UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Classification of customers in the credit-G data set

Authors:
Mohana Fathollahi
Kaia G. Urdahl

June 17, 2021

Table of Contents

1. Introduction

The aim of this paper is to utilize different methods within supervised machine learning in order to classify customers as bad or good customers in the credit- g data set. Based on classification, banks will decide to pay the loan to good customers and not pay the loan to bad customers.
The methods that have been used in this paper are Generative and Discriminative models. For generative methods, LDA,QDA and Naive Bayes, and for discriminative models, K-NN, Logistic Regression, Decision tree family and Ensembles have been considered.

In this analysis it has been assumed that the reader has some knowledge of the machine learning topic, and therefore some basic concepts are assumed known. However, where it has been found suitable, or necessary for the reader to have an explanation the basic concepts are described briefly.

2. Related work

There exists an extensive amount of previous work completed on the credit-g data set. With the awareness of the already existing research on the data set, the proposed models and the purpose of this paper will be contextualized on the basis of methods used and focus on sensitivity.
Although some of the models presented in this analysis also have been utilised in other studies, the purpose of this paper is to further investigate utilisation of different models and how they perform regarding classification of the data set. It is also worth mentioning that a lot of the previous work done on a modified version of this data set, is done in context of an online competition on the platform kaggle, and we have not utilized any code from these projects directly.
However, the code in this project is influenced, and inspired, by the code presented in the data lab of the course ML-MIRI spring 2021, teached by Marta Vicente.

3. Data

3.1. The credit-g Data Set

The credit-g data set was collected by Dr. Hans Hofmann. This data set has been downloaded from: https://www.openml.org/d/31 (3). It contains 1000 numbers of customers, which is classified and labeled either as “good” or “bad”. There are 19 features, we have 7 numerical features and 12 categorical variables.

Additionally, Credit-g is an unbalanced data set, with 700 good, and 300 bad customers.

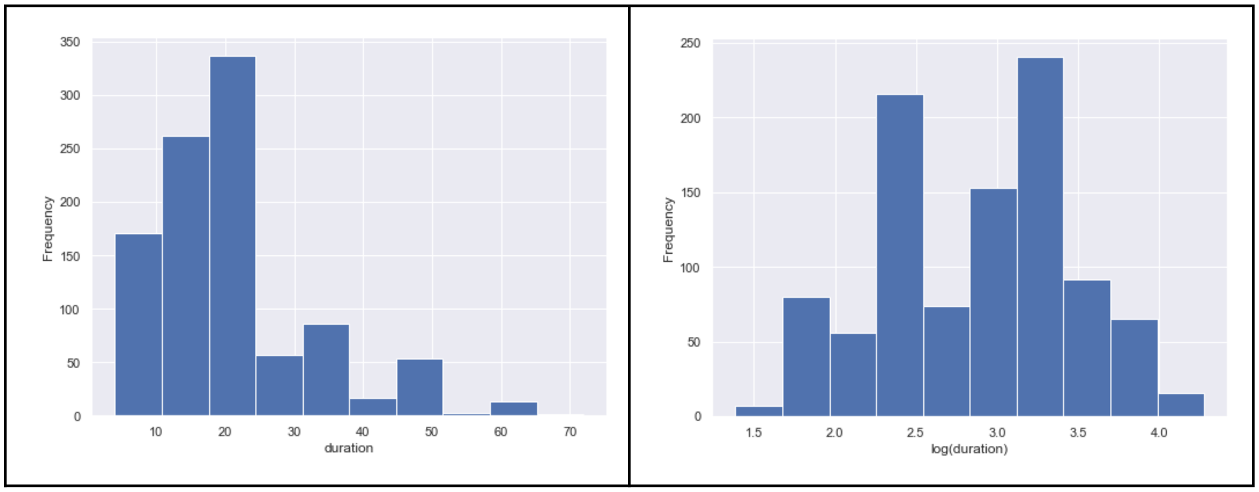
In order to get a better understanding of the customers in the dataset, it was desired to visually display some of the features. The results of these visualisations are shown in Attachment 1.

3.2. Preprocessing of the Data

In the credit-g dataset we did not have missing values so we do not need to remove or use other methods to fill empty places. Therefore steps for preprocessing have been provided below.

3.2.2. Handling categorical data

Based on the visualization of the dataset, it was seen that some of the variables were skewed.
A widely utilized method to address skewed data is the log transformation. This method is a data transformation method based on the assumption that if our original data follows, or approximately, follows a log-normal distribution, we can replace each variable x with the log(x)(1). This transformation will, given that the assumption of log-normal distribution holds, result in a more gaussian shape, which is better for the statistical analyses. Hence, this model was utilized on the most skewed variables, “*duration*” and “*credit_amount*”. The result of this transformation is visualized in the figure 1, where the variables on the left side displays the visualisation before utilization of the log transformation, and the right side displays the result afterwards. Based on the results in the displayed data, it is reasonable to believe that the assumptions about the log-normal distribution of the data is correct.



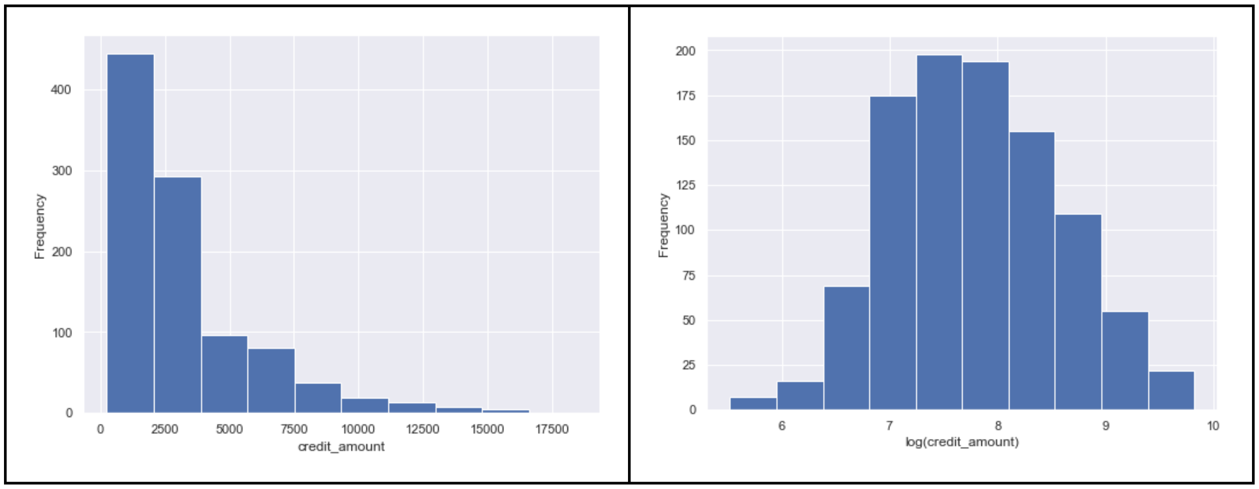


Figure 1: Visualisation of the variables “duration”, and “credit_amount” before and after the log transformation.

3.2.1. Converting variables

- Converting Ordinal Categorical columns to numeric

  In this case, since the maximum number of categories in a variable was not larger than 10, we did not need to merge categories. A dictionary was defined to convert each categorical value to a numerical value.

- Converting Binary nominal Categorical columns to numeric using 1/0 mapping

  GoodCredit=1 means the loan was a good decision.

  GoodCredit=0 means the loan was a bad decision.

3.2.2. Data Transformation

After splitting the data set this step was done, because our goal was to scale the data between zero and one.

3.3. Splitting the Data Set

3.3.1. Training, validation and test split

In order to analyse, and perform different machine learning methods on the dataset, the data set was divided into two. Where 80% of the total dataset was utilized for training, and the rest

20% were used for test set. For tuning hyperparameters we selected 20% of the train set as a validation set. In the following (table 1) the number of customers in each set has been provided.

Sets	**Training set**	**Validation set**	**Test set**
N of data(percentage)	640 (60%)	160 (20%)	200 (20%)

Table 1: A visualisation of how we splitted the data into training, validation and test.

3.3.2. Handling unbalance in the data set

As mentioned, the number of categorized "*good*" customers is almost twice the number of "*bad*" customers. This imbalance may result in poor generalization results, as well as bias towards the class with the magnitude of the data points.
In order to account for this unbalance in the set, stratified partitions were utilized. Hereby a random selection of the customers in the respective stratas (groups), "good" and "bad", were chosen in such a way that the proportion between them were equal in each partition. In this manner the splits reduced the proportion of prone towards one of the categories.

4. Methods

To classify customers as either "*bad*" or "*good*", both discriminative and generative supervised learning methods were utilized. Specifically this approach was utilized because we wanted to test how the different methods performed on the data set, and which one of the strategies worked better than the other. In this section the chosen methods will be briefly explained.

4.1. The different suggested models

The common denominator for the suggested models are that they are fast, easy to understand and interpret, and are simple to implement.
Primarily these algorithms were chosen based on the features of the data set, such that it is imbalanced and has a lot of variables. Even Though we tried to minimize the effect of the imbalance by implementing stratified partitions and oversampling, we tried to choose models that could possibly handle the difficulties in the dataset.

For all the methods except Random Forest this was one of the main reasons. As for the Random Forest, it is in addition easily understandable, and has a feature for balancing errors in imbalanced data sets, which was considered relevant when dealing with our dataset. Due to this feature we initially assumed that this model was going to excel in the classification of the credit -g data set.

4.2. Validation protocol

To validate models we need to select a suitable validation metric based on our goal and data set. An important point that we should keep in mind is that predicting a bad customer as a good customer is worse than predicting a good customer as a bad customer. Additionally, as described before this dataset is imbalanced, so it makes it hard to classify bad and good customers with high confidence levels. Therefore to select the best model we should consider f1-score(class 0), because this metric gives us a good balance between precision and recall.
At the end to estimate the generalization performance of the best model, the selected model will be run on a test set.

5. Results

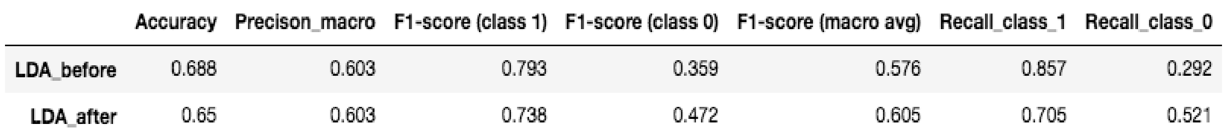
5.1. Generative models

5.1.1. LDA

As we described before, the Credit-g data set is imbalanced, one of methods to deal with this problem is oversampling. Therefore oversampling has been in the LDA and QDA methods. We will compare results before and after using oversampling.
In this technique we will increase the number of bad customers so that number of bad and good customer will be equal and we will have a balanced data set, like below:

Before balancing: Counter({1: 448, 0: 192})
After balancing: Counter({0: 448, 1: 448})

As we can see in the table (table 2) below, the performance of the LDA method after using oversampling in class 0 is better than before using oversampling (0.472>0.359).



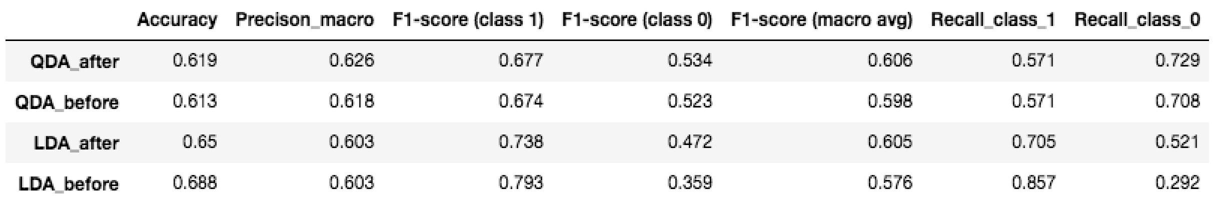
	Accuracy	Precison_macro	F1-score (class 1)	F1-score (class 0)	F1-score (macro avg)	Recall_class_1	Recall_class_0
LDA_before	0.688	0.603	0.793	0.359	0.576	0.857	0.292
LDA_after	0.65	0.603	0.738	0.472	0.605	0.705	0.521

Table 2: Performance of the LDA before and after utilization of oversampling.

5.1.2. QDA

The QDA model has been run for different regularization parameters and the best one will be selected, based on maximum f1-score class(0). In this case we have best performance in the best-parm = 1.

Like the LDA methode we run QDA after oversampling, in the table below (table 3) the results of two methods in two cases have been provided. As we can see, using oversampling in QDA is not as efficient as the LDA method. It increases performance in QDA just 0.011, but in LDA it improves 0.113. However QDA has better performance in comparison with LDA (0.534>0.472).

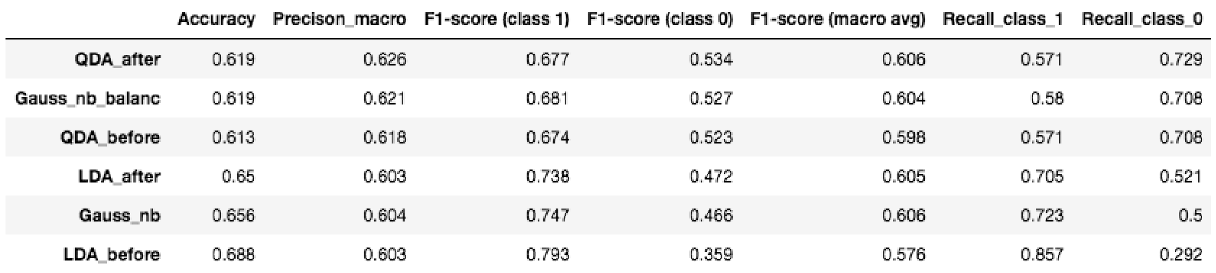


	Accuracy	Precison_macro	F1-score (class 1)	F1-score (class 0)	F1-score (macro avg)	Recall_class_1	Recall_class_0
QDA_after	0.619	0.626	0.677	0.534	0.606	0.571	0.729
QDA_before	0.613	0.618	0.674	0.523	0.598	0.571	0.708
LDA_after	0.65	0.603	0.738	0.472	0.605	0.705	0.521
LDA_before	0.688	0.603	0.793	0.359	0.576	0.857	0.292

Table 3: The performance results of both QDA and LDA.

5.1.3. Gaussian Naive Bayes

As we know in all generative models we assume assumptions, if these assumptions are true for a data set these models are so powerful. Gaussian naive bayes model is like QDA but with a diagonal covariance matrix, which means that features are independent.
Before using oversampling this method does not work well. After using oversampling it works better, but QDA like before is on the top of the table and has better performance, as seen in table 4.



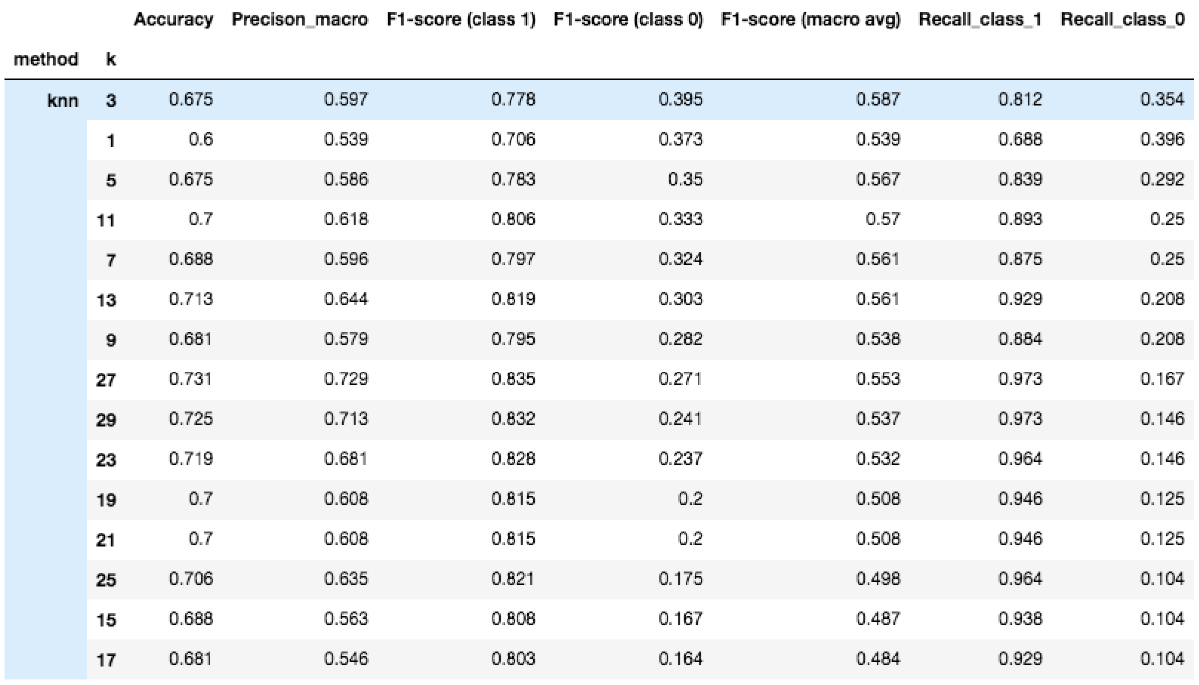
	Accuracy	Precison_macro	F1-score (class 1)	F1-score (class 0)	F1-score (macro avg)	Recall_class_1	Recall_class_0
QDA_after	0.619	0.626	0.677	0.534	0.606	0.571	0.729
Gauss_nb_balanc	0.619	0.621	0.681	0.527	0.604	0.58	0.708
QDA_before	0.613	0.618	0.674	0.523	0.598	0.571	0.708
LDA_after	0.65	0.603	0.738	0.472	0.605	0.705	0.521
Gauss_nb	0.656	0.604	0.747	0.466	0.606	0.723	0.5
LDA_before	0.688	0.603	0.793	0.359	0.576	0.857	0.292

Table 4: Visualization of the results of the utilization of QDA, LDA, and Gaussian naive bayes.

5.2. Discriminative classifiers

5.2.1. KNN

In the k nearest neighbor model, different numbers for k have been used and one of them that has better performance will be selected. It is important to keep in mind that a small number of k will cause overfitting and a large number of k will give us a model that is too simple and it will be a biased model. In the table below result of using different k has been provided:



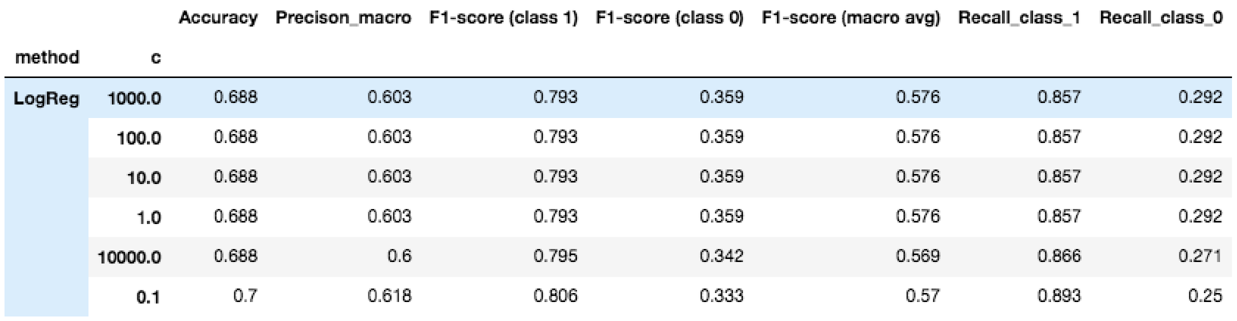
method	k	Accuracy	Precison_macro	F1-score (class 1)	F1-score (class 0)	F1-score (macro avg)	Recall_class_1	Recall_class_0
knn	3	0.675	0.597	0.778	0.395	0.587	0.812	0.354
	1	0.6	0.539	0.706	0.373	0.539	0.688	0.396
	5	0.675	0.586	0.783	0.35	0.567	0.839	0.292
	11	0.7	0.618	0.806	0.333	0.57	0.893	0.25
	7	0.688	0.596	0.797	0.324	0.561	0.875	0.25
	13	0.713	0.644	0.819	0.303	0.561	0.929	0.208
	9	0.681	0.579	0.795	0.282	0.538	0.884	0.208
	27	0.731	0.729	0.835	0.271	0.553	0.973	0.167
	29	0.725	0.713	0.832	0.241	0.537	0.973	0.146
	23	0.719	0.681	0.828	0.237	0.532	0.964	0.146
	19	0.7	0.608	0.815	0.2	0.508	0.946	0.125
	21	0.7	0.608	0.815	0.2	0.508	0.946	0.125
	25	0.706	0.635	0.821	0.175	0.498	0.964	0.104
	15	0.688	0.563	0.808	0.167	0.487	0.938	0.104
	17	0.681	0.546	0.803	0.164	0.484	0.929	0.104

Table 5: Visualisation of utilization of different k's and the performance of knn

The best k that can detect bad customers better is equal to 3, but its performance for class zero is not as good as QDA.(0.395<0.534)

5.2.2. Logistic Regression

For logistic regression different values for c have been tested. It will help us to find the best model based on different regularization parameters. Based on the table below we can find the best value for c= 1000.



method	c	Accuracy	Precison_macro	F1-score (class 1)	F1-score (class 0)	F1-score (macro avg)	Recall_class_1	Recall_class_0
LogReg	1000.0	0.688	0.603	0.793	0.359	0.576	0.857	0.292
	100.0	0.688	0.603	0.793	0.359	0.576	0.857	0.292
	10.0	0.688	0.603	0.793	0.359	0.576	0.857	0.292
	1.0	0.688	0.603	0.793	0.359	0.576	0.857	0.292
	10000.0	0.688	0.6	0.795	0.342	0.569	0.866	0.271
	0.1	0.7	0.618	0.806	0.333	0.57	0.893	0.25

Table 6: Visualisation of utilisation of Logistic Regression with different values for c

After finding the best c, comparing models that have been done until now will come in this table below. LDA and Logistic regression are on the bottom of the table. Why does it happen?Is there anything common in these models?

Although these two methods have different structure, like assumptions we had in LDA, these methods separate data by a line or hyperplane.
Logistic regression is a linear method and separates two classes with a hyperplane, and LDA has a linear boundary.

	Accuracy	Precison_macro	F1-score (class 1)	F1-score (class 0)	F1-score (macro avg)	Recall_class_1	Recall_class_0
QDA_after	0.619	0.626	0.677	0.534	0.606	0.571	0.729
Gauss_nb_balanc	0.619	0.621	0.681	0.527	0.604	0.58	0.708
QDA_before	0.613	0.618	0.674	0.523	0.598	0.571	0.708
LDA_after	0.65	0.603	0.738	0.472	0.605	0.705	0.521
Gauss_nb	0.656	0.604	0.747	0.466	0.606	0.723	0.5
knn_best	0.675	0.597	0.778	0.395	0.587	0.812	0.354
LDA_before	0.688	0.603	0.793	0.359	0.576	0.857	0.292
Logestic_reg	0.688	0.603	0.793	0.359	0.576	0.857	0.292

Table 7: summary of model performance

5.2.3. Decision tree family

In the decision tree family, we run Decision tree, Random forest and Extratree. To improve the result of each model we tune hyperparameters with gridsearchcv from sklearn.

5.2.3.1. Decision tree

One risk that exists in decision trees is overfitting. In the first step we should find the depth of the tree, if it is too large, the risk of overfitting will be increased.
Tree depth:15
Nodes:265
To make sure that the model is overfitting or not, we should check validation scores. Based on this result, the model does not have a good performance in detecting bad customers.

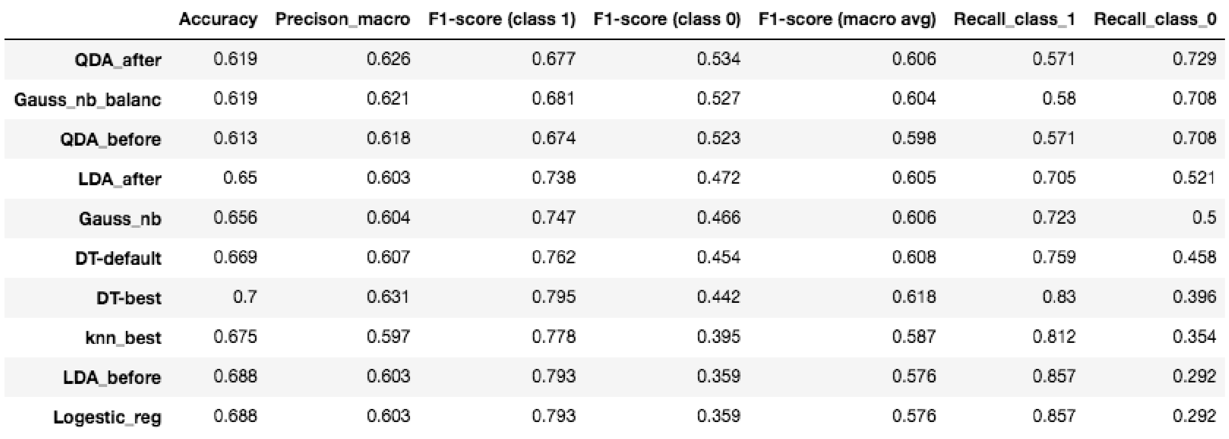
```
              precision    recall  f1-score   support

         bad       0.45      0.46      0.45        48
        good       0.77      0.76      0.76       112

    accuracy                           0.67       160
   macro avg       0.61      0.61      0.61       160
weighted avg       0.67      0.67      0.67       160
```

To improve model performance, we tuned hyperparameters. We used gridsearch and cross validation = 10. Our best hyperparameters have been provided below.
{'criterion': 'entropy',
 'max_depth': 14,
 'max_features': 'log2',
 'min_samples_leaf': 6,
 'min_samples_split': 14}

In the next step we should ask, does our tuning improve model performance or not. To answer this question we should consider table below:



	Accuracy	Precison_macro	F1-score (class 1)	F1-score (class 0)	F1-score (macro avg)	Recall_class_1	Recall_class_0
QDA_after	0.619	0.626	0.677	0.534	0.606	0.571	0.729
Gauss_nb_balanc	0.619	0.621	0.681	0.527	0.604	0.58	0.708
QDA_before	0.613	0.618	0.674	0.523	0.598	0.571	0.708
LDA_after	0.65	0.603	0.738	0.472	0.605	0.705	0.521
Gauss_nb	0.656	0.604	0.747	0.466	0.606	0.723	0.5
DT-default	0.669	0.607	0.762	0.454	0.608	0.759	0.458
DT-best	0.7	0.631	0.795	0.442	0.618	0.83	0.396
knn_best	0.675	0.597	0.778	0.395	0.587	0.812	0.354
LDA_before	0.688	0.603	0.793	0.359	0.576	0.857	0.292
Logestic_reg	0.688	0.603	0.793	0.359	0.576	0.857	0.292

Table 8: summary of model performance

Based on the result, DT_best has a lower f1-score in class 0. However it behaves well in the training set and has larger value in f1-score but it does not behave in the validation set as like the training set.

5.2.3.2. Random Forest

To limit overfitting and find a robust classification method random forest has been used. In this model instead of using a single decision tree, many decision trees have been used.
In this part we used random forest, balanced random forest and tuned hyperparameters to find the best random forest.
The best parameter after tuning by gridsearch and cross validation= 10 are like below:

```
'class_weight': 'balanced_subsample',
 'max_depth': 10,
 'min_samples_leaf': 6,
 'min_samples_split': 4,
 'n_estimators': 200
```

In the table below the result of using different models has been provided. As we can see Random forest with best parameters has better performance compared to other methods.

	Accuracy	Precison_macro	F1-score (class 1)	F1-score (class 0)	F1-score (macro avg)	Recall_class_1	Recall_class_0
RF-best	0.719	0.666	0.798	0.536	0.667	0.795	0.542
QDA_after	0.619	0.626	0.677	0.534	0.606	0.571	0.729
Gauss_nb_balanc	0.619	0.621	0.681	0.527	0.604	0.58	0.708
QDA_before	0.613	0.618	0.674	0.523	0.598	0.571	0.708
RF-default	0.744	0.694	0.83	0.481	0.655	0.893	0.396
RF-balance	0.769	0.756	0.851	0.479	0.665	0.946	0.354
LDA_after	0.65	0.603	0.738	0.472	0.605	0.705	0.521
Gauss_nb	0.656	0.604	0.747	0.466	0.606	0.723	0.5
DT-default	0.669	0.607	0.762	0.454	0.608	0.759	0.458
DT-best	0.7	0.631	0.795	0.442	0.618	0.83	0.396
knn_best	0.675	0.597	0.778	0.395	0.587	0.812	0.354
LDA_before	0.688	0.603	0.793	0.359	0.576	0.857	0.292
Logestic_reg	0.688	0.603	0.793	0.359	0.576	0.857	0.292

Table 9: adding RF-best to model performance

5.2.3.3. Extratree

The process of running codes related to extratree is the same as random forest. We will not describe it and just provide the result of this model.

5.2.4. Ensembles

Maybe we can improve results by using ensembles method, in this method we multiple diverse models with different fashion such as soft and hard votings. In the hard fashion, majority vote has been used and in the soft fashion averaging probabilities has been used.
In the following we bring some of these combinations:

1. GaussianNB, extra_trees, randomforest_model and tree_model, hard voting
2. GaussianNB, extra_trees, randomforest_model and tree_model, soft voting
3. Tuned Randomforest and QDA (the first two models in our case)

The result of running all models in this research has been presented below.

	Accuracy	Precison_macro	F1-score (class 1)	F1-score (class 0)	F1-score (macro avg)	Recall_class_1	Recall_class_0
RF-best	0.719	0.666	0.798	0.536	0.667	0.795	0.542
voting_best	0.619	0.626	0.677	0.534	0.606	0.571	0.729
QDA_after	0.619	0.626	0.677	0.534	0.606	0.571	0.729
Gauss_nb_balanc	0.619	0.621	0.681	0.527	0.604	0.58	0.708
QDA_before	0.613	0.618	0.674	0.523	0.598	0.571	0.708
extra_trees-best	0.7	0.645	0.784	0.51	0.647	0.777	0.521
RF-default	0.744	0.694	0.83	0.481	0.655	0.893	0.396
RF-balance	0.769	0.756	0.851	0.479	0.665	0.946	0.354
LDA_after	0.65	0.603	0.738	0.472	0.605	0.705	0.521
ens_soft	0.719	0.656	0.809	0.471	0.64	0.848	0.417
Gauss_nb	0.656	0.604	0.747	0.466	0.606	0.723	0.5
ens_hard	0.694	0.629	0.786	0.462	0.624	0.804	0.438
DT-default	0.669	0.607	0.762	0.454	0.608	0.759	0.458
DT-best	0.7	0.631	0.795	0.442	0.618	0.83	0.396
extra_trees_default	0.719	0.654	0.815	0.416	0.615	0.884	0.333
knn_best	0.675	0.597	0.778	0.395	0.587	0.812	0.354
Logestic_reg	0.688	0.603	0.793	0.359	0.576	0.857	0.292
LDA_before	0.688	0.603	0.793	0.359	0.576	0.857	0.292

Table 10: performance of all models

Based on the above table we can find that voting-best has good performance and has low difference with RF-best. To find exactly which one of these methods is better we can run both models in the test set and what will happen.

```
              precision    recall  f1-score   support

         bad       0.34      0.55      0.42        60
        good       0.74      0.54      0.62       140

    accuracy                           0.54       200
   macro avg       0.54      0.54      0.52       200
weighted avg       0.62      0.54      0.56       200
```

Result of using voting-best

```
              precision    recall  f1-score   support

         bad       0.52      0.68      0.59        60
        good       0.84      0.73      0.78       140

    accuracy                           0.71       200
   macro avg       0.68      0.71      0.69       200
weighted avg       0.75      0.71      0.72       200
```

Result of using Random forest

Based on these results we can find that random forests have a better performance on test set. In the next step we should find which factors are more important.

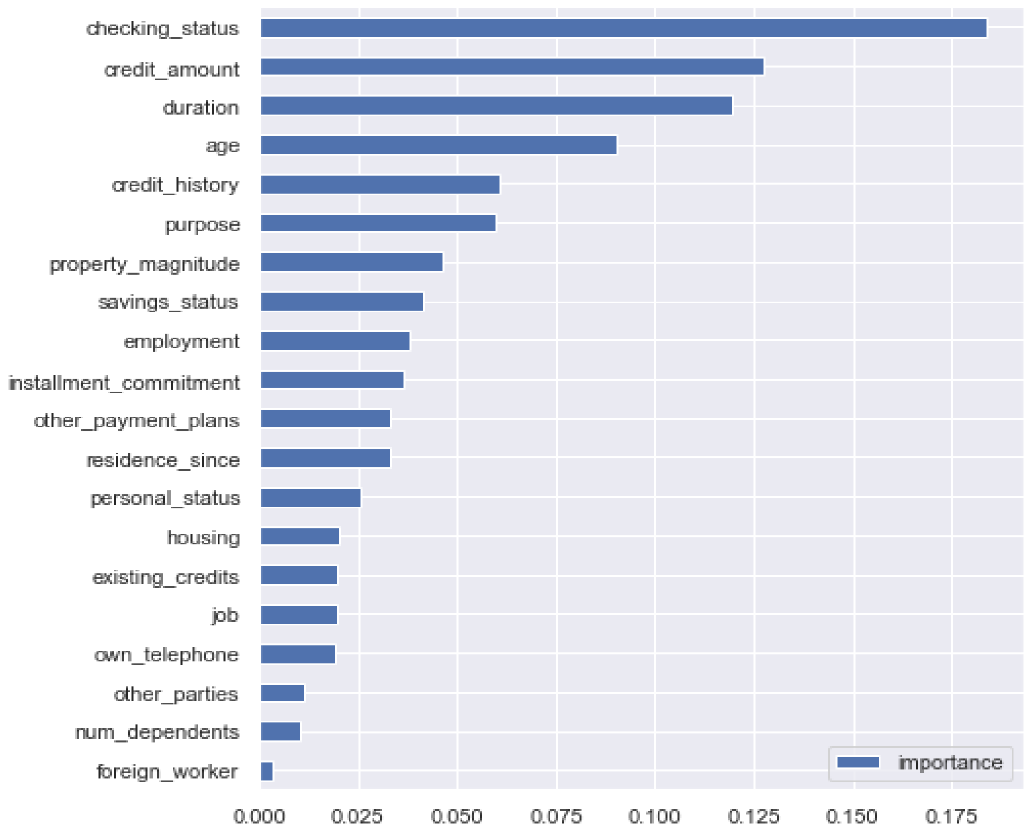


Figure2: important factors

Based on this figure checking status is more important in classifying a customer as a bad or good customer.

If we check the data set we can see that most of the customers that were labeled as a good customer did not have check status, Status of existing checking account. The reason may come to mind is that customers who do not have money in their account prefer to apply for a loan, and customers who have a lot of money in their account do need to apply for it. Therefore number of people who apply for a loan in no checking status is greater than other groups and they will have a more chance to be a good customer too.

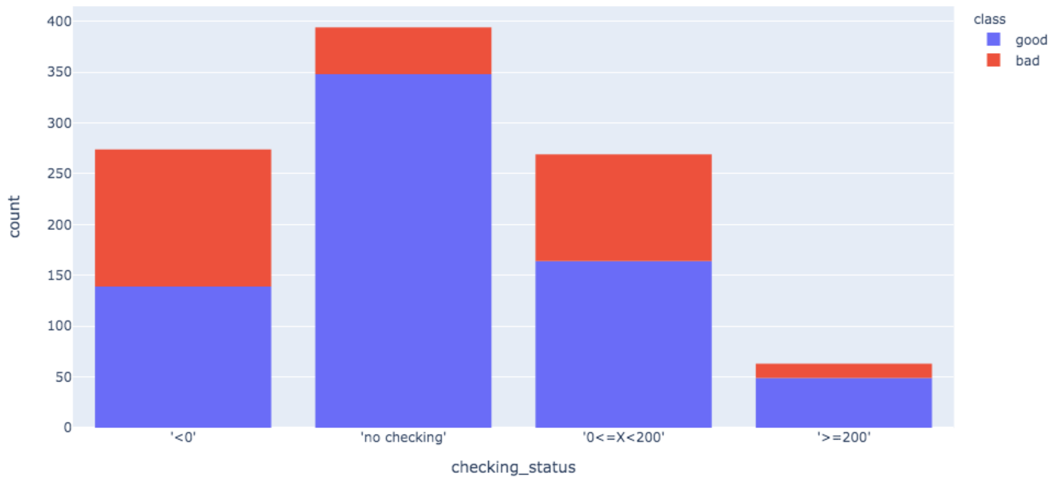


Figure 3: checking status

The second important feature is credit-amount, as we can see in figure below, most of the good customers have 2500 credit that are categorized as low credit based on the range of credit in this dataset. Therefore it makes sense that they will apply for a loan more than other customers. Like before they will have more chances to be a good customer.

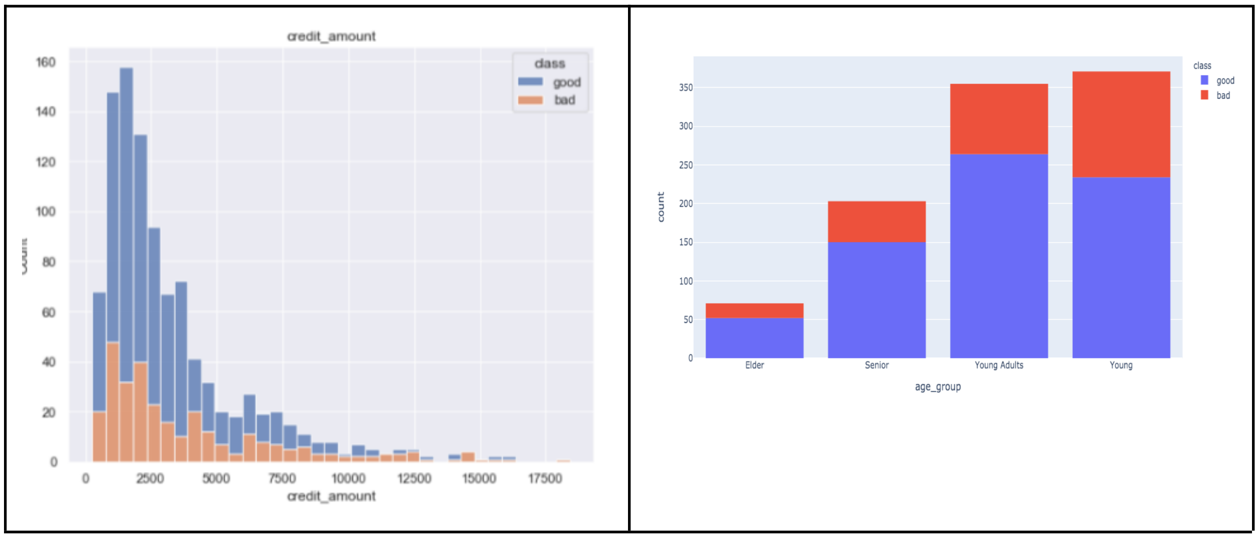


Figure4: Credit_amount and age range

In the right plot we can see different range of customers that apply for a loan, we used this range to categorised different ages; “Young” (age-range 19-29), “Young Adults” (age-range 30 -40), “Senior” (age-range 41-55), and “Elder”(age-range over 55 years).

Number of customers that are young is more than other groups and 36% of these customers are labeled as bad customers. The second group are young customers and 25% of them are bad customers. It seems that the number of bad customers when people get older is decreasing.

6. Reference

1) https://medium.com/@kyawsawhtoon/log-transformation-purpose-and-interpretation-9444b4b049c9 fetched; 14.05.21

2) https://medium.com/strands-tech-corner/unbalanced-datasets-what-to-do-144e0552d9cd fetched; 14.05.21: 21:39
3) https://www.openml.org/d/31 fetched
4) https://towardsdatascience.com/decision-trees-and-random-forests-df0c3123f991
5) https://stats.stackexchange.com/questions/431695/german-credit-dataset-interpretation-of-checking-status-feature