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X-\mu_i)'\Sigma^{-1}(X-\mu_i)}. \tag{2.4}$$

The decision rule in LDA is based on *Linear Score Function* defined as

$$s_i^L = -\frac{1}{2}\mu_i'\Sigma^{-1}\mu_i + \mu_i'\Sigma^{-1}X logP(\pi_i) = d_{i0} + \sum_{j=1}^{p} d_{ij}x_j logP(\pi_i) = d_i^L(X) + logP(\pi_i), \tag{2.5}$$

where $d_{i0} = -\frac{1}{2}\mu_i'\Sigma^{-1}\mu_i$ and $d_{ij} = j$th element of $\mu_i'\Sigma^{-1}$ and $d_i^L(X)$ is the linear discriminant function [6].

As an example we have a set of data points from two features x_1 and x_2, shown in Figure 2.1. LDA wants to create a new axis with help from *Bayes formula* that separates the classes as much as possible. For each population Equation 2.5 is calculated and the example is classified into the the population with the largest score value. The new axis can appear as something similar to Figure 2.2.

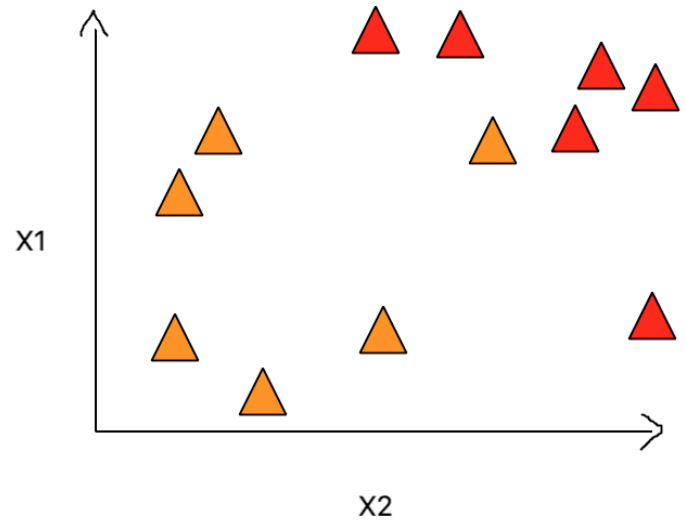


Figure 2.1: Presentation of an example set of a few data points of two classes.

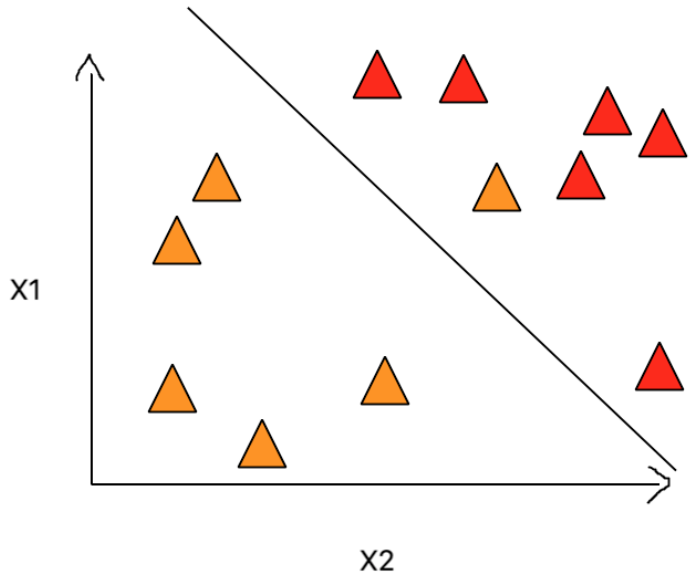


Figure 2.2: Presentation of the new axis created from the two classes.

2.1.2 Logistic Regression

Logistic regression (LR) is a linear classification model. To classify particular events in LR a *log-odds function* is used. In LR we model the probability that a response belongs to a certain class, rather than modelling the response directly. If we have a feature vector X and a response either being $y = 1$ or $y = 0$, the *odds ratio* for $y = 1$ can be described as

$$OR = \frac{P(y = 1|X)}{1 - P(y = 1|X)}. \tag{2.6}$$

The *Logit function* can be derived from Equation 2.6 by applying the logarithm to the right hand side and setting it equal to a linear equation:

$$a + b^T X = log(\frac{P(y = 1|X)}{1 - P(y = 1|X)}), \tag{2.7}$$

where a is an unknown parameter and b is an unknown vector with parameters. The inverse of Equation 2.7 is called a logistic or sigmoid function. The shape of the logistic function is presented in Figure 2.3. It is defined as

$$P(y = 1|X) = \frac{1}{1 + e^{-(a+b^T X)}}, \tag{2.8}$$

where e is Euler's number. For any $X \in R$, $P(y = 1|X)$ will be given a value between $0 - 1$ [7]. Since the parameters a and b from Equation 2.7 is unknown we want to estimate these parameters. Equation 2.8 outputs predicted probabilities, which in this case describes the predicted probabilities that X belongs to $y = 1$. We want to estimate the parameters a and b so that Equation 2.8 will give an output as close to 1 as possible for the predicted values that are correctly put into the right class [8]. This can be done by the likelihood function. This function can be defined as

$$L(a, b) = \prod_{i=1}^{n} P(y_i = 1|X_i)^{y_i}(1 - P(y_i = 1|X_i))^{1-y_i}). \tag{2.9}$$

taking the log of Equation 2.9 yields the *log likelihood* function:

$$LL(a, b) = \sum_{i=1}^{n} y_i log(P(y_i = 1|X_i)) + (1 - y_i)log((1 - P(y_i = 1|X_i)). \tag{2.10}$$

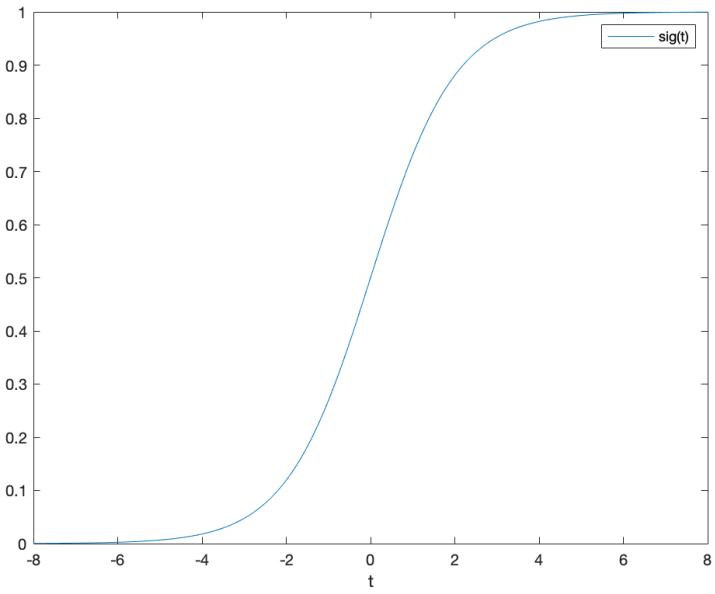


Figure 2.3: Representation of the sigmoid function.

If LR is used for multiclass classification we want to classify a sample into the class with the highest probability.

2.1.3 Random forest algorithm

A Random Forest (RF) is a non linear classifier based on an ensemble ML algorithm, also called bagging. In bagging several subsets from a data set are created chosen randomly with replacement. Bagging can be presented by

$$\hat{f}_{bag}(X) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x), \tag{2.11}$$

where $\hat{f}_{bag}(X)$ is an arbitrary model, B is represented by samples with replacement of our original data (bootstrapped samples), and $\hat{f}^{*b}$ is our trained model on the bth bootstrapped data sample [8]. Each collected subset data is used to train decision trees. This results in a collection of trained decision trees. The reason bagging is used is because taking the prediction average over the collected trained trees gives a more robust result than having only a single decision tree. RF is based on bagging algorithm but instead of using all features to grow a tree, RF instead uses random features [9].

A decision tree can for example look like the one in Figure 2.4. This decision tree describes whether someone should play golf today or not. Depending on the outlook of the weather one may choose to play or not. In a decision tree we always start at a root node, in this case the root node is "outlook". From this node several splits will split the data. Our internal nodes are then created, which in this case are represented by "humidity" and "wind".

Performing RF, there is a builtin feature importance, where a *Gini Impurity* is computed for each feature. Measurements are taken at each node of how the features decrease the impurity of the split. The feature with the highest decrease is selected for the internal node. Investigating the importance of features makes it possible to drop features not important for the model.

In RF we have hyperparameters that can either increase the predictive power or make the model faster. We have one hyperparameter describing the number of trees, which can improve the algorithm more when tuned. Another hyperparameter is the maximum number of features to split a node.

Figure 2.4: Representation of a decision tree where the outcome decides if you should play golf or not.

2.2 *Performance metrics*

2.2.1 Cohen Kappa

When dealing with imbalanced data in the response variable it is trickier to calculate the accuracy of your model since the outcome can be misleading. For example if one class consists of 96% of the data and the accuracy ends up to be 96% it does not mean the results are good. In those cases one way to measure accuracy is to use *Cohen's Kappa* [10]. Cohen's Kappa is a measure of how often two raters agree with each other. For example if you have a set of predicted values and a set of the real values you can check how often these two agree with each other. If $Kappa = 0$ it means there is a random agreement between raters, if $Kappa = 1$ there is a complete agreement between the raters and if $Kappa >= 0.4$ it means we have a good agreement. By considering Figure 2.5 we can understand more about the results from different raters. In this figure "A" means the total number of instances both raters said were correct, "B" means the number of instances Rater 1 said were incorrect but Rater 2 said were correct, "C" means the number of instances Rater 2 said were incorrect and Rater 1 said were correct and lastly "D" means that both the rater said it was incorrect. The reason for both "A" and "D" being highlighted is because in those cases Rater 1 and Rater 2 are in agreement with each other.

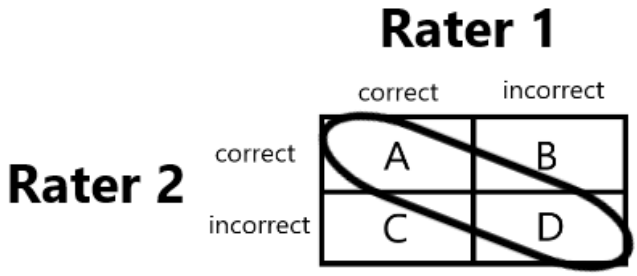


Figure 2.5: Agreement for the raters visualised.

With this information we can calculate the *Kappa* value defined as

$$K = \frac{P_0 - P_e}{1 - P_e}, \tag{2.12}$$

where P_0 is the number of agreed cases divided by the total number of cases, from Figure 2.5 we can express P_0 as

$$P_0 = \frac{A + D}{A + B + C + D}.$$

P_e is the probability of random agreement for both Rater 1 and Rater 2 which from Figure 2.5 can be expressed as

$$P_e = P_C + P_I, \tag{2.13}$$

where P_C is the expected probability that both would be correct at random and P_I is the expected probability that both would be incorrect at random.

2.2.2 Confusion Matrix and AUC-ROC Curve

A *confusion matrix* is a performance measure of a classification model using two or more classes. For each combination of two classes we can present the predicted and actual values in a matrix [11].

Table 2.1: confusion matrix where TP = true positive, FN = false negative, FP = false positive and TN = true negative.

Predicted/Actual	Positive	Negative
Positive	TP	FP
Negative	FN	TN

Calculating the *confusion matrix* using Table 2.1 produces values of the sensitivity and specificity by

$$sensitivity = \frac{TP}{TP + FN},$$

$$specificity = \frac{TN}{TN + FP}.$$

Where the sensitivity measures the proportion of positives that are correctly identified and specificity measures the proportion of negatives that are correctly identified.

The Receiver operating characteristic curve (ROC-curve) can be plotted by the sensitivity and specificity. The ROC-curve is a probability curve and the area under the curve (AUC) describes the probability of predicting into the right class by a value between $0 - 1$ [12].

Chapter 3

Method

In this section I will present the pre-processing of my data and the different models I have investigated. I have split my response variable into four classes in all of my models but I have also split it into two classes in both RF and LR.

3.1 *Data pre-processing*

In this study I have used a program called *R studio* and the programming language *R*. I have implemented anonymised data from CSN's database and the raw data is presented in Figure A.1 in Appendix. The variable descriptions are presented in Table 3.1. The data set I received consists of a total of 2032095 individuals. For each one there is data stored over a four year period, having different starting points ranging between 2010 and 2017. There is 30 descriptive variables for each individual and out of these, 19 is numerical and 11 is categorical. These variables are only a few out of the ones CSN has in store, they were chosen based on what could be interesting in this study.

DEBT_MONTH	The last month and year in the four year period for that person.
GENDER	What gender the person has.
INCOME_YEAR1	Income in the first year of the four year period.
INCOME_YEAR2	Income in the second year of the four year period.
INCOME_YEAR3	Income in the third year of the four year period.
INCOME_YEAR4	Income in the fourth year of the four year period.
VALID_ADDRESS	If the persons address is valid.
REFUND	If the person has refund to repay because of unfinished studies for ex.
SEVERAL_LOANS	If the person has more than two loans.
EDUCATION	Highest education level.
RETURNS	Amount of letter returns back to CSN from a certain address.
BIRTH	Which year the person is born.
STATUS	Population status.
COUNTY	What county the person lives in.
REPAYMENT_DATE	Year when the person started to pay back.
DEBT	The full debt when the person started to pay back.
PAYED_YEAR	When the current four year period started for the person.
REDUCED	How much the CSN has been reduced for that person.
ORIGIN_CHARGE_DECISION	Describes how the original debt has been calculated.
CHARGE_DECISION	Describes what the charge decision is. If for ex. ORIGIN_CHARGE_DECISION is changed.
REASON_REDUCED	Reason debt has been reduced for that person.
PREL_AMOUNT	New amount to pay back over the four year period for that person after reduction.
PAYED_YEAR1	What has been payed year one out of the four year period for that person.
PAYED_YEAR2	What has been payed year two out of the four year period for that person.
PAYED_YEAR3	What has been payed year three out of the four year period for that person.
PAYED_YEAR4	What has been payed year four out of the four year period for that person.
PAYED_TOTAL	Total amount that has been paid over the four years for that person.
DEBT_TOTAL	Total amount that should be repayed over the four years for that person.
REMAINING	How much the person has left to pay back after one year.

Table 3.1: Variable descriptions from the raw data set.

The data had a number of issues that had to be solved before going further. The variables removed are stated in a list here below with an explanation to why that happened.

- In the variable RETURNS I had a factor with "-1" values.
  This was solved by changing all "-1" to "0".

- The variable REPAYMENT_DATE had a problem with non-existing years.
  This variable was removed.

- The variable REMAINING only considered the first year of the four year period.
  This variable was removed.

- The variables DEBT_MONTH and PAYED_YEAR did not contribute anything because of the lack of information.
  These variables were removed.

- PAYED_YEAR1, PAYED_YEAR2, PAYED_YEAR3 and PAYED_YEAR4 depends on the response variable.
  These variables were removed because training models on variables depending on the response variable gives misleading results.

There were some other pre-processing steps needed to be done before running the models which are listed here below.

- The variables GENDER, RETURNS, VALID_ADDRESS, REFUND, SEVERAL_LOANS, BIRTH and PAYED_YEAR were presented as numerical.
  I changed them to factors with the tool `as.factor()` in *R studio* because there is no clear order or distance in these variables.

- In the variable REPRATE I had a factor with "NA" values when DEBT_TOTAL was 0.
  This was solved by changing REPRATE to 1 when DEBT_TOTAL is 0.

- For the variables VALID_ADDRESS, REFUND and SEVERAL_LOANS there were a blank factor.
  I solved this by changing the blank spaces for an "N" (no), because these variables had two factor levels explaining a "yes" or "no" outcome.

After removing and modifying a few variables the variable AGE_PERIOD was created. This was done by using the variables PAYED_YEAR and BIRTH to calculate how old a person is at the beginning of the four year period.

3.2 *Response variable*

The response variable REPRATE was created. This was done by dividing PAYED_TOTAL and DEBT_TOTAL. REPRATE is defined as the ratio between what an individual has payed over this four year period divided by their full debt over this period. I split this REPRATE variable into classes before training the models, either four classes for the multiclass classification or two classes for the binary classification. When using multiclass classification we call the response variable REP_multi and the classes for REP_multi are divided as: class 1 contains individuals with a repayment rate of 100%, class 2 contains individuals with 0%, class 3 contains individuals with $> 0\%$ and $< 50\%$ and class 4 contains individuals with $>=$ 50% and $< 100\%$. For binary classification we call the response variable REP_binary and the classes are divided as: class 1 contains individuals with a repayment rate of 100% and class 2 contains the individuals with $>= 0\%$ and $< 100\%$

3.3 *Training set and Test set*

Before training the models the data set are divided into a training set and a test set. This is done by using the base function *sample* as

```
sample(1:nrow(data), 0.75*nrow(data)).
```

This means 75% of the data are represented by the training set and 25% of the data are represented by the test set. The reason we split our data into two sets is because we want to test our models on a set of data that it has never seen before. One problem that can occur in ML is that our model learns the training data too well and will perform bad on the test data - also called overfitting [13]. After training a model on a test set we can use the function *predict* from the package 'car' to predict the outcome of a testing set.

3.4 *Linear Discriminant Analysis*

The categorical variables had to be removed before using LDA, since LDA can only handle numerical variables. In Figure A.2 in Appendix all the variables used in this model is presented. From *R Studio* I installed a package called 'MASS' so I could use the *lda* function [14]. It was used as

$$\texttt{lda(REPRATE} \sim \texttt{.,}\ \textbf{data} = \texttt{traindata)},$$

where REPRATE is represented by REPRATE_multi. After the model was trained I used the function *predict* to predict the outcome of the test set. I calculated the accuracy of the predicted model and then the *Cohen's Kappa* from Equation 2.12 to see how well the model performed on the test data. The linear discriminants were visualized by applying the transformation matrix on the scaled data after centering it around the weighted means. I used the function *ggplot* in package 'ggplot2' [15] to visualize this.

The *confusion matrix* was then calculated to measure the performance of the model. I used the function *confusionMatrix* [16] in the package 'caret' to calculate the values of the specificity and sensitivity for the different combinations of classes. I plotted the ROC-curve by using the the function *multiclass.roc* [17] in package 'pROC'.

3.5 *Logistic Regression*

In this section I will explain how I performed both multinomial and logistic regression.

3.5.1 Multiclass approach

I installed the package 'nnet' which containes a function called *multinom* [18] that can calculate multinomial regression. I called the function as

```
multinom(REPRATE ~ ., data = traindata),
```

where REPRATE is represented by REPRATE_multi.

After the model was trained I used the function *predict* to predict how well the trained model performed on the test data. I calculated the accuracy of the predicted model. Then I used Equation 2.12 to calculate *Cohen's Kappa*. The *confusion matrix* was calculated so I could get a result of the sensitivity and specificity. I wanted to plot the sensitivity and specificity so I used the *multiclass.roc* function to plot the ROC-curve.

3.5.2 Binary approach

I installed the package 'stats' to use the function *glm* [19]. I used 100000 values out of the full data set to train the model since I could not get the model to converge using the full data set. I removed the variable AGE_PERIOD since it consisted of too many factors for the model to converge. The function was used as

```
glm(REPRATE ~ ., data = traindata,
        family = quasibinomial(link = "logit"),
        control = list(maxit = 1000)),
```

where REPRATE is presented by REPRATE_binary, `family = quasibinomial(link = "logit")` means that we want a quasibinomial distribution which is logistic, and `control = list(maxit = 1000)` means we increase the number of maximum iterations.

I used the *predict* function on my trained model to predict how well the trained model performed on my test data. I calculated *Cohen's Kappa* and the *confusion matrix*. I got a result of the sensitivity and specificity and then plotted the ROC-curve with help from the function *roc*.

3.6 *Random forest algorithm*

I installed the package 'ranger' to use the function *ranger* [20]. This function is well suited for performing RF for high dimensional data. In both multiclass classification and binary classification I tuned my hyperparameters by using *Grid Search*: First I used the default `num.trees = 500`. I changed `mtry` in a range of $2 - 10$ with a step of 1, keeping the same `num.trees`. After finding the optimal `mtry` I kept `mtry` constant and changed `num.trees` in a range of $100 - 1000$ with a step of 100 to find the optimal num.trees for that particular `mtry`.

Running RF yields a result for an *Out Of Bag error* (OOB), which is a measurement for the prediction error.

3.6.1 Multiclass approach

I used the function *ranger* as

```
ranger(REPRATE ~ ., data = traindata,
       num.trees=NTREES, mtry=MTRY, importance = 'impurity'),
```

where REPRATE is represented by REPRATE_multi, `num.trees` is the number of trees, `mtry` is the number of variables to possibly split at each node and importance means we want the output of variable importance. I then used the function *predict* to predict the performance of my test data.

After tuning my hyperparameters I calculated the *confusion matrix* so I could get the sensitivity and specificity. I did this by using the function *confusionMatrix*. Because of the low specificity I decided to add weights to the different classes before running the model. This could be done by adding the hyperparameter 'case.weights' to the *ranger* function. First I assigned class 1 a weight of 1 and the other classes a weight of

```
sum(amount in class 1)/sum(amount in class i),
i = 2,3,4.
```

I kept increasing the weights of all the classes except class 1 to improve the result as much as possible. I wanted to balance out the probability for the observations when running the model. The accuracy was calculated and also the *Cohen's Kappa*. The sensitivity and specificity was calculated again after adding weights to the function. The ROC-curve was plotted from this by using the function *multiclass.roc*.

The variable importance received from *ranger* was divided by the sum of the variable importance over all of the variables to get a percentage value, then they were plotted for visualization.

3.6.2 Binary approach

I used the function *ranger* as

```
ranger(REPRATE ~ ., data = traindata,
        num.trees=NTREES, mtry=MTRY, importance = 'impurity'),
```

where REPRATE is represented by REPRATE_binary, `num.trees` is the number of trees, `mtry` is the number of variables to possibly split at each node and importance means we want the output of variable importance. I used the trained model in the function *predict* to predict the performance of my test data.

I tuned the hyperparameters and calculated the *confusion matrix* with the function *confusion_matrix* in package 'mlr3measures'. I plotted the variable importance I received from *ranger*. To improve the results I added weights to my classes. Class 1 to a weight of 1, class 2 to a weight of

```
sum(amount in class 1)/sum(amount in class 2).
```

To improve my results even more I doubled the weight for class 2 and ran the model again. After improving the results I calculated the *confusion matrix* again and the belonging sensitivity and specificity. I finished off with plotting the ROC-curve with the function *roc*.

Chapter 4

Result

In this section I will present the different results received from the different models.

4.1 *Linear Discriminant Analysis*

Using the *lda* function I got a three coefficients of linear discriminants. $LDA_1 = 0.9801$, $LDA_2 = 0.0198$ and $LDA_3 = 0.0002$. In Figure 4.1 the first two discriminant variables are plotted to get a visualization of the data. The data are split into different classes but the classes does not have any clear space between them, nor does class 3 or class 4 show any clear points.

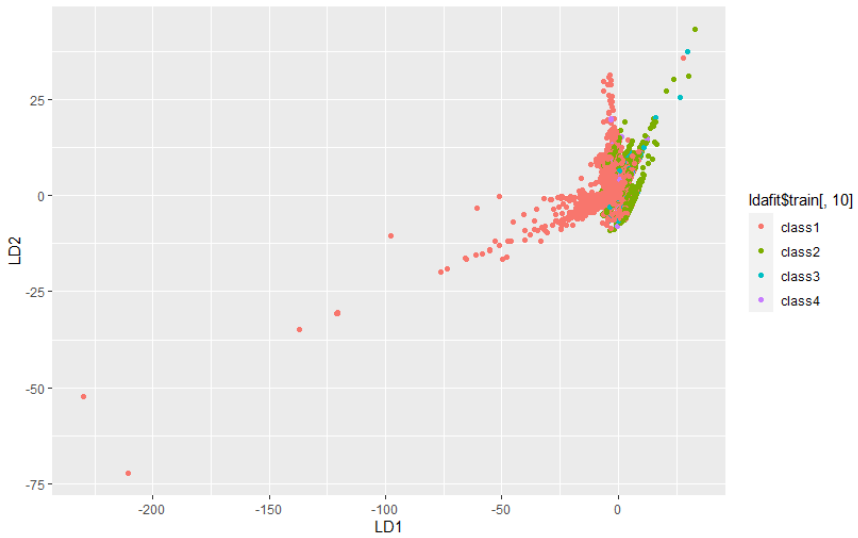


Figure 4.1: The first two discriminants plotted and divided into different colors depending on class.

I calculated the accuracy to be 96% and *Cohen's Kappa* to be 0.19. I calculated the *confusion matrix* and received results of the sensitivity and specificity presented in Table 4.1. The ROC curve is seen in Figure 4.2. The AUC resulted in 0.6646, which could be calculated by the mean from all pairwise class comparisions.

Table 4.1: Sensitivity and specificity for the different classes in LDA.

Class	**Sensitivity**	**Specificity**
Class 1	0.99712	0.127
Class 2	0.13888	0.997
Class 3	0.00288	1
Class 4	0	1

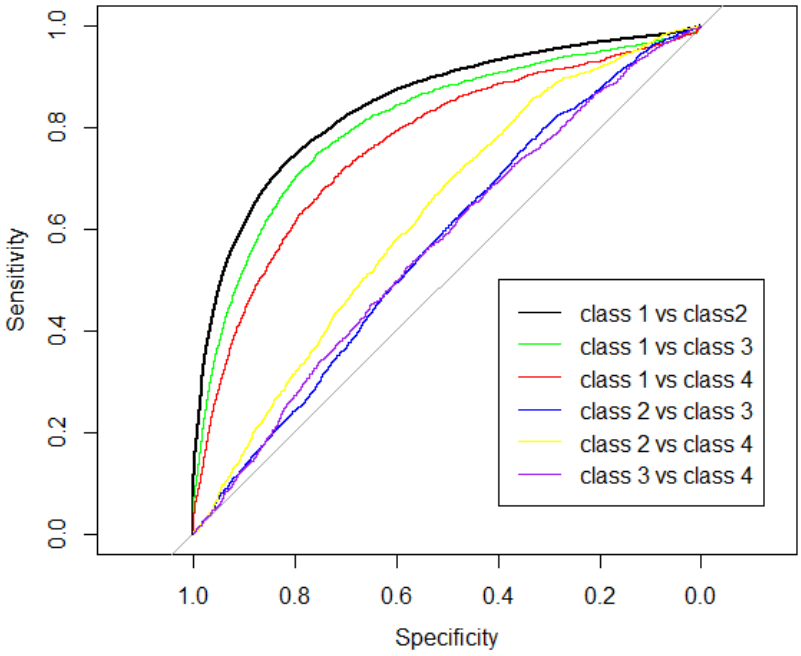


Figure 4.2: ROC curves from LDA.

4.2 *Logistic Regression*

4.2.1 Multiclass approach

I used the *multinom* function to perform LR. I predicted my test data on the model and received an accuracy of 97% and a *Cohen's Kappa* of 0.484. I calculated the *confusion matrix* to get a result of the sensitivity and specificity, the results are performed in Table 4.2

Table 4.2: Sensitivity and specificity for the different classes in LR.

Class	**Sensitivity**	**Specificity**
Class 1	0.99180	0.436
Class 2	0.46229	0.991
Class 3	0.01070	1
Class 4	0.00526	0.999

In Figure 4.3 the sensitivity is plotted against the specificity using the *multiclass.roc* function. The AUC was calculated to be 0.7093.

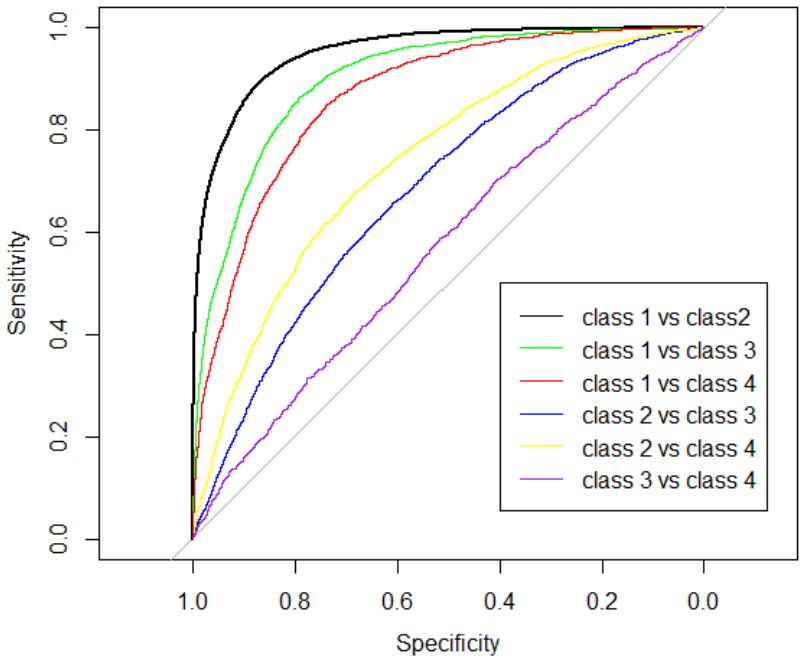


Figure 4.3: ROC curve for the multiclass LR.

4.2.2 Binary approach

I used the *glm* function to perform binary LR. After predicting the test data and creating a new vector of the predicted classes I received an accuracy of 97% and a *Cohen's Kappa* of 0.536. I calculated the *confusion matrix* and collected results of the sensitivity and specificity. I got the sensitivity to be 0.9940 and the specificity to be 0.4326. The ROC curve is presented in Figure 4.4 and the AUC was calculated to be 0.946.

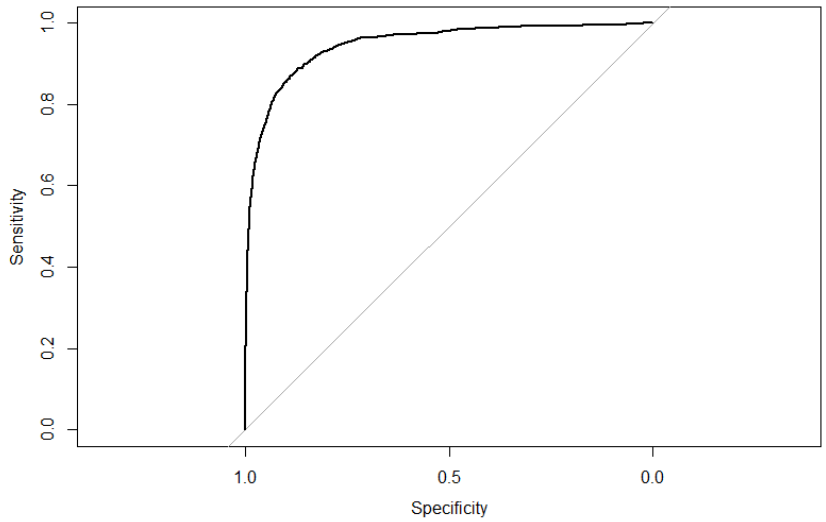
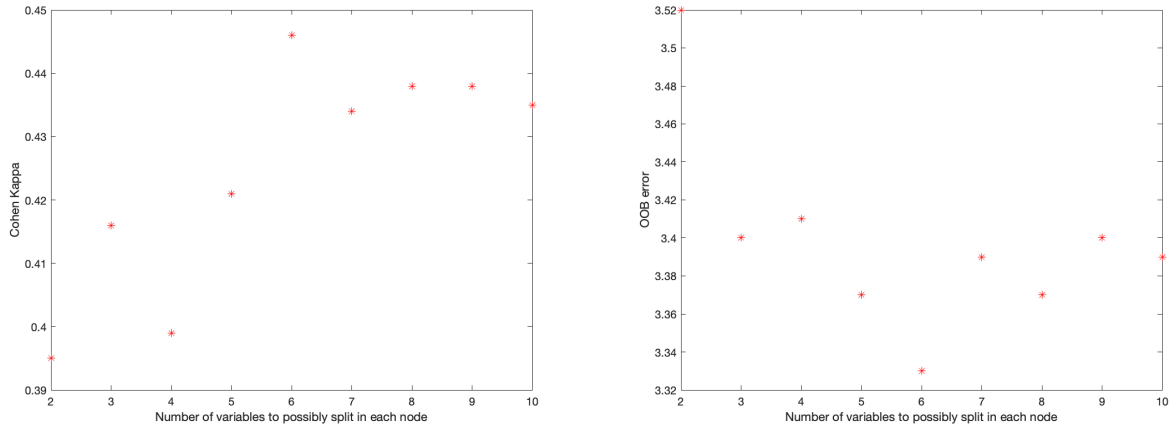


Figure 4.4: ROC curve for the binary LR.

4.3 *Random forest algorithm*

4.3.1 Multiclass approach

In Figure 4.5 the *Cohen's Kappa* and *OOB error* is presented depending on the number of variables to split in each node. I chose `mtry` to 6 since that gave me the best result for *Cohen's Kappa* while still having a good result of the *OOB error*.



(a) Cohen Kappa

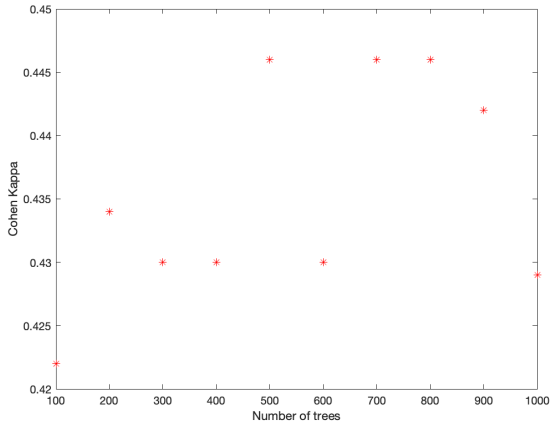
(b) OOB error

Figure 4.5: How Cohen's Kappa and OOB error is changing depending on mtry for the default number of trees = 500.

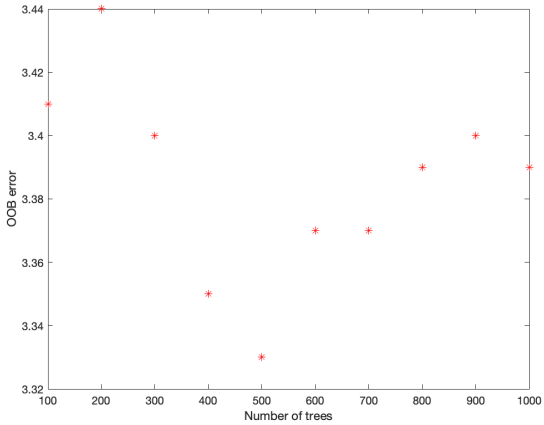
In Figure 4.6 I kept `mtry` constant at 6 and instead changed `num.trees`. I found that `num.trees` = 500 gave me a great result while still keeping OOB low. Running my full data set on `num.trees` = 500 and `mtry` = 6 gave me an accuracy of 96%, a *Cohen's Kappa* of 0.622 and *OOB error* of 2.52%. However the sensitivity for both class 3 and class 4 became very low. After adding weights to all classes I got an improved result which can be seen in Table 4.3. In this case I got an *OOB error* of 2.71% and *Cohen's Kappa* of 0.6.

In Figure 4.7 the variable importance of the different features are presented. The different numbers corresponds to one feature.

The ROC curve are presented in Figure 4.8 and the AUC was calculated to be 0.675.



(a) Cohen Kappa



(b) OOB error

Figure 4.6: How Cohen's Kappa and OOB error is changing depending on number of trees for mtry = 6.

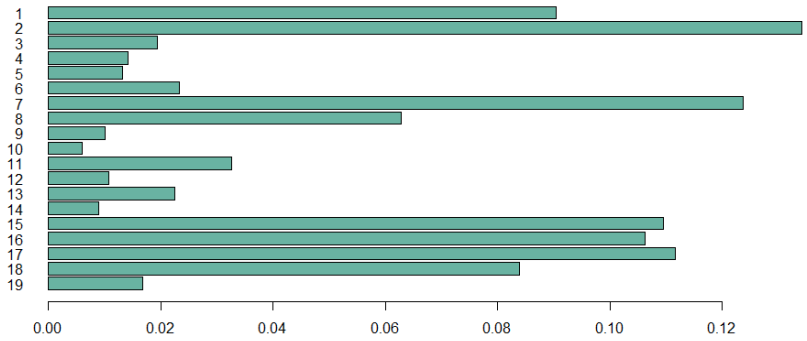


Figure 4.7: The variable importance of the different features. In Tab. A.1 in the Appendix it is presented which variable the different numbers are assigned to.

Table 4.3: Sensitivity and specificity calculated from RF with weights of the different classes.

Class	**Sensitivity**	**Specificity**
Class 1	0.9739	0.788
Class 2	0.8351	0.972
Class 3	0.1017	0.999
Class 4	0.0829	0.999

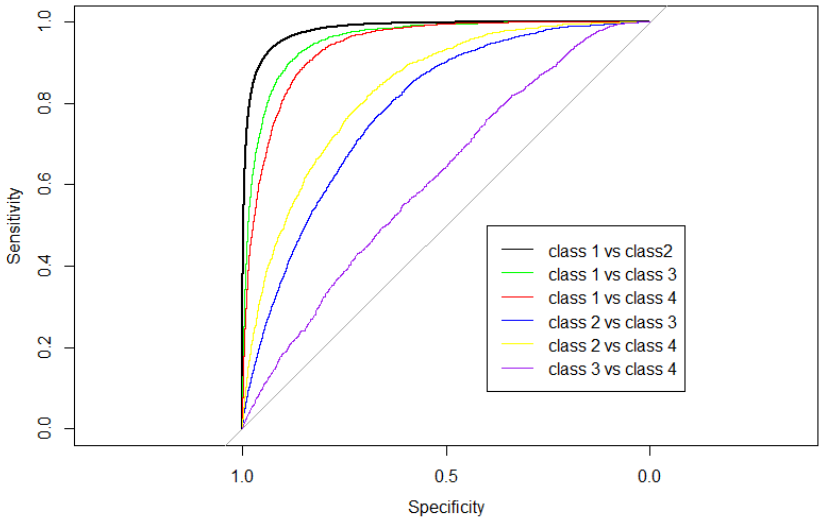
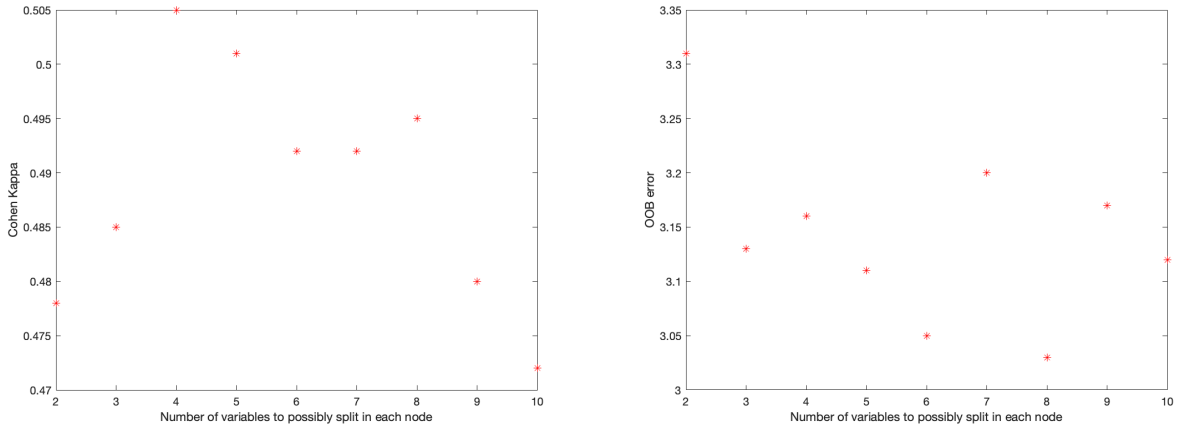


Figure 4.8: ROC curves from multiclass RF.

4.3.2 Binary approach

In Figure 4.9 it is presented how *Cohen's Kappa* and *OOB error* is changing depending on `mtry` for a default `num.trees`. From this I chose `mtry = 4`, because it gave me the best result. I did the same thing again but keeping `mtry = 4` and changing `num.trees`. This is seen in Figure 4.10. I chose `num.trees = 200`.

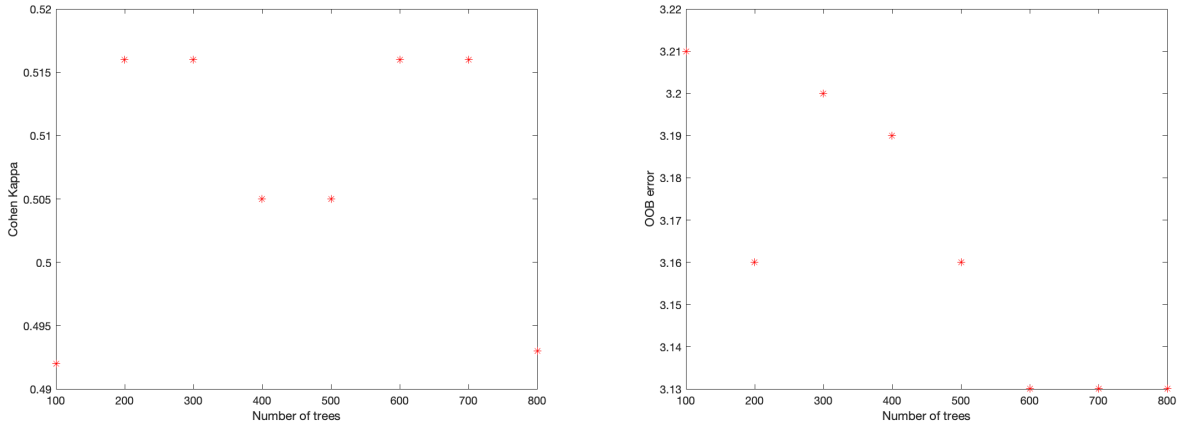


(a) Cohen Kappa

(b) OOB error

Figure 4.9: How Cohen's Kappa and OOB error is changing depending on mtry for default number of trees = 500.

Using the tuned hyperparameters on the full data set gave me an accuracy of 96%, a *Cohen's Kappa* of 0.675 and an *OOB error* of 2.23%.



(a) Cohen Kappa

(b) OOB error

Figure 4.10: How Cohen Kappa and OOB error is changing depending on num.trees for mtry = 4.

Calculating the *confusion matrix* gave me a sensitivity of 0.9946 and a specificity of 0.5873. Class 1 was seen as the positive outcome and class 2 was seen as the negative outcome. To improve the result I added weights to the classes and ran the model again. I obtained a sensitivity of 0.9695 and a specificity of 0.8058. However the *Cohen's Kappa* decreased to a value of 0.623 and the *OOB error* also increased to 3.05%. The ROC curve is plotted in Figure 4.11 and the AUC was calculated to be 0.9764.

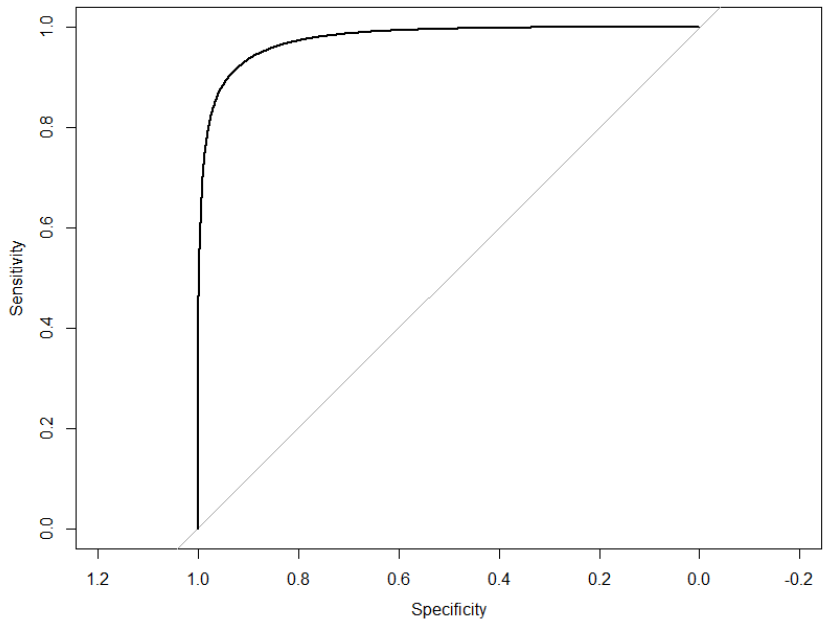


Figure 4.11: ROC curve for binary RF.

I plotted the variable importance received from *ranger* and the result is given in Figure 4.12.

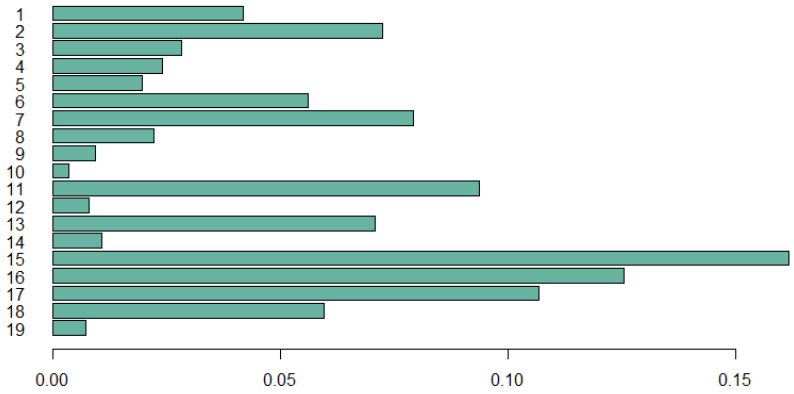


Figure 4.12: The variable importance of the different features in the binary case.

Chapter 5

Discussion and conclusion

In this section I will discuss the results I have received but also how this study can be developed in the future.

5.1 *Main conclusion*

There are many reasons why individuals cannot repay their loans which cannot be measured. For example if someone have lost someone that is close to them, if they get sick so they cannot work or maybe if someone goes through a breakup. All of this can affect someone not paying back their loan but that is something we cannot take into account by a variable in a data set.

However we can build models based on the data that we do have and see if we can find patterns from that. If we consider all three models I used in the multiclass approach, all of them gave different results. At a first glance the results seems really good since the accuracy ranged between 96-97% for all models. However we cannot observe only accuracy when dealing with imbalanced data for the response variable. Instead we have to compare other performance metrics. Comparing the sensitivities, LDA had by far the worst performance, LR was not far behind and RF performed the best results, but still not acceptable. Instead if we compare the AUC, LDA had a value of 0.6646, LR of 0.7093 and RF of 0.675. This means that all three models almost had a seventy-thirty chance of predicting into the right class. Because of the bad results from multiclass classification I decided to try binary classification for both RF and LR to see if I could improve the results, which I did. Comparing the AUC, LR got an AUC of 0.946 and RF got an AUC of 0.9764, both are really great results. If

we instead compare the sensitivity and specificity, RF gave the most promising result, with a sensitivity of 0.9695 and a specificity of 0.8058. This means we can predict 97% of the people in class 1 and 81% of the people in class 2 into the right class. In RF the variable importance were also presented and considering the binary classification obtaining the best result out of the two, Figure 4.12 is more dependable than Figure 4.7. In Figure 4.12 it is clear that three of the income variables - 15, 16 17 are the most important ones for the model. Thinking logically, having an income can affect a lot on the ability to pay back your loan. Another thing I found interesting was that variable 13 - education, were one of the important variables. Depending on your education the ability to be recruited for a job can differ, thus affecting the ability to repay your loan.

5.2 *Future work*

There are several things to consider in a future study. For example to use more feature extraction to create new variables that would be more important and interesting than the ones already existing. It could be a possibility to try the function *multinom* for logistic regression to investigate if the model is able to converge. Another thing to consider would be to develop the code so it does not matter for what time someone starts paying back their money. Right now these models works in a way where a person has a four year period of paying back/not paying back their loan. In the future it would therefore be interesting to observe if we can predict how someone that only has been paying back for, lets say a month, is going to pay back in the future. Or taking it even further, if we can predict how a person that has not even started to pay back, will pay back in the future. There are a lot of possibilities in this area but considering the outlook of the data, classification should be considered for this problem. The results would be very misleading if regression analysis would be considered, then the predicted results would be values close to 100 when there are so many that are exactly 100.

Bibliography

[1] MIT technology. What is machine learning?, 2018. URL https://www.technologyreview.com/2018/11/17/103781/what-is-machine-learning-we-drew-you-another-flowchart/.

[2] CSN. Andel av debiterat belopp som betalas in. 2013.

[3] CSN. Uppdrag: Metodutveckling prediktiv analys, med tillämpning; identifiera kännetecken hos låntagare som i högre utsträckning riskerar att inte fullgöra sina betalningsåtaganden. 2018.

[4] Investopedia. Predictive analytics, 2021. URL https://www.investopedia.com/terms/p/predictive-analytics.asp.

[5] Towards Data Science. Linear discriminant analysis, 2020. URL https://towardsdatascience.com/linear-discriminant-analysis-explained-f88be6c1e00b.

[6] Shuzhun Fan. Understanding the mathematics behind linear discriminant analysis (lda), 2018. URL https://shuzhanfan.github.io/2018/07/understanding-mathematics-behind-lda/.

[7] Machine Learning Mastery. Logistic regression for machine learning, 2020. URL https://machinelearningmastery.com/logistic-regression-for-machine-learning/.

[8] Trevor Hastie Robert Tibshirani Gareth James, Daniela Witten. An intoduction to statistical learning. 2017.

[9] Built in. A complete guide to the random forest algorithm, 2019. URL https://builtin.com/data-science/random-forest-algorithm.

[10] Towards Data Sciene. Cohen's kappa, 2020. URL https://towardsdatascience.com/cohens-kappa-9786ceceab58.

[11] Towards Data Science. Understanding confusion matrix, 2018. URL https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62.

[12] Towards Data Science. Understanding auc - roc curve, 2018. URL https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.

[13] Machine Learning Mastery. Overfitting and underfitting with machine learning algorithms, 2016. URL https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/.

[14] Rdocumentation. lda: Linear discriminant analysis, . URL https://www.rdocumentation.org/packages/MASS/versions/7.3-53.1/topics/lda.

[15] Rdocumentation. ggplot2, . URL https://www.rdocumentation.org/packages/ggplot2/versions/3.3.3.

[16] Rdocumentation. confusionmatrix, . URL https://www.rdocumentation.org/packages/caret/versions/3.45/topics/consequencefusionMatrix.

[17] Rdocumentation. multiclass.roc: Multi-class auc, . URL https://www.rdocumentation.org/packages/pROC/versions/1.17.0.1/topics/multiclass.roc.

[18] Rdocumentation. multinom: Fit multinomial log-linear models, . URL https://www.rdocumentation.org/packages/nnet/versions/7.3-15/topics/multinom.

[19] Rdocumentation. glm: Fitting generalized linear models, . URL https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/glm.

[20] Rdocumentation. ranger: Ranger, . URL https://www.rdocumentation.org/packages/ranger/versions/0.12.1/topics/ranger.

Appendices

Appendix A

```
$ DEBT_MONTH                : chr
$ GENDER                    : int
$ INCOME_YEAR1              : int
$ INCOME_YEAR2              : int
$ INCOME_YEAR3              : int
$ INCOME_YEAR4              : int
$ VALID_ADDRESS             : chr
$ REFUND                    : chr
$ SEVERAL_LOANS             : chr
$ EDUCATION                 : chr
$ RETURNS                   : int
$ BIRTH                     : int
$ STATUS                    : chr
$ COUNTY                    : chr
$ REPAYMENT_DATE            : chr
$ DEBT                      : int
$ PAYED_YEAR                : int
$ REDUCED                   : int
$ ORIGIN_CHARGE_DECISION: chr
$ CHARGE_DECISION           : chr
$ REASON_REDUCED            : chr
$ PREL_AMOUNT               : int
$ PAYED_YEAR1               : int
$ PAYED_YEAR2               : int
$ PAYED_YEAR3               : int
$ PAYED_YEAR4               : int
$ PAYED_TOTAL               : int
$ DEBT_TOTAL                : int
$ REMAINING                 : int
$ NR_OF_REMINDERS           : int
```

Figure A.1: All the variables and their type from the full data set.

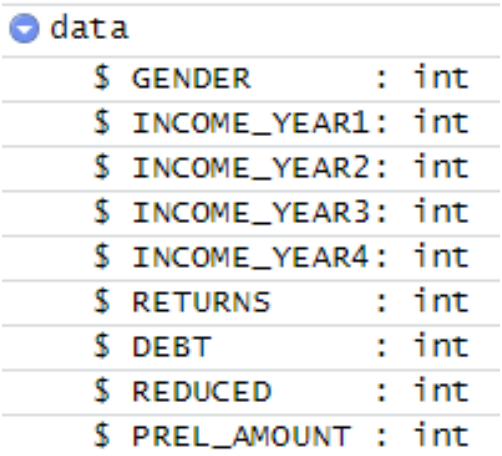


Figure A.2: All the variables used in Linear Discriminant Analysis.

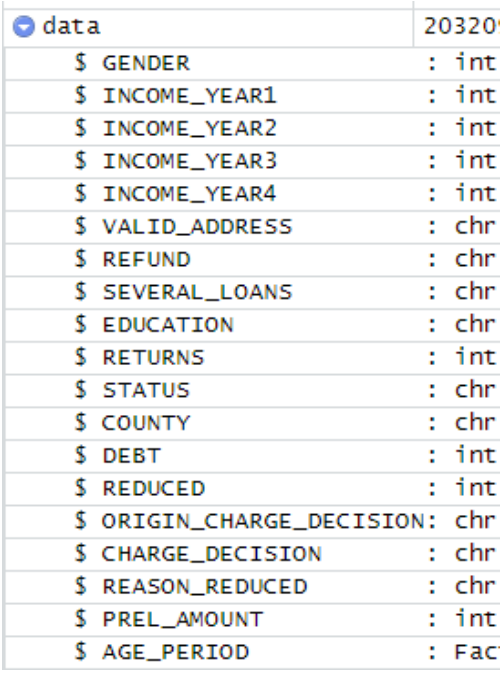


Figure A.3: All the variables and their type from the data set that was used in LR and RF Models.

Table A.1: What number the different features are assigned in variable importance, Figure 4.7

Number	Variable name
1	AGE_PERIOD
2	PREL_AMOUNT
3	REASON_REDUCED
4	CHARGE_DECISION
5	ORIGIN_CHARGE_DECISION
6	REDUCED
7	DEBT
8	COUNTY
9	STATUS
10	RETURNS
11	EDUCATION
12	SEVERAL_LOANS
13	REFUND
14	VALID_ADDRESS
15	INCOME_YEAR4
16	INCOME_YEAR3
17	INCOME_YEAR2
18	INCOME_YEAR1
19	GENDER