INSTITUTO SUPERIOR TÉCNICO

ADVANCED AUTOMATION

Data Analysis:

CREDIT CARD CHURNING

TÉCNICO LISBOA

Masters in Mechanical Engineering

PROFESSOR JOÃO SOUSA

ÁLVARO LOPES N96148

ANDRÉ LOPES N96351

JOÃO OLIVEIRA N96403

MIGUEL VALVERDE N96455

January 20, 2023

Index

List of Figures

List of Tables

Abstract

In October 2021, a study forecasted that by 2023, 29.3 billion credit, payment and pre-paid cards would be in circulation worldwide [4]. This number is increasing year by year and with that, the necessity for banks to be able to predict which of their customers are likely to terminate a credit card or not. In this report, we will look at credit card churning data and use it to build predictive models capable of determining if a customer is going to terminate an account or not. We used Classification and Tree-Based methods as well as Artificial Neural Network (ANN) and Support Vector Machine (SVM) to develop models for this problem. Under ANN and various classification methods, the best accuracy scores we achieved were in the order of 90% (a decent value but probably too low due to the complexity of ANN, for example). SVM yielded slightly better results than ANN or classification methods but tree-based classification, more specifically the Random Forest method, was clearly the best. With this method, we could achieve accuracies up to 99% without much computational effort.

Author Contributions

All of the elements of the group were present in the initial project choice and analysis as well as in the outlining of its structure and division among group members. The author Miguel Valverde was responsible for researching some papers and thesis on this subject in order to understand what type of methods were generally used to process this data. The authors Álvaro Lopes and André Lopes contributed to the data importation and correlation between the variables. André Lopes also worked on the data visualization aspect by plotting histograms, boxplots, etc for the processed data.

The authors André Lopes and João Oliveira were responsible for implementing feature selection methods and support vector machine (SVM) along with accuracy scores and ROC curves, among others. Álvaro Lopes did the same for tree-based classification methods and neural networks, and Miguel Valverde for classification methods. All of the elements participated in the elaboration of the report as well as in the discussion and analysis of the results.

1 Introduction

This report aims to present the project developed for the Advanced Automation course. In it, we will cover what is the project goal and our reason for choosing it, the methods we have used and their characteristics, and the results we obtained for each of them. A big part of the report will be spent on understanding the pros and cons of each method and when it is best to use them.

As was already mentioned in the project proposal, our group chose to analyse **Databases**, more specifically, data related to **credit card churning**. We ended up choosing Databases because the possibility of processing huge amounts of raw data and using them to make decisions on real-world scenarios, was something we as a group found very appealing. We would also be able to apply many of the methods we have learned in class and understand them better.

After some research in online databases for data processing and machine learning, we ended up choosing this credit card dataset [1]. Not only is it interesting to know what factors influence a customer cancelling a credit card, but also the fact that the dataset had quite a big number of features and allowed us to develop a neural network, proved to be the factors for our choice. We

will now analyse the characteristics of the dataset, visualise some of its data and look at what methods other works with a similar data type use (best-performing methods).

2 Dataset and Appropriate Methods

2.1 Dataset Context and Usage

This dataset is **updated annually** and has various data on customers with credit cards. Issuing a credit card is always a risky endeavour as the customer may use the card and end up having no money to pay back the borrowed money or be dissatisfied with the service and end up cancelling it. For banks, it is useful to know what is the profile of the client which is going to be reliable and stick with them. If they know this, they can use the data to make decisions on the issuing of credit cards or not for new customers.

At present, with the development of machine learning algorithms, many predictive methods such as **Boosting**, **Random Forest**, and **Support Vector Machines** (SVM) have been introduced into credit card cancelling prediction. However, it may be difficult to provide customers and regulators with a reason for rejection or acceptance based on these methods. Each case is a case and many times the accuracy rates are high but not as high as we would want. Still, building machine learning models to predict if an applicant is a '*good*' or '*bad*' client is becoming a more common practice.

2.2 Data Values

We started by importing the '*credit_card_churn.csv*' file containing all the data regarding the customers. In total there are 10127 **entries** (customers) and 19 **features** (columns) - figure 1. 5 of those are categorical features and 14 numerical features. This data tells personal information about a customer such as his age (*Customer_age*); sex (*Gender*); salary *(Income_category*); educational level (*Education_level*); marital situation (*Marital_stat us*) and number of successors (*Dependencies*).

It also has banking data such as if the account was terminated or not (*Attrition_flag*); card level (*Card_category*); banking relation duration (*Months_on_book*); number of products acquired by the client (Total_relationship_count); inactivity (*Months_Inactive_12_mon*); contacts (*Contacts_Co unt_12_mon*); credit limit (*Credit_limit*); balance (*Total_revolving_balance*); openness to buy (*Avg_ Open_To_Buy*); change in transaction amount (*Total_Amt_Chng_Q4_Q1*); total transaction amount (*Total_Trans_Amt*); total transaction count (*Total_Trans_Ct*); change in transaction count (*Total_Ct_Chng_Q4_Q1*) and average card utilisation ratio (*Avg_Utilization_Ratio*).

CLIENTNUM	Attrition_	Customer	Gender	Dependen	Educatior	Marital_S	Income_C	Card_Cate	Months_c	Total_Reli	Months_I	Contacts_	Credit_Lin	Total_Rev	Avg_Open	Total_Amt	Total_Tra	Total_Tra	Total_Ct_	Avg_Utiliz
7.69E+08	Existing C	45	M	3	High Scho	Married	$60K - $8	Blue	39	5	1	3	12691	777	11914	1.335	1144	42	1.625	0.061
8.19E+08	Existing C	49	F	5	Graduate	Single	Less than	Blue	44	6	1	2	8256	864	7392	1.541	1291	33	3.714	0.105
7.14E+08	Existing C	51	M	3	Graduate	Married	$80K - $1	Blue	36	4	1	0	3418	0	3418	2.594	1887	20	2.333	0
7.7E+08	Existing C	40	F	4	High Scho	Unknown	Less than	Blue	34	3	4	1	3313	2517	796	1.405	1171	20	2.333	0.76
7.09E+08	Existing C	40	M	3	Uneducat	Married	$60K - $8	Blue	21	5	1	0	4716	0	4716	2.175	816	28	2.5	0
7.13E+08	Existing C	44	M	2	Graduate	Married	$40K - $6	Blue	36	3	1	2	4010	1247	2763	1.376	1088	24	0.846	0.311
8.1E+08	Existing C	51	M	4	Unknown	Married	$120K +	Gold	46	6	1	3	34516	2264	32252	1.975	1330	31	0.722	0.066
8.19E+08	Existing C	32	M	0	High Scho	Unknown	$60K - $8	Silver	27	2	2	2	29081	1396	27685	2.204	1538	36	0.714	0.048
7.11E+08	Existing C	37	M	3	Uneducat	Single	$60K - $8	Blue	36	5	2	0	22352	2517	19835	3.355	1350	24	1.182	0.113
7.2E+08	Existing C	48	M	2	Graduate	Single	$80K - $1	Blue	36	6	3	3	11656	1677	9979	1.524	1441	32	0.882	0.144
7.09E+08	Existing C	42	M	5	Uneducat	Unknown	$120K +	Blue	31	5	3	2	6748	1467	5281	0.831	1201	42	0.68	0.217
7.11E+08	Existing C	65	M	1	Unknown	Married	$40K - $6	Blue	54	6	2	3	9095	1587	7508	1.433	1314	26	1.364	0.174
7.11E+08	Existing C	56	M	1	College	Single	$80K - $1	Blue	36	3	6	0	11751	0	11751	3.397	1539	17	3.25	0
8.16E+08	Existing C	35	M	3	Graduate	Unknown	$60K - $8	Blue	30	5	1	3	8547	1666	6881	1.163	1311	33	2	0.195
7.12E+08	Existing C	57	F	2	Graduate	Married	Less than	Blue	48	5	2	2	2436	680	1756	1.19	1570	29	0.611	0.279
7.15E+08	Existing C	44	M	4	Unknown	Unknown	$80K - $1	Blue	37	5	1	2	4234	972	3262	1.707	1348	27	1.7	0.23
7.1E+08	Existing C	48	M	4	Post-Grac	Single	$80K - $1	Blue	36	6	2	3	30367	2362	28005	1.708	1671	27	0.929	0.078
7.53E+08	Existing C	41	M	3	Unknown	Married	$80K - $1	Blue	34	4	4	1	13535	1291	12244	0.653	1028	21	1.625	0.095
8.06E+08	Existing C	61	M	1	High Scho	Married	$40K - $6	Blue	56	2	2	3	3193	2517	676	1.831	1336	30	1.143	0.788

Figure 1: Data for first 20 customers of the dataset

2.3 Data Visualization

Let's start by looking at the **numerical features**. In order to better visualise them we can plot, for example, histograms or boxplots. These can be seen in further detail in section 5.1 of the appendix. From the *Customer_age* boxplot we can conclude, for instance, that the average customer age is approximately between 40 and 50 though there are exceptions (with possible values of around 25 or 75). On the other hand, histograms allow us to see other things a boxplot cannot. If we look at the *Credit_limit* histogram, we can see that there is a big spike from 0 to 5000. This is clearly the most common value. Until 30 000 the number of customers clearly decreases. However, there is a little spike once again at 35 000. We were starting to see what appeared to be a tendency up to that point. So, the fact that there is a spike there, may be indicative of something.

The same process can be done for the **categorical features**. We chose to present these features in a bar chart (which can be consulted in section 5.1 of the appendix). If we look at figure 8 which depicts the *Attrition_flag* distribution for different card categories, we can see that there are more accounts terminated for platinum credit cards than for the other categories.

These are all useful and interesting plots but probably the most meaningful and useful ones are the **correlation matrixes** (figures 13 and 14). These plots correspond to each numerical value a correlation coefficient that feature has with another feature. If that correlation value is very small, we usually say it is **statistically insignificant**. This means that there is not a strong correlation between these 2 features so one does not really impact the other.

In figure 13 we can see the p-values for the correlation between all of the numerical features. The entrances of the matrix which have no value (blank square) are like that because the p-value is low (smaller than 1%). We see, for example, that there there is a perfect correlation (p-value of 1) between a customer's willingness to buy and the credit limit. There is also a very high correlation between a customer's age and months on book, which makes sense, as older people probably will have been registered in the bank for more years.

2.4 Best Methods for Similar Datasets

Before trying to apply every existing method at random, we carried out a small research [5] for theses and papers which dealt with credit card data. One of the most known theses about credit card approval [6] says that 'we observed that Nonlinear SVM has performed better than ANN and linear SVM'. **Linear SVM** and artificial neural network (**ANN**) provided good results but **nonlinear SVM** performed even better in this case. Other works seem to reinforce this statement: 'among which SVM with polynomial kernel and random forest achieve the best result' [13].

When directly compared, SVM appears to perform better than **Logistic Regression** or **Naïve Bayes Classifier** [7] but other methods which pop up quite often in other works are the **Random Forest** and the **Gradient Boost Classifier** method: 'We got the highest accuracy of 90% from Gradient Boosting Classifier' [8]; 'The best classification model in the study is Random Forest, followed by KNN, and lastly NN' [9]; 'The results show that the Novel Xgboost Classifier seems to perform significantly better than Random Forest (RF) for credit card approval prediction in terms of accuracy' [10]; 'because of the simplicity of the Boruta-GBM model, it is convenient to intelligently

select high-quality customers for bank loans' [11] 'Confusion matrix analysis is in accordance to Decision Tree and Random Forest algorithm helping us attain an accuracy of 86%.'[12].

Methods such as **KNN** or **Decision Tree** appeared above but GBC and Random Forest (and their variants) are the most common. We now have a good starting point. We expect methods such as SVM, GBC; Random Forest or ANN to perform reasonably well given the fact that they yielded good results in similar works. However, we should have in mind that each dataset is different and methods which work quite well for one dataset may not work for others. Furthermore, there are many variants and add-ons to the methods which we can include to improve accuracy scores.

3 Methods Implementation and Results

This section of the report will go through every method used and briefly explain them. We will first talk about feature selection and then go into: classification methods, tree-based classification methods, support vector machine and neural networks.

3.1 Feature Selection

Feature selection is the process of selecting a subset of features to be used in training a machine learning model. Although feature selection does not engineer new features nor change any feature, it is of extreme importance to a machine learning model as it allows the model to focus on the features that relate to the task at hand and avoid wasted resources or missing leads on features that are essentially dead weight. Among the most important drawbacks of a bad or nonexistent feature selection are dimensionality problems when there are many features to few observations; over-fitting due to redundant features, unnecessary training time and interpretability problems.

In our case, we proceeded to use unsupervised/manual feature selection techniques along with some supervised wrappers and embedded techniques. Firstly, from the observation of figure 13 we decided to drop two features "*Avg_Open_To_Buy*" and "*Total_Trans_Ct*" for being ambiguous with other features. Furthermore, we tested the option of dropping all the features with a variance lower than 0.05 but it yielded worse results compared to forward feature selection so it was abandoned.

We also applied a supervised wrapper technique, **Forward selection**, to the Logistic Regression model improving the model by a few percentile points, reaching the 90% hit rate, and simultaneously making the model simpler and faster to process. Other linear models were improved with a variant of this method, **backwards selection**. At last, the effect of **Lasso selection** on a 10 fold cross validation split was tested. The results were good although some of the simpler methods still showed better results.

3.2 Classification Methods

In a classification method, our goal is to build a function that takes as input the feature(s) vector and predicts something. In our case, the prediction we want to obtain is binary: will the customer terminate the card (card churning) or not? We start by dividing the data into training and testing data with a ratio of, respectively, 70%/30%. With this done we decided to apply the following classifica-

tion methods: Logistic Regression, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Gaussian Naive Bayes and KNN.

For each of the methods, accuracy scores were calculated and the ROC curves were plotted (allows us to calculate AUC) - figure 2. The LDA method appears to be the best model followed closely by the logistic regression model. The QDA, Naive Bayes and KNN models are not that satisfactory.

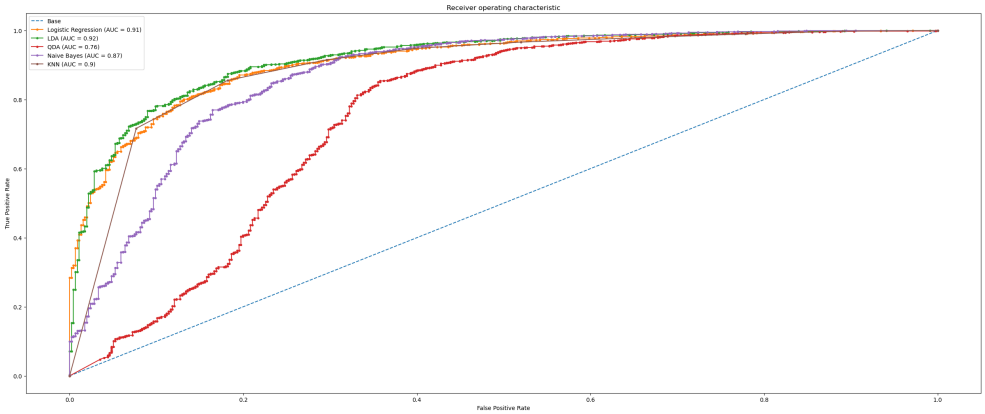

Figure 2: ROC for the implemented classification methods

At a first glance. We also performed a **10 fold cross validation** to every method (figure 16). This graph is very conclusive as it allows us to see what is likely an anomaly in the data, which is common to every method. In the fourth division of the 10 folds, we see that the mean squared error (MSE) is much higher than the others which on average have practically the same value.

3.2.1 Logistic Regression Model

This is a classification method characterised by its simplicity and is mostly applicable to binary and linear problems. In our problem, we expect a binary outcome so this model is adequate. We verify that the accuracy score is 89.67% and that it has an AUC (area under the curve) value of 91%. Given the simplicity and computational efficiency of the model, this is not a bad result. Even more so when compared to other classification methods.

3.2.2 Linear Discriminant Analysis Model

The goal of this method is to maximise the variance between classes and minimise the variance inside the class itself. This is achieved with a linear discriminant function and based on the assumption that the data in each class can be described with a Gaussian probability function with the same variance. However, this approach can be quite general and other distributions can be used as well. For this method, we obtained an accuracy score is 90.36% and it has an AUC of 92%. We can see that the results are extremely similar to the logistic regression model but are still slightly far away from the accuracy we are striving for. LDA models can prove a better option if the classes are well separated, cases in which the logistic regression model estimates become unstable.

3.2.3 Quadratic Discriminant Analysis Model

The QDA model is very similar to the LDA but we estimate an individual covariance matrix for each observation class. It is particularly useful when there is previous knowledge of which classes exhibit different covariances. When applied to our problem, we see that it does not apply well at all (accuracy score of 86.41% and it has an AUC of 76%).

3.2.4 Gaussian Naive Bayes Model

This is a classification method based on Bayes theorem and the assumption of independence between predictors, that is, a Naive Bayes classifier, assumes the presence of a certain characteristic inside a class is not related to any other characteristic. This method is easy to assemble and particularly useful for very big datasets. Applied to our problem, we verified an accuracy score of 89.41% and a AUC of 87%. These are decent values but still not better than LDA and logistic regression.

3.2.5 KNN Model

The KNN - K-nearest neighbour, is a method in which the input consists of the k closest training examples in a data set. This means that we assume all of the data points can be grouped by class (each class has data points with similar values). With the purpose of determining the best value for k, we implemented a cycle in which for each k from 1 to 100, the accuracy scores were calculated. With that, we were able to plot this data (figure 15) and conclude that the best value for k is 7. Increasing the k value further than 7 does not increase the accuracy. With the value of 7, we obtained an accuracy score of 89.63% and a AUC of 90%. These are decent values but still not better than LDA and logistic regression.

3.3 Tree-based Classification Methods

The operating principle of tree-based methods consists of stratifying or segmenting the predictor space into a number of simple regions. These regions are called nodes and will be used for the decision-making of the model. In each node, the data will be further stratified, optimising a certain criterion. In this case, the optimisation criterion is the Gini index, which is a measure of the total variance of the data in a node. So, it functions as a measure of node purity, since a small value indicates that a node contains predominantly observations from a single class, meaning that the data is well segmented. It is considered a top-down approach because it begins at the top of the tree and then successively splits the predictor space. At each split, two new branches are created, and this is made until each terminal node (leave) has fewer than some minimum number of observations.

Random forests are another approach to tree-based methods. Its principle is to grow multiple decision decorrelated trees. Each time a split in a tree is executed, a random selection of predictors is chosen. The predictions of the trees will then be merged together, for a more accurate prediction. The final prediction result will be the most voted one by the independent trees.

In this project, boosting was also used. This method creates copies of the training data, fitting a separate decision tree to each copy. The trees are trained sequentially, with each model trying to compensate for the weaknesses of its predecessor, so it uses information from the previous trees. In the end, a single predictive model is obtained. The data was also divided into a training set containing 70% of the data and a test set (the other 30%). Various decision tree classifier models

were built with the same training and test sets, to see the effect that varying the depth of the three models would have on the accuracy of the model. The table below represents this.

Depth	Accuracy (%)	Improvement (%)
2	89.04	-
4	91.81	3.1
6	94.01	2.4
8	94.34	0.4
10	94.18	-0.2

Table 1: Accuracy improvement with tree model depth

As we can observe, there is an improvement of the model accuracy with the increase of the depth, but reaching a certain point, the increase in the depth can actually make the model worse. It is interesting to see that even with a depth of 2, the accuracy is quite high.

For the random forest model, we used 200 estimators and the results were excellent. The mean ROC curve for this model approaches a perfect curve, with a mean AUC of 0.98.

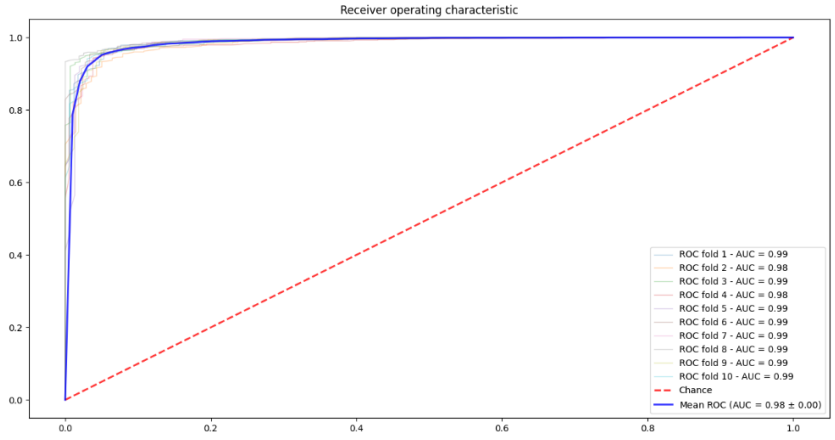

Figure 3: ROC curves for the Random Forest Model

Finally, the obtained accuracy score for the boosting model was 97.30%.

The variable importance for each model was studied. It was verified that with the increase of the depth in the simple trees, the number of variables that contributed to the construction of the model increased. This is expected since with a tree with more depth, there will be more nodes, so the number of variables in the decision that further splits the tree will be higher.

Through all the models, a pattern appeared. The most important variables were almost all the same. Things like total transaction amount and total balance were highly relevant. This makes sense since a customer that will cancel a bank account probably doesn't use it much, so the total transaction amount will likely be low. Also, if a client plans on closing an account, it will cash it out before the closing, so it makes sense that the total balance variable is important for the construction of the model. The variable importance for the random forest model is represented in the image below.

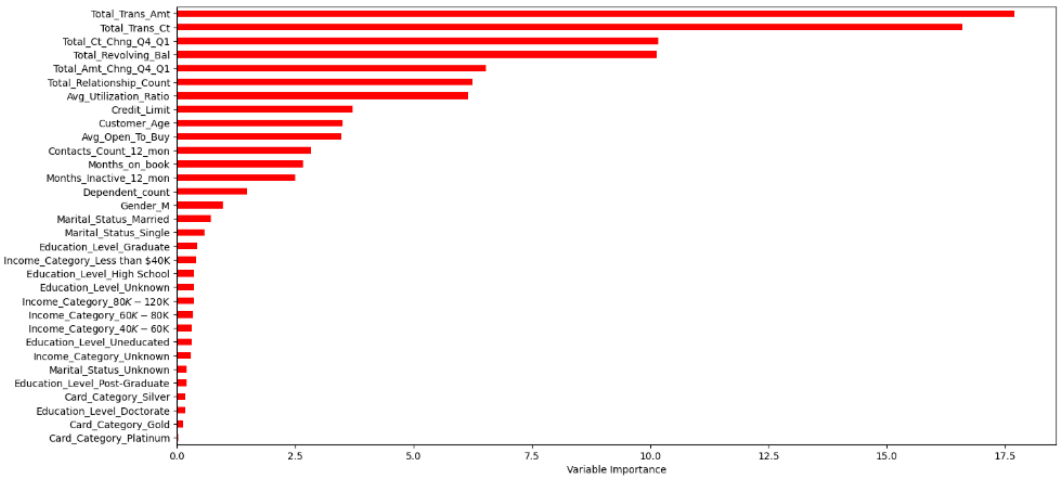

Figure 4: Variable importance for the random forest model

3.4 Support Vector Machine

In simple terms, a Support Vector Machine is a model used for both classification and regression problems. These can be linear or non-linear and it is based on creating hyperplanes that separate data into different groups. If the data has 3 inputs and one output, it can be represented in a 3D plot and the hyperplane will be a regular 2D plane on a 3D space. In our case, we will be using a lot more than 3 inputs, therefore it isn't possible to represent the groups created by the SVM in a plot. The functions that classify the data points into one group or another are called kernels and changing the used kernel can have significant implications on the obtained results.

On a first look into SVMs, all features were used at first to fit a basic model with a linear kernel and an arbitrarily chosen value for the penalty parameter C. As this was an expensive task when it comes to computational resources, a Principal Component Analysis was performed before trying other kernels and parameters. The analysed kernels are the linear, polynomial, radial basis function (*rbf* for short) and sigmoid. In order to pick the number of principal components to use for the following SVM methods, the explained variance was plotted as can be seen in figure 19 (though it is not clear what number of principal components should be chosen). In hindsight, choosing 21 or 22 principal components could be the better option, due to the falloff in explained variance when more are considered. Another choice option is to set a threshold of cumulative explained variance (e.g. 60%) and opt for the number of principal components that satisfies that cumulative explained variance (see figure 20). To provide more information on how many components should be used, the accuracy score of a linear kernel SVM with parameter $C = 1$ was measured for every number of principal components available (32). These scores are presented in figure 21.

The number of principal components that achieved the highest accuracy score was used for the remaining work performed on SVMs. Due to the high number of components used, the 2D plot of the SVM is irrelevant as there are no obvious lines that split the data into existing customers and churned customers. To try and achieve the best results for each of the kernels studied, a grid search cross validation was used where the penalty parameter C could take one of the multiple values listed below and the γ parameter was also one of the possible values listed below:

- List of C : $[0.1, 1, 10, 100]$
- List of γ : $[0.0001, 0.001, 0.01, 0.1, 1, 10]$

The results for the linear kernel SVM are presented in figure 22. Note that the linear kernel doesn't use the γ parameter. For the polynomial kernel SVM and for the other kernels, with the linear one as the exception, the grid search cross validation was done across both parameters. In order to not make the computation times even larger, there was no iteration on the degree of the polynomial kernel and because the data isn't overly complex the default value of 3 should be enough. The same can be said for the coeff0 parameter available using the *sklearn* module, as the default value of 0 is usually fine for most cases according to multiple sources.

The results for the polynomial kernel SVM can be checked in figure 23. The *RBF* kernel is the most popular kernel used in SVMs, both in papers and in online help forums. Again, a grid search cross validation was done to calculate which parameter values would be optimal. The results are shown in Appendix C.2 [5.4] in figure 24. The sigmoid kernel also uses the parameter *coeff0*, but like with the polynomial kernel, it was left to its default value 0 to compute the algorithms faster. The results from this kernel can be seen in figure 25. Below is a figure with the ROC curves plotted for each studied kernel with the best parameters with the respective AUC scores as well.

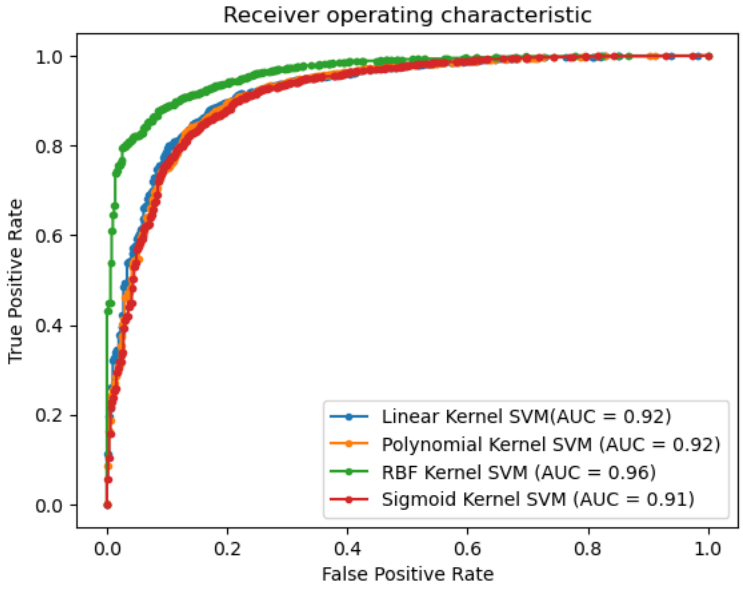

Figure 5: ROC curves for different kernels and their respective AUC scores

The AUC scores corroborate our findings in some papers which claim that for most cases the RBF kernel is the best performant.

3.5 Neural Networks

Finally, we will implement neural network methods. These are machine learning algorithms that are based on the learning mechanisms of human brains, mimicking the way that biological neurons connect and communicate. The knowledge of the network is acquired from the input, which would

be the surrounding environment in the case of humans, and the connection intensities between the neurons are called weights, which are saved (represent the acquired knowledge of the network).

In the beginning, the weights are small and random. The input is feed to the network, in the input layer, which passes the data to the rest of the network. The data is passed through the hidden layers until it reaches the output layer, which holds the calculated result or the output of the problem. An error is calculated and the weights are adjusted until reaching the desired number of iterations, which are called epochs.

The data was divided by the following manner: 70% for training, 20% for validation and 10% for testing. Like for all the other models, the training data set is used to fit the model and the test data set is used for the evaluation of the final model. In neural networks, the validation data set is used to evaluate the model during its training, while tuning its parameters.

There are two main aspects when working with neural networks, the hidden layers and the activation functions. Since the problem in hand is a binary classification problem, the sigmoid activation function was chosen for the output layer as it is usually the best suited for this kind of problems. On the other hand, for the hidden layers, the choice was to use *ReLu* functions because it yields better results than other functions. One thing to note is that the *dying of ReLu* problem was not explored and there is therefore room for improvement (for example choosing a leaky *ReLu* could have fixed such an issue).

As a rule of thumb, a first model with only one hidden layer was built, with this layer having 16 neurons (which is about the mean of the number of inputs and outputs). While this model presented good accuracy values ($> 90\%$), the AUC scores were a bit lacklustre, dropping down to around 82% for the test data. With this model there was little to no overfit as the validation loss didn't stray to far away from the training loss. For a second model, a second hidden layer was added, with half the neurons of the first hidden layer (16). The accuracy score was still above the 90% threshold but the AUC score for the test data dropped to 80%.

The results for both of these models are plotted in figures 26 and 27. Besides the very similar results, which by itself indicates that one hidden layer is enough and adding more doesn't come with benefits, one can see that the model with the two hidden layers has a more pronounced overfit than the first model with only one hidden layer.

However, by tuning the parameters, it could be possible that a model with two hidden layers would perform better. To test this hypothesis, different number of nodes per hidden layer were tried. The best models ended up being the ones with less nodes per layer, (figure 28). The learning rate was also changed keeping the same number of nodes across the board (best estimate is 4 nodes for the first layer and 2 nodes for the second layer). In figure 29 we can see quite a few phenomenons:

- The model with the fastest learning rate (0.01) doesn't perform very well, with a loss that is greater than the other models' loss and suffers from overfitting almost instantly;
- The model with the second slowest learning rate isn't very good as it also suffers from overfitting before reaching 250 epochs;
- The lowest learning rate makes it so that the respective model requires lots of iterations before converging;

This gives an optimal model with learning rate of 0.0001 (doesn't suffer from overfit at all).

Applying the same parameters to a 1-hidden layer model provides a very similar, marginally better model. Both models (1-hidden layer and the 2-hidden layers) have their losses represented in figures 30 and 31, respectively. Their scores were the following:

	1 hidden layer		2 hidden layers	
	Accuracy	AUC	Accuracy	AUC
Training	92.42%	83.27%	92.82%	83.92%
Validation	91.42%	79.88%	91.67%	81.03%
Test	89.65%	79.08%	90.23%	79.67%

Table 2: Neural network models performance scores for different number of hidden layer

4 Conclusion

With the execution of this project, we were able to apply the studied methods along the course to real data and tried to conclude which of them are the best for our case (predicting if a client will cancel a credit card). The relevance of such a thing for a bank is obvious. This is an example of the many use cases and importance of predictive models, which have a wide range of applications.

Looking at the data and after getting a grasp of what it consisted of, there were already some hints about how low complexity our model could possibly have. That hypothesis was validated when we concluded that the best overall method for our data, and for our prediction variable was the random forest. This method not only fitted the data almost perfectly, but also didn't require much computational power. While some other methods, namely the SVM, also achieved good accuracy and AUC scores, they required much greater computational power in comparison.

Good results were obtained right from the start with simple classification methods such as the Logistic Regression and K-Nearest Neighbours which is why we were fairly surprised about the poor performance of the neural networks, especially considering the latter would take several more orders of magnitude of time to compute than the simpler methods.

We have also noticed that many of the best-performing methods for our problem were exactly those referenced by other works with similar types of data. From the research work carried out in section 2.4, we were expecting Random Forest, Gradient Boost and SVM to be the best performing. And this is exactly what happened! Random Forest achieved accuracy scores of up to 99%, followed by Gradient Boosting with 97% and in the final spot of the podium SVM with 92% (though this one required much more computational time).

Overall, there are not many doubts that the tree-based classification methods were the better option for our problem. Despite this, the linear classification methods we used proved to have decent enough accuracy scores with very little complexity of the models. We believe our problem was simple and easy to classify, thus the reason for these classification methods yielding good results. However, for a more complex problem, what we could probably expect, would be for these methods to reduce their efficacy and for other, more complex, such as the SVM and ANN to perform better.

References

[1] Credit card dataset used for the project

[2] Project Github Repository

[3] Course Page (project statement and course slides)

[4] Credit card statistics

[5] Google Scholar

[6] PEIRIS, M. P. C. Credit Card Approval Prediction by Using Machine Learning Techniques. 2022. PhD Thesis.

[7] ZHAO, Yiran. Credit Card Approval Predictions Using Logistic Regression, Linear SVM and Naïve Bayes Classifier. In: 2022 International Conference on Machine Learning and Knowledge Engineering (MLKE). IEEE, 2022. p. 207-211.

[8] DALSANIA, Naman; PUNATAR, Devang; KOTHARI, Deep. Credit Card Approval Prediction using Classification Algorithms.

[9] FLORES, Leonard, et al. A Classification Approach in the Probability of Credit Card Approval using Relief-Based Feature Selection. In: 2022 2nd Asian Conference on Innovation in Technology (ASIANCON). IEEE, 2022. p. 1-7.

[10] YASASVIA, Pathipati; KUMARB, S. Magesh. Improve Accuracy in Prediction of Credit Card Approval Using Novel XGboost Compared with Random Forest. 2022.

[11] CHEN, Ze; LIN, Geng; JIN, Xiuling. Credit Approval Prediction Based on Boruta-GBM Model. In: 2021 7th International Conference on Systems and Informatics (ICSAI). IEEE, 2021. p. 1-5.

[12] RAMACHANDRA, H. V., et al. Design and Simulation of Loan Approval Prediction Model using AWS Platform. In: 2021 International Conference on Emerging Smart Computing and Informatics (ESCI). IEEE, 2021. p. 53-56.

[13] FU, Zhoutong; LIU, Zhedi. Classifier Comparisons On Credit Approval Prediction.

5 Appendix

5.1 Appendix A - Data Visualisation

Appendix A.1 - Numerical Features' Charts

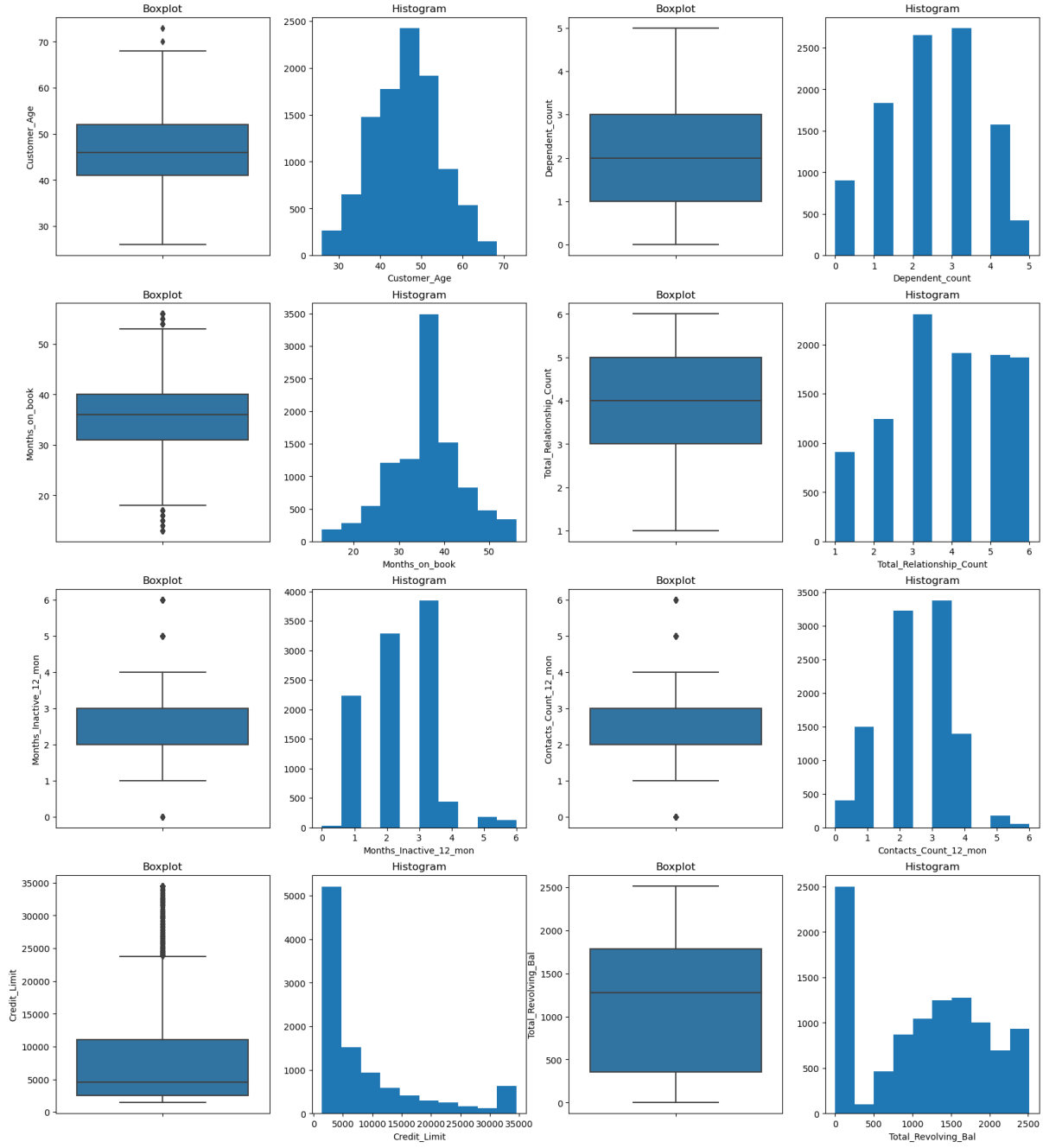

Figure 6: Histograms and boxplots for numerical features

.

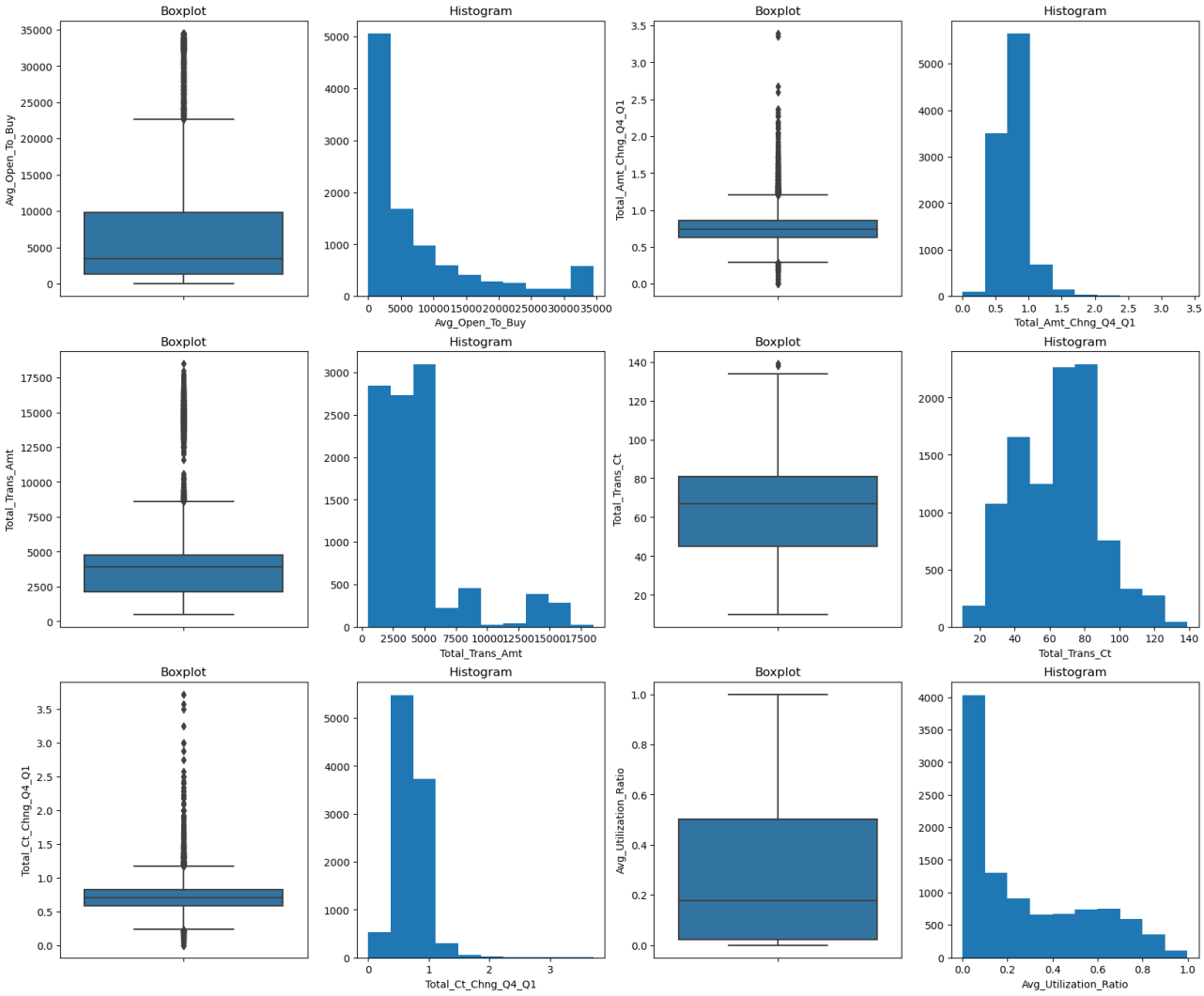

Figure 7: Histograms and boxplots for numerical features 2

Appendix A.2 - Categorical Features' Charts

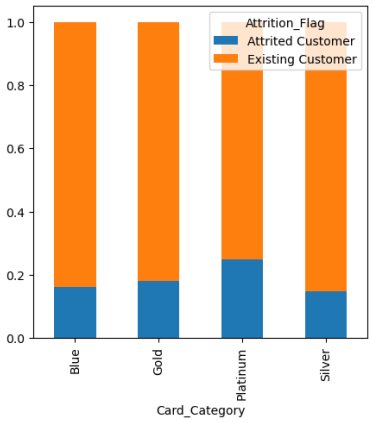

Figure 8: *Attrition_flag* values for different card categories

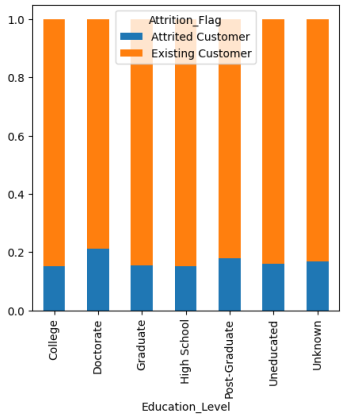

Figure 9: *Attrition_flag* values for different education levels

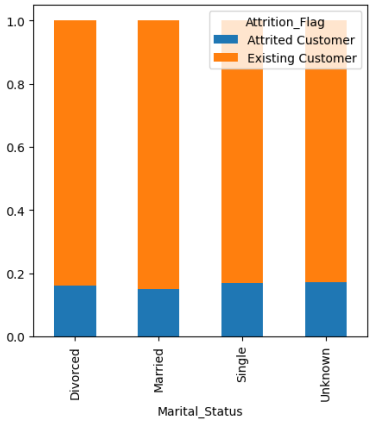

Figure 10: *Attrition_flag* values for different marital situations

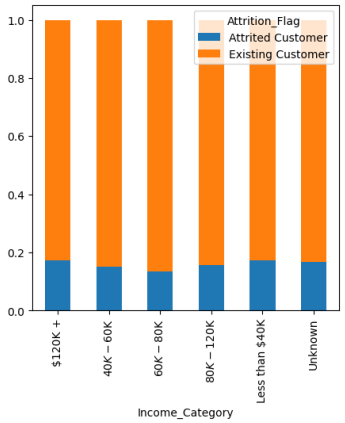

Figure 11: *Attrition_flag* values for different customers' incomes

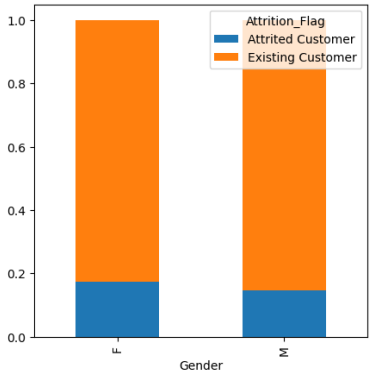

Figure 12: *Attrition_flag* values for the different genders

Appendix A.3 - Correlation Matrixes

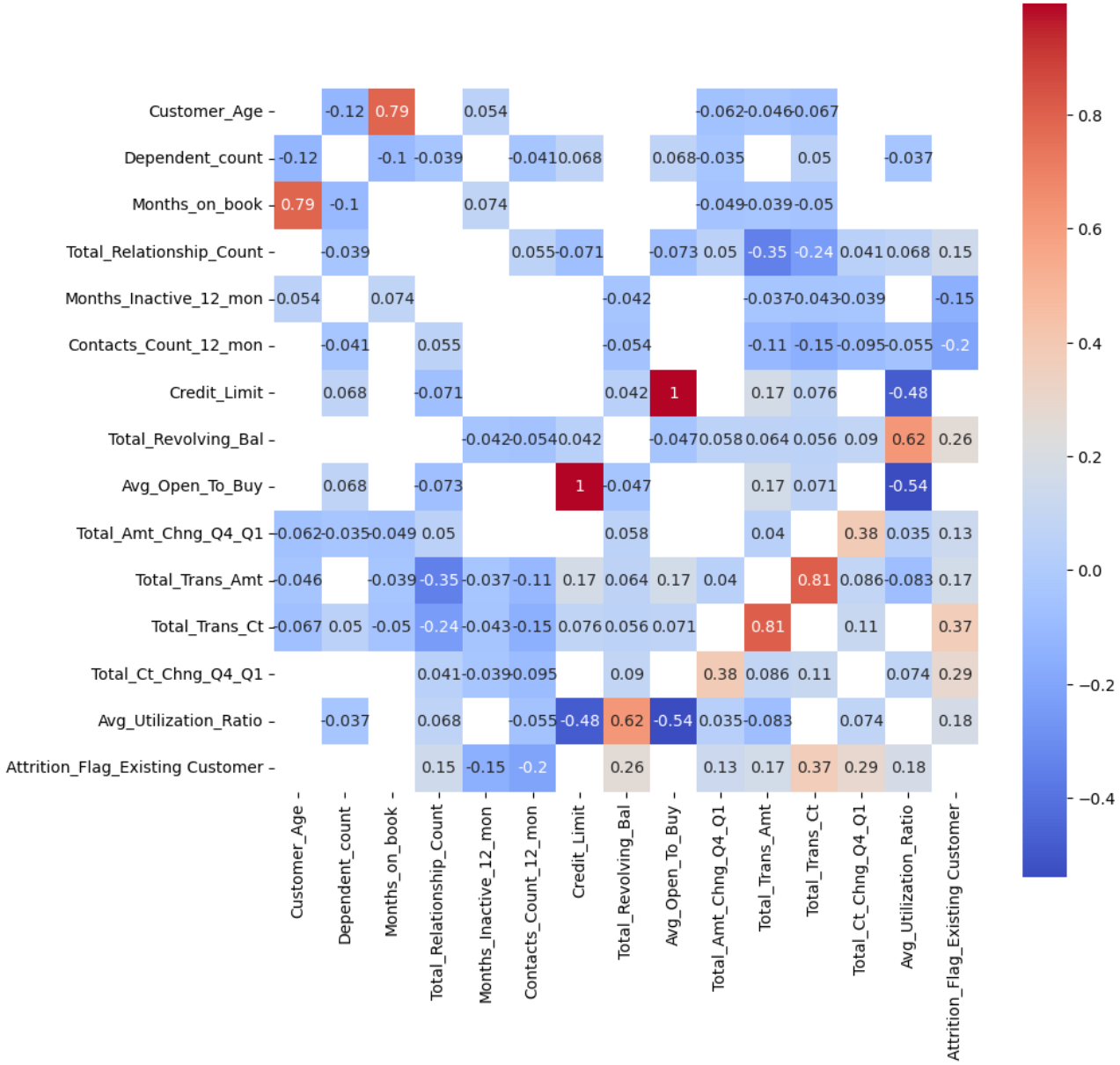

Figure 13: Correlation matrix for numerical features

.

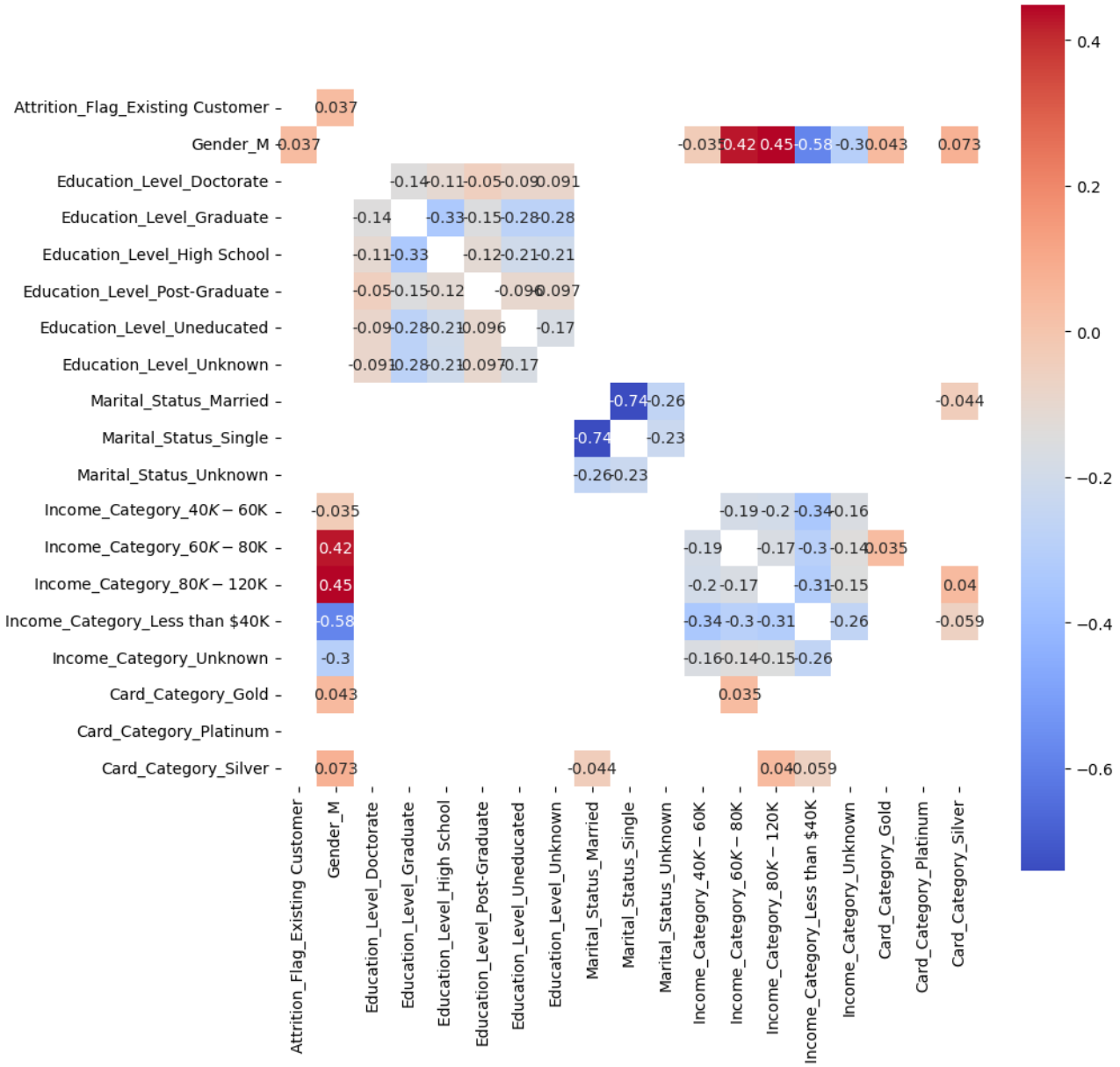

Figure 14: Correlation matrix for categorical values

5.2 Appendix B - Classification Methods

Appendix B.1 - KNN Accuracy score

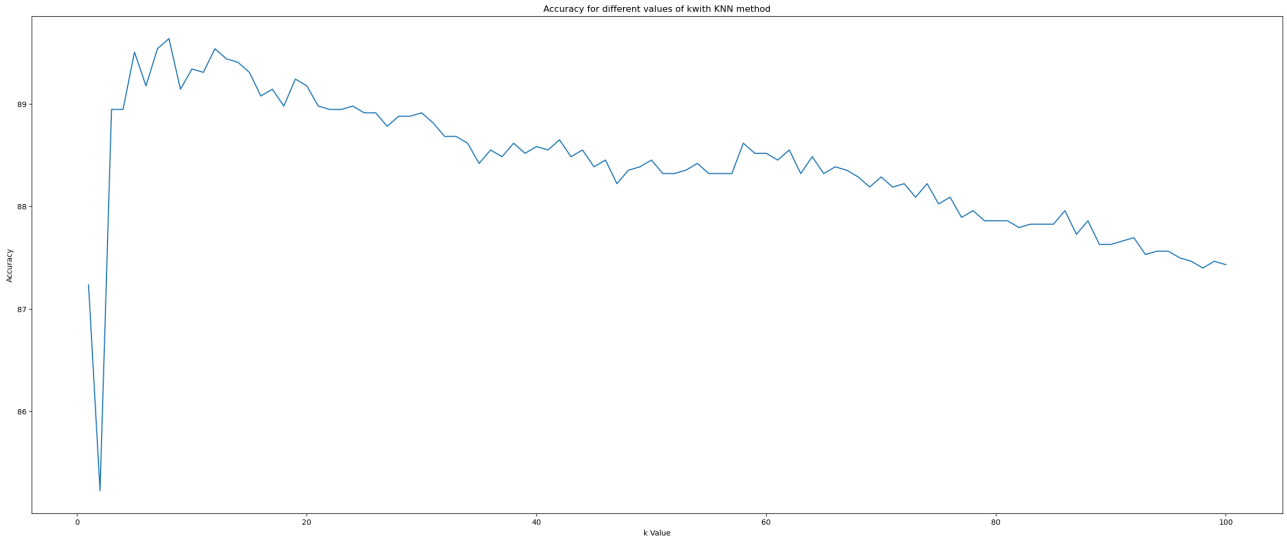

Figure 15: Accuracy score for different values of K

Appendix B.2 - 10-Fold Cross Validation

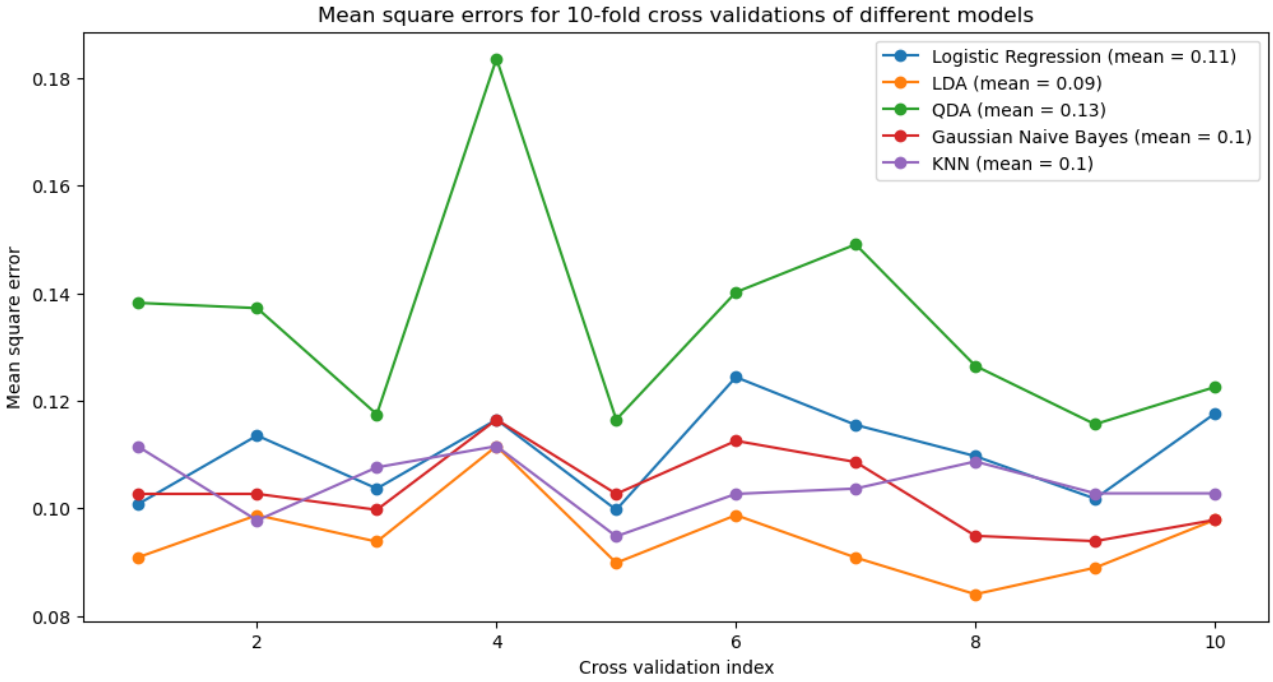

Figure 16: Mean square errors for 10-fold cross validations of different models

5.3 Appendix C - Decision Tree

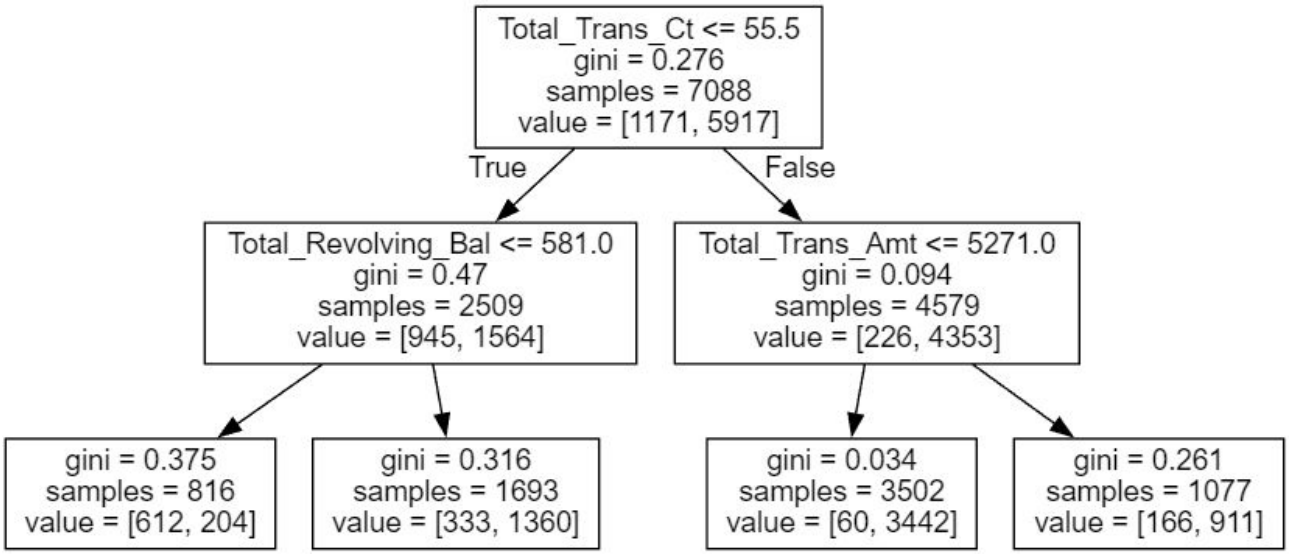

Figure 17: Decision tree with depth 2

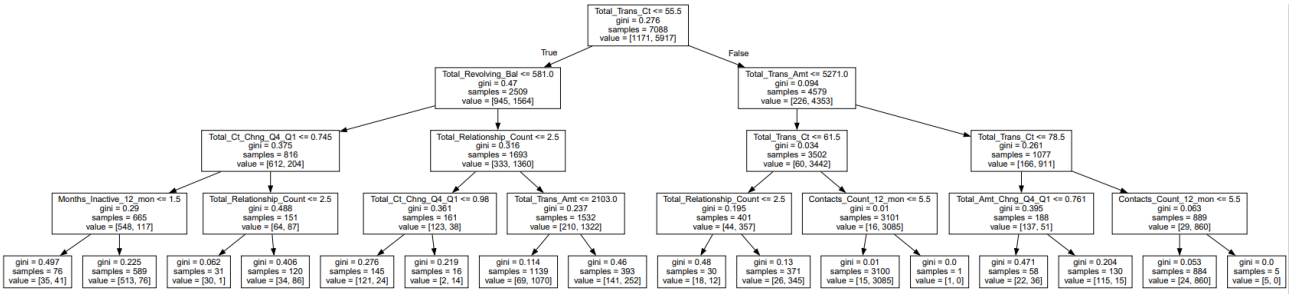

Figure 18: Decision tree with depth 4

5.4 Appendix D - Support Vector Machines

Appendix D.1 - Principal Component Analysis

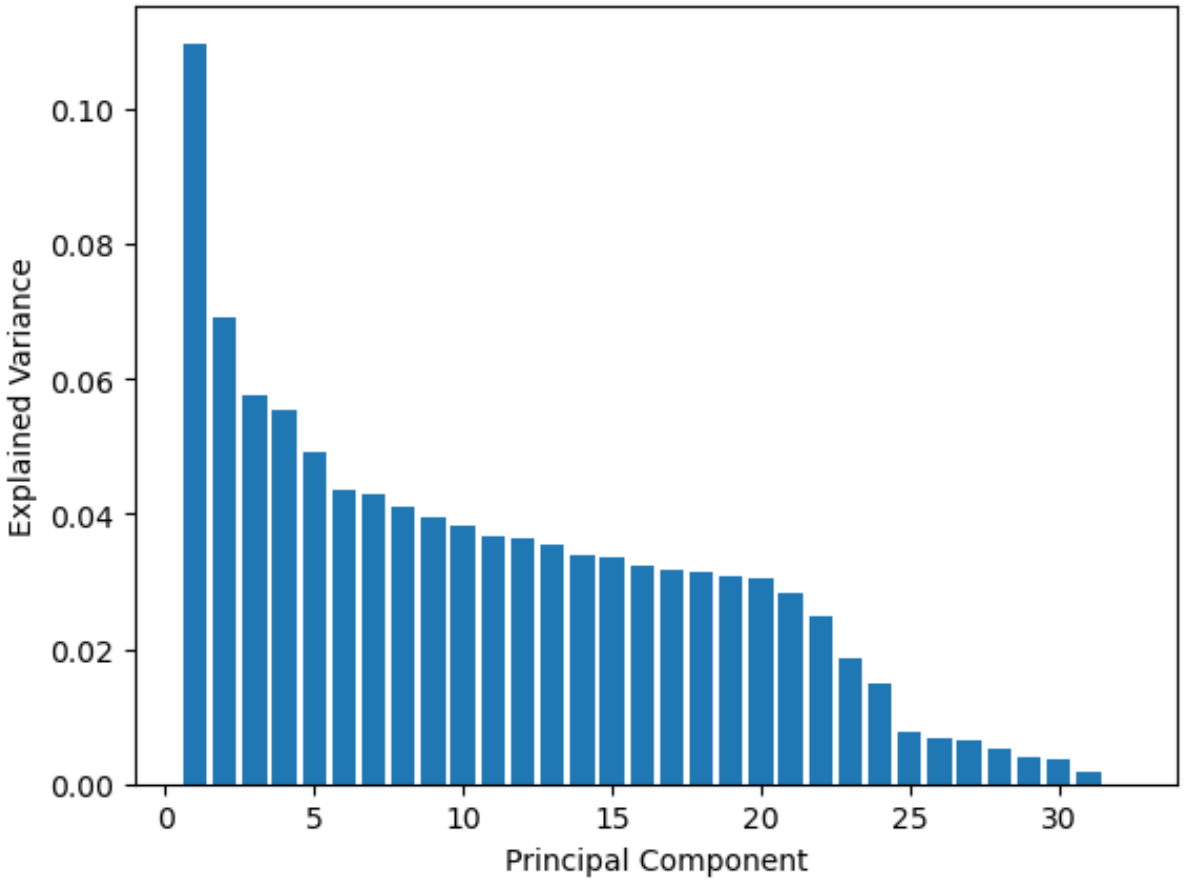

Figure 19: Explained variance for each principal component in PCA

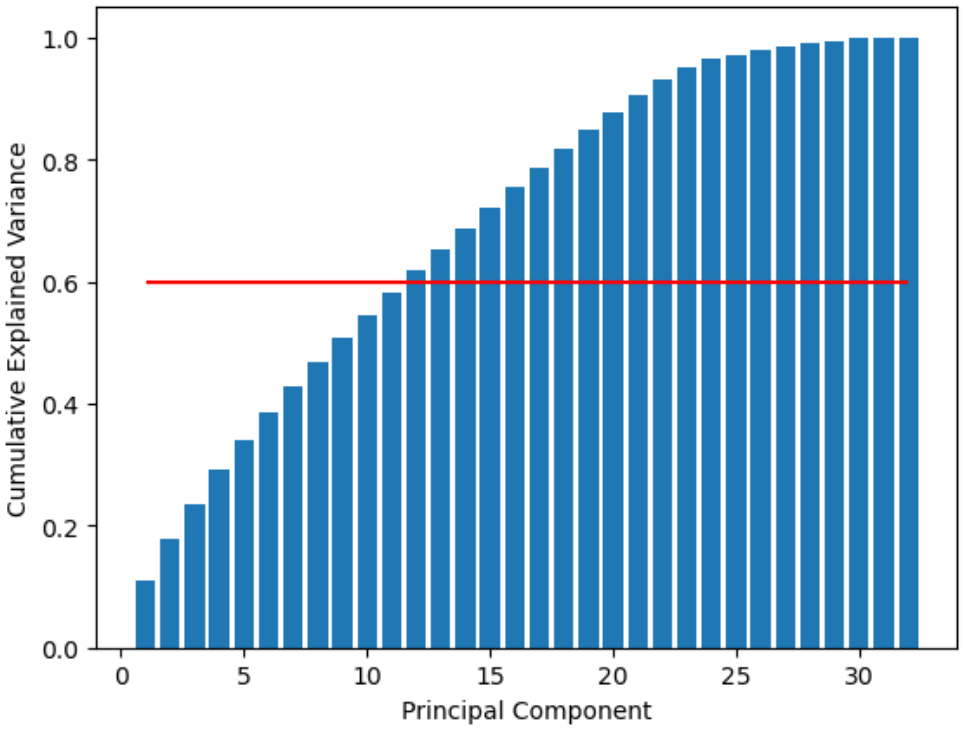

Figure 20: Cumulative explained variance for given number of principal component in PCA

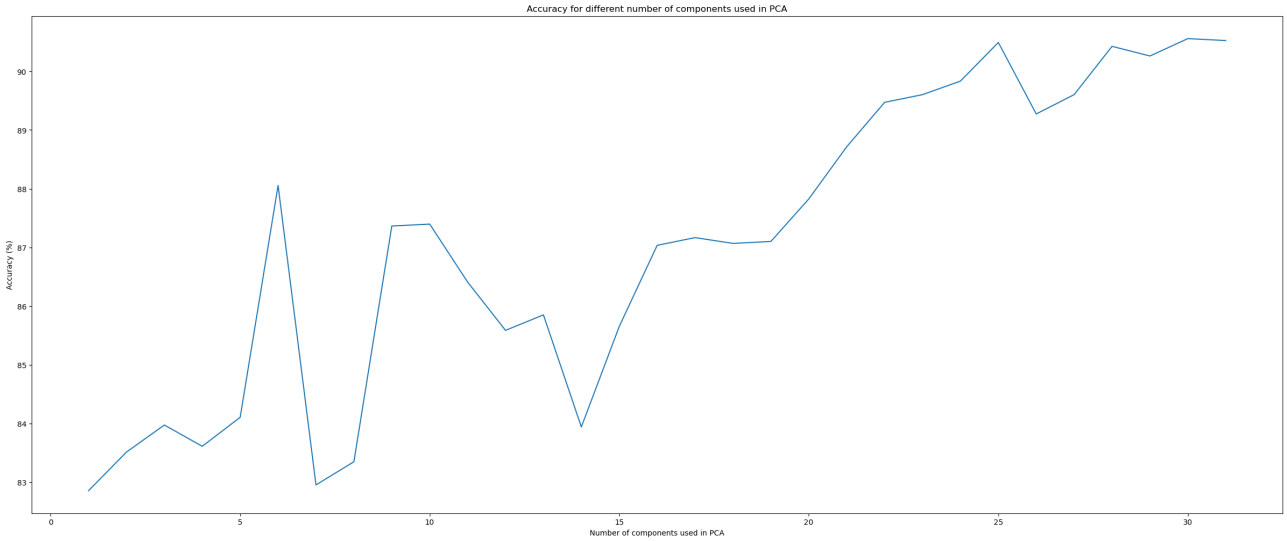

Figure 21: Testing accuracy as a function of the number of principal components used

Appendix D.2 - SVMs Mean Test Scores

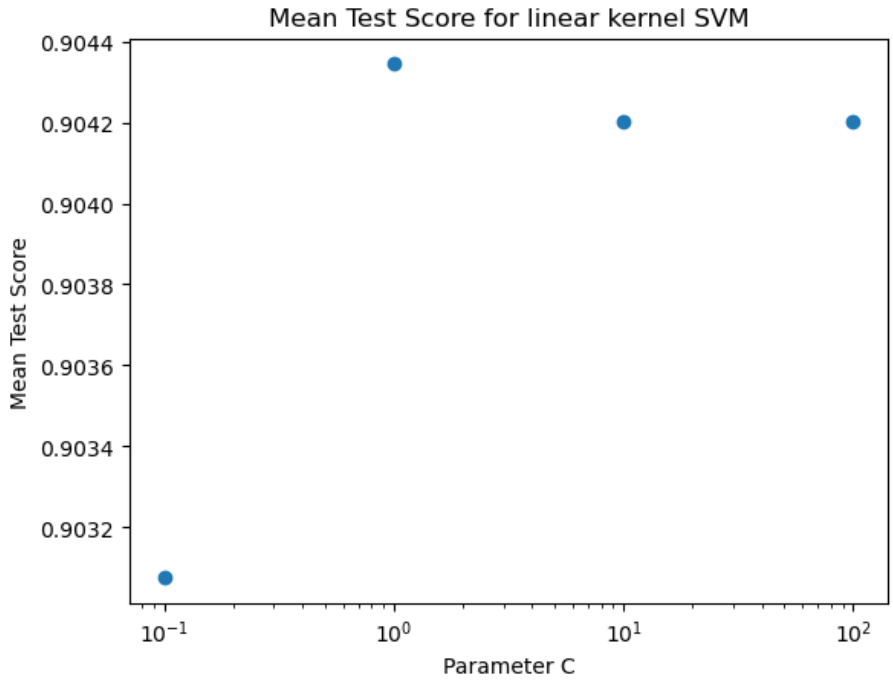

Figure 22: Mean test score for the linear kernel SVM as a function of the penalty parameter C

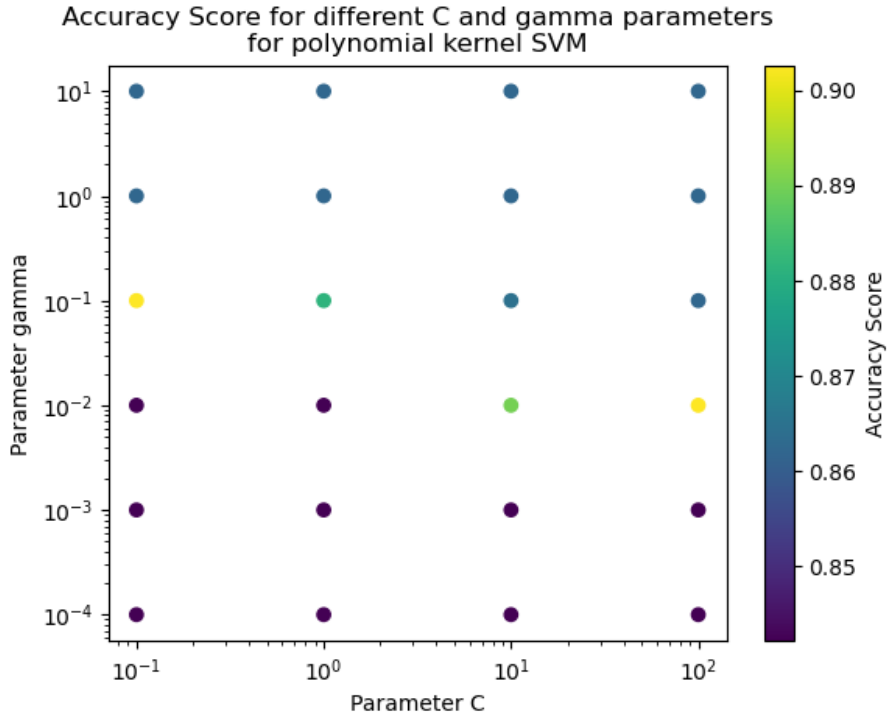

Figure 23: Mean test score for the polynomial kernel SVM as a function of the penalty parameter C and parameter γ

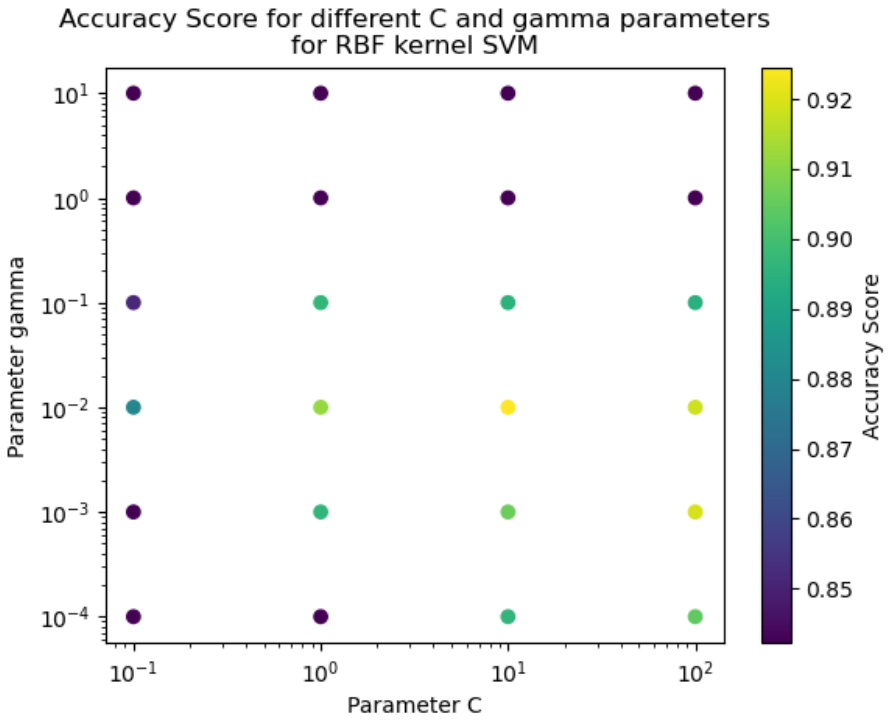

Figure 24: Mean test score for the RBF kernel SVM as a function of the penalty parameter C and parameter γ

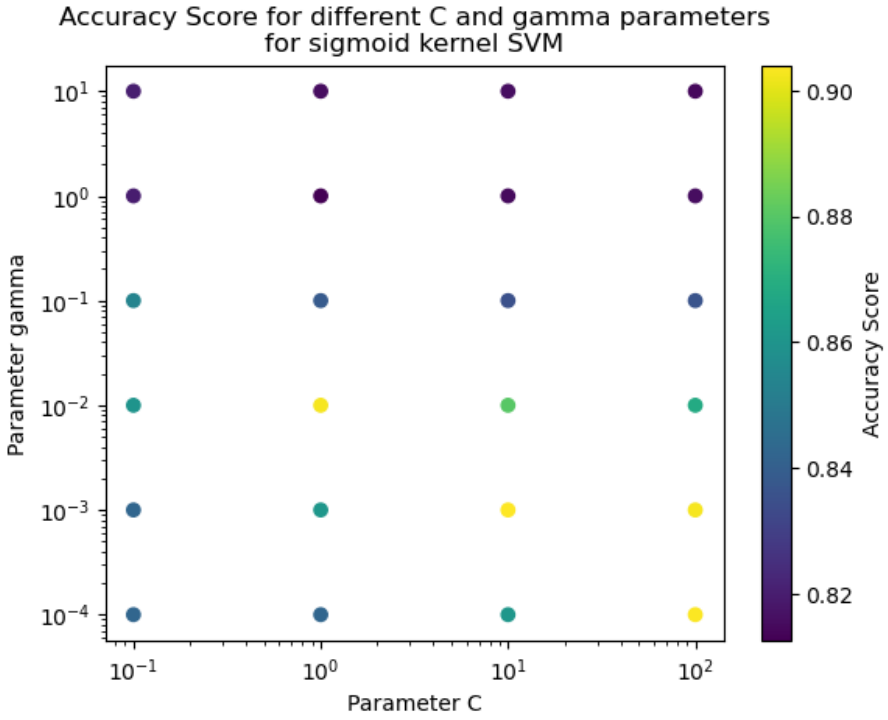

Figure 25: Mean test score for the sigmoid kernel SVM as a function of the penalty parameter C and parameter γ

5.5 Appendix E - Neural Networks

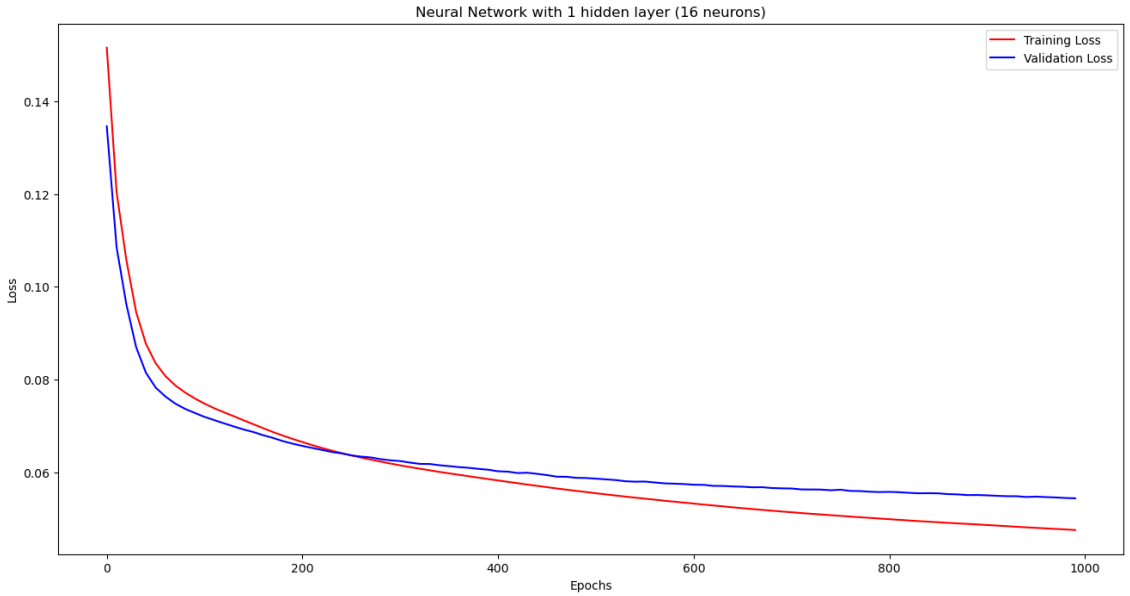

Figure 26: Training and Validation Losses for 1 hidden layer (16 nodes)

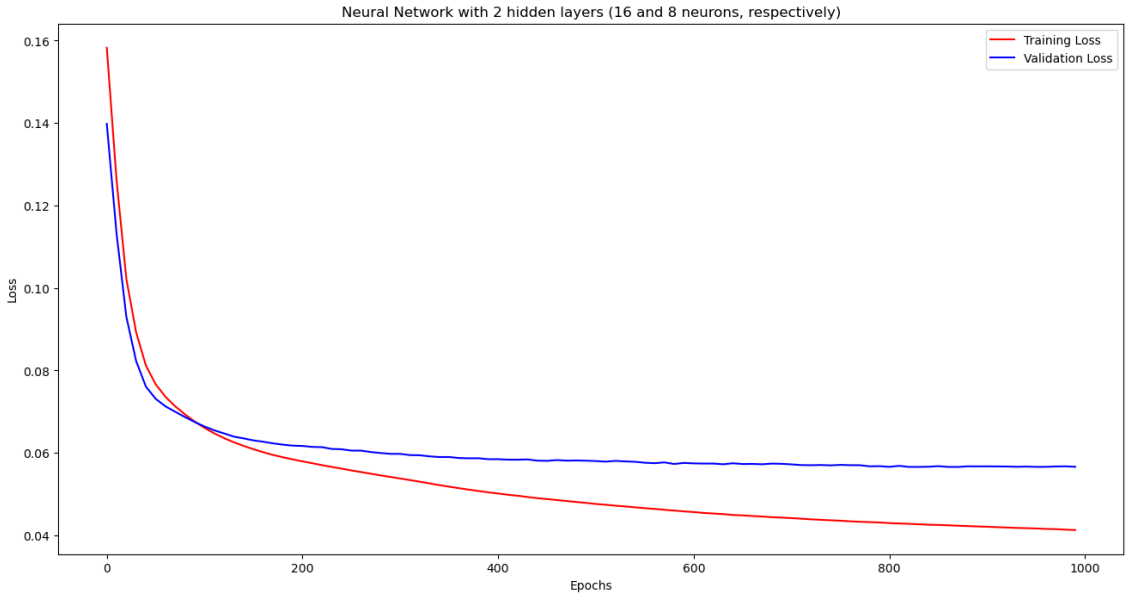

Figure 27: Training and validation losses for 2 hidden layers (16 nodes and 8 nodes respectively)

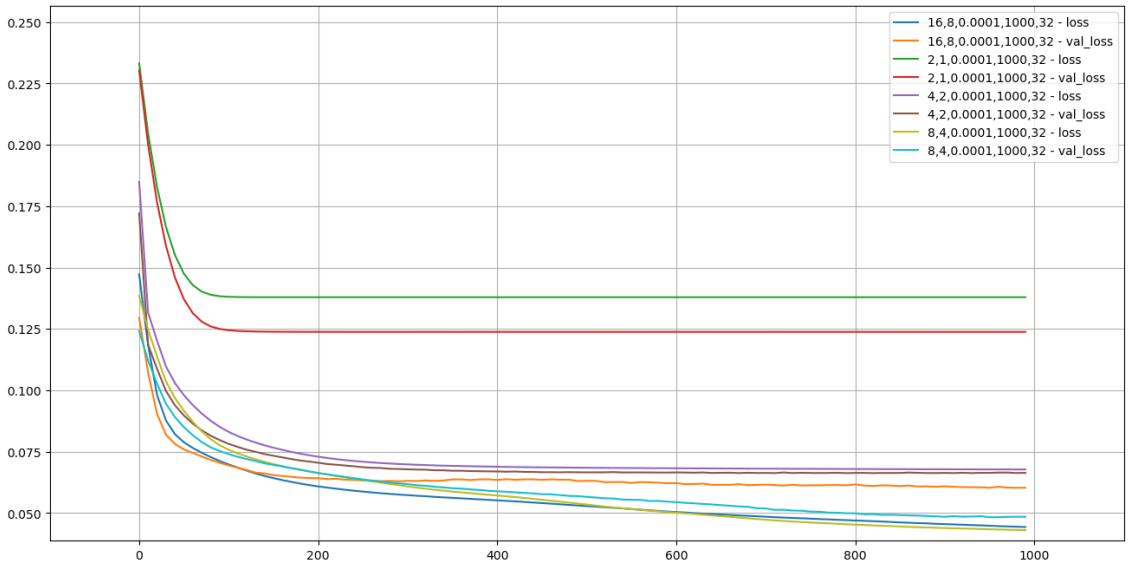

Figure 28: Number of nodes iteration

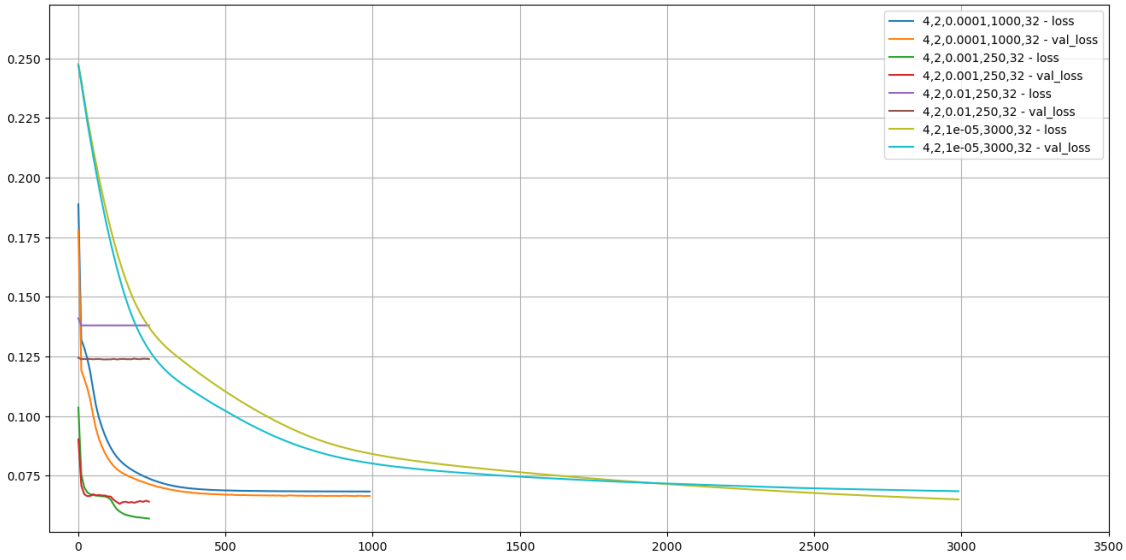

Figure 29: Learning rate iteration

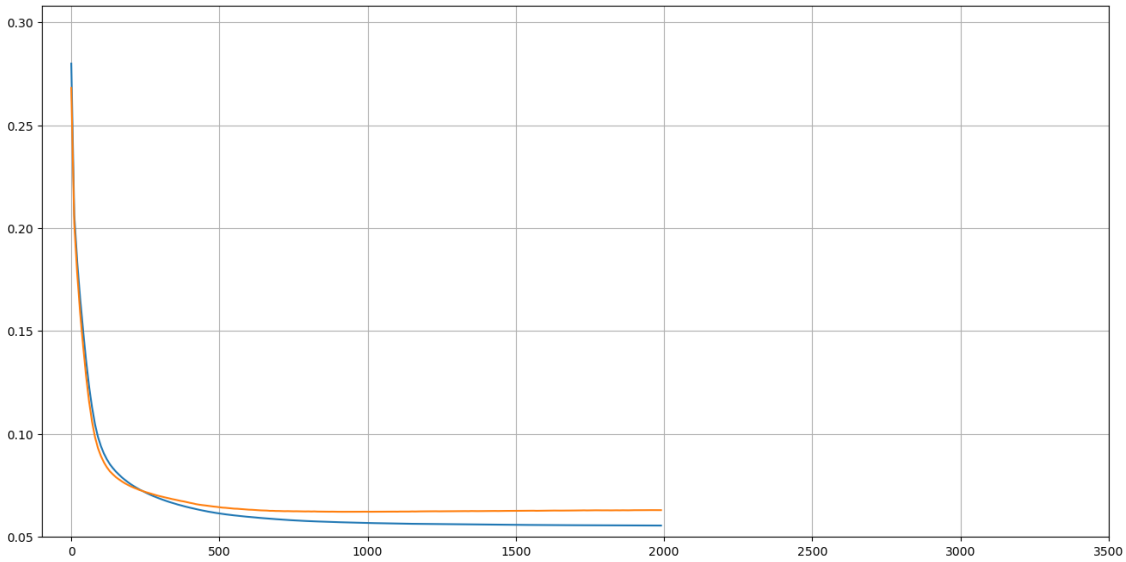

Figure 30: Best 1-hidden layer model

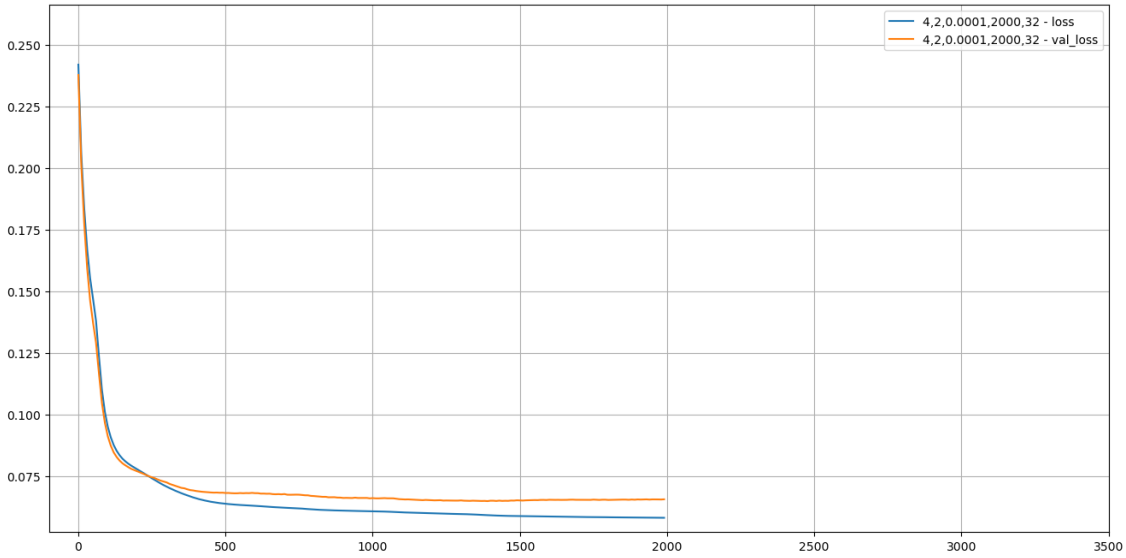

Figure 31: Best 2-hidden layers model