Analytical Modeling - Exam 2019

Introduction

Credit risk is one of the major financial risks that exists in the banking system. How a bank manages its credit risk is very critical for its performance over time; capital depletion through loan losses has been the proximate reason for most organization failures. Distinguishing and rating credit risk is the primary step in managing it effectively. The applicant is a good credit risk if the user is likely to repay the loan. Conversely, if the user is not likely to repay the loan, the applicant is considered as a bad credit risk. The bank has to analyze applicant's demographic and socio-economic profiles before a decision is made regarding his and her loan application.
The data set provided for this exam is the German credit data set (1994), the country and date tells us that we will be dealing with Deutsch Mark as currency (DM). The objective is to solve a machine-learning problem such as credit risk classification. The objective of solving a credit risk classification problem is to minimize loss from the bank's perspective; for example, when a person applies for a loan, the bank has to make a decision regarding whom to give approval for the loan and whom not to.
We will be using Python to work with this data set. Python is an interpreted language. The Python interpreter runs a program by executing one statement at a time. The real strength of Python is its multiple powerful libraries such as pandas and numpy. Numpy is a library that can perform numerical calculations at high speed. In general, when performing numerical calculations in a scripting language such as Python, it will be slower than C / C ++ because there are various interpreters. However, with numpy it is possible to perform matrix calculations and high-level math functions as fast as C / C ++, this makes Python not at a disadvantage in term of speed compared to R (data.table, dplyr) or Julia. The working environnment used for this exam will be Jupyter notebook, the complete code can be found on Github or Nbviewer.

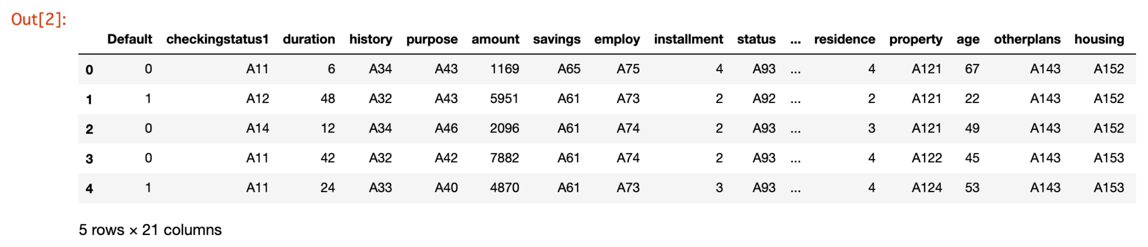
Exploratory Data Analysis (EDA)

The data set presents itself as a .csv file. Python has both built-in and third-party libraries for converting a .csv file into a Python object. Here we'll use the pandas library and its "pd read csv" to convert our data set into a pandas dataframe which will be used for our EDA. As for the other libraries involved in this work, we will use seaborn and matplotlib for visualization and sklearn for the machine learning part. OS is mostly use to interact with the operating system to get the work directory for example.

```
In [1]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as  plt
        import sklearn
        import os
```

```
In [2]: df = pd.read_csv("Exam2019.csv")
        df.head()
```

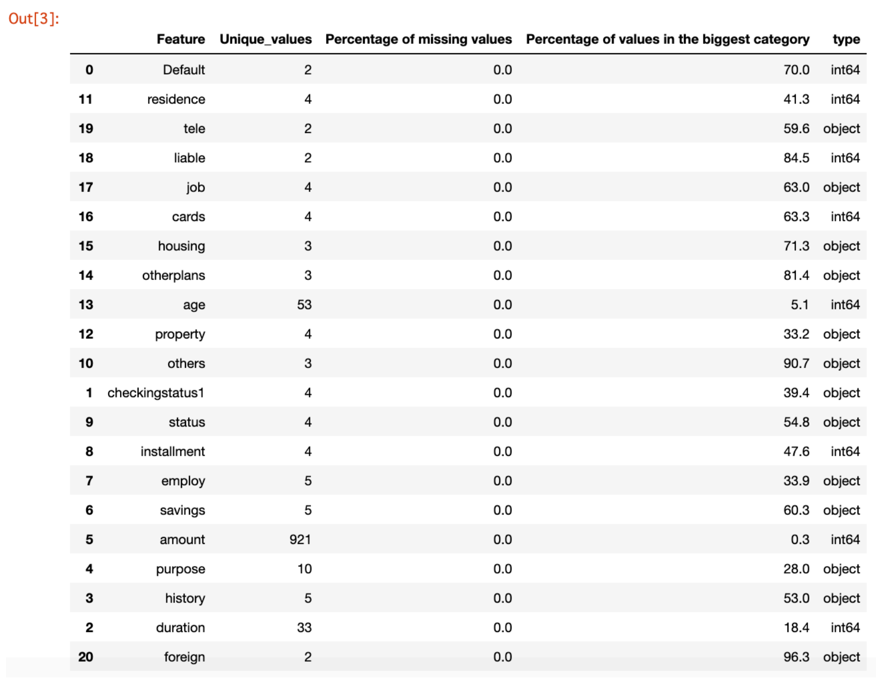
Here we use the command "df.head()" to have a first quick view of our dataframe. For large dataframes, the head method selects only the first five rows:


Out[2]:

	Default	checkingstatus1	duration	history	purpose	amount	savings	employ	installment	status	...	residence	property	age	otherplans	housing
0	0	A11	6	A34	A43	1169	A65	A75	4	A93	...	4	A121	67	A143	A152
1	1	A12	48	A32	A43	5951	A61	A73	2	A92	...	2	A121	22	A143	A152
2	0	A14	12	A34	A46	2096	A61	A74	2	A93	...	3	A121	49	A143	A152
3	0	A11	42	A32	A42	7882	A61	A74	2	A93	...	4	A122	45	A143	A153
4	1	A11	24	A33	A40	4870	A61	A73	3	A93	...	4	A124	53	A143	A153

5 rows × 21 columns

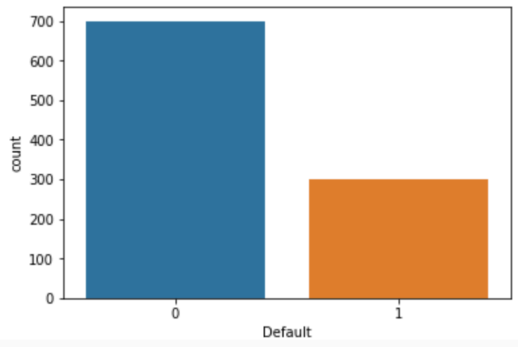
This dataset contains 20 categorical and symbolic attributes, the ”Default” column will be our target objective for the machine learning part. To go further we create a matrix which contains statistical information, this would be important at the start of an EDA to find clue such as the number of missing value and the data type for each column, this results in the following table:


Out[3]:

	Feature	Unique_values	Percentage of missing values	Percentage of values in the biggest category	type
0	Default	2	0.0	70.0	int64
11	residence	4	0.0	41.3	int64
19	tele	2	0.0	59.6	object
18	liable	2	0.0	84.5	int64
17	job	4	0.0	63.0	object
16	cards	4	0.0	63.3	int64
15	housing	3	0.0	71.3	object
14	otherplans	3	0.0	81.4	object
13	age	53	0.0	5.1	int64
12	property	4	0.0	33.2	object
10	others	3	0.0	90.7	object
1	checkingstatus1	4	0.0	39.4	object
9	status	4	0.0	54.8	object
8	installment	4	0.0	47.6	int64
7	employ	5	0.0	33.9	object
6	savings	5	0.0	60.3	object
5	amount	921	0.0	0.3	int64
4	purpose	10	0.0	28.0	object
3	history	5	0.0	53.0	object
2	duration	33	0.0	18.4	int64
20	foreign	2	0.0	96.3	object

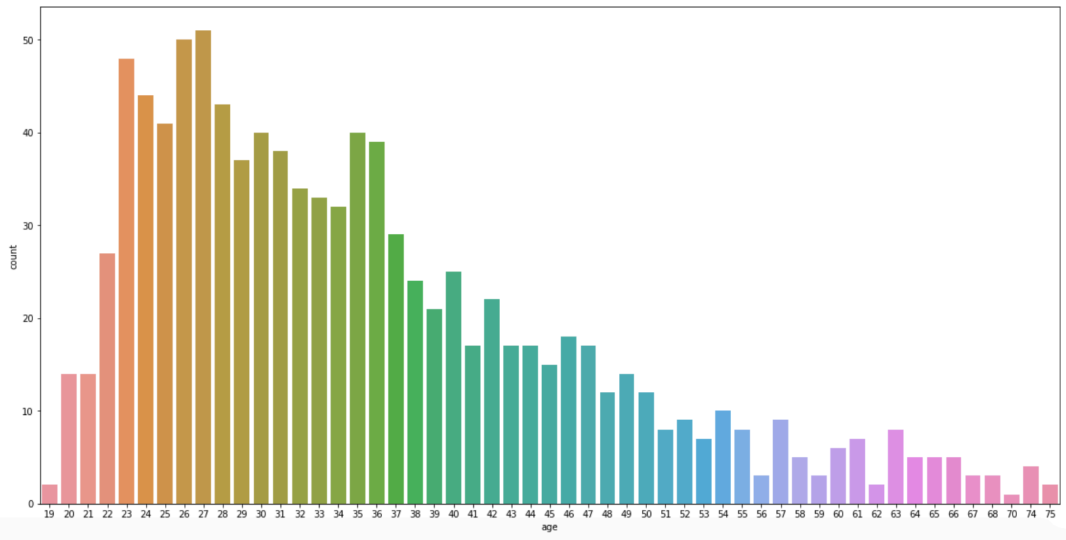
Here are few inferences that were obtained:
- There are no missing values in the dataset.
- The number of unique value is rather low except for one column which is our continuous variable (amount).
- The data set is unbalanced within each categories.
We confirm this first insight with a plot regarding our target variable ”Default”.

```
In [4]: ax = sns.countplot(x="Default", data=df)
        df['Default'].value_counts()

Out[4]: 0    700
        1    300
        Name: Default, dtype: int64
```



The plot confirms the fact that the dataset is not balanced, we will see later how we deal with it in the machine learning part.
We are dealing with a data set which is influenced by its socio-demographic aspect, let's check the age distribution:

```
In [6]: plt.figure(figsize=(20,10))
        ax = sns.countplot(x="age", data=df)
```



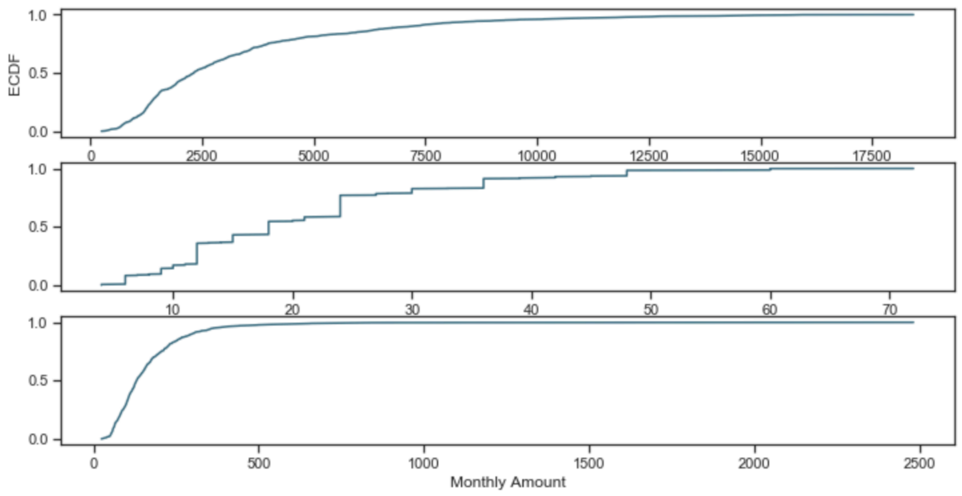
Plotting the frequency of the credit loan against the age groups shows that credit demands from applicants between the ages of 20 and 39 make up for about two-thirds of the total credit demands.
If we take a closer look at the amount, we see that most of the credit loan amounts are in the range of 2,000DM-4,000DM. The largest amount given is as high as 18,000DM. Count is 1,000 for this column, and we know that there are no missing value which means that all the values are present for each attribute in 1000 rows of the dataset.

```
In [7]: df['amount'].describe()

Out[7]: count    1000.000000
        mean     3271.258000
        std      2822.736876
        min       250.000000
        25%      1365.500000
        50%      2319.500000
        75%      3972.250000
        max     18424.000000
        Name: amount, dtype: float64
```

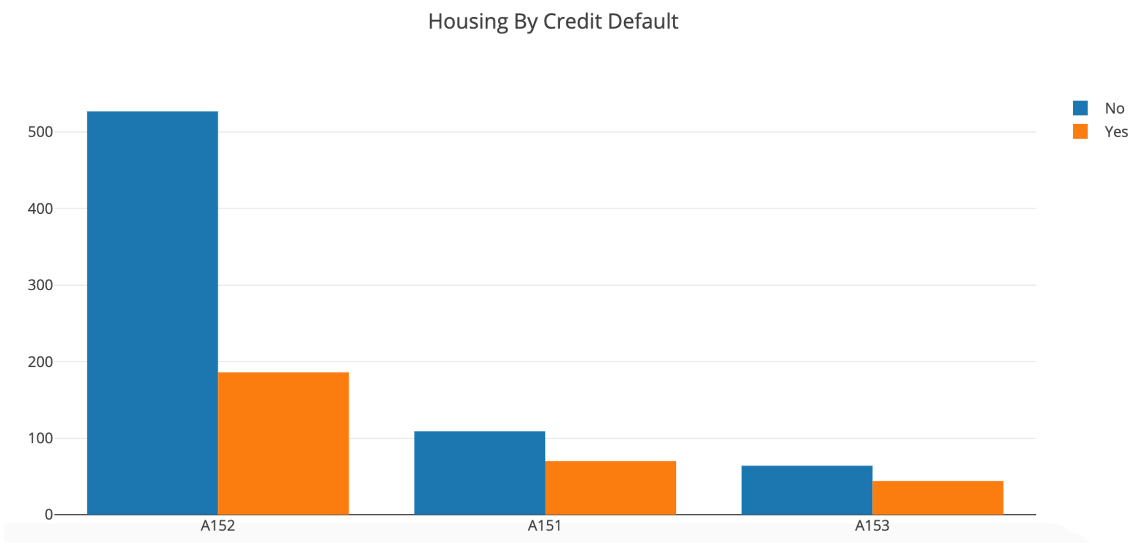
Now we also know that each credit has a specific duration, to avoid comparing each time amount and duration, we will create a new column which is the monthly amount for each credit (amount divided by duration).

```
In [8]: df['monthlyamount'] = df['amount']/df['duration']
        df['monthlyamount'].describe()

Out[8]: count    1000.000000
        mean      167.687020
        std       153.490959
        min        24.055556
        25%        89.600000
        50%       130.333333
        75%       206.183333
        max      2482.666667
        Name: monthlyamount, dtype: float64
```

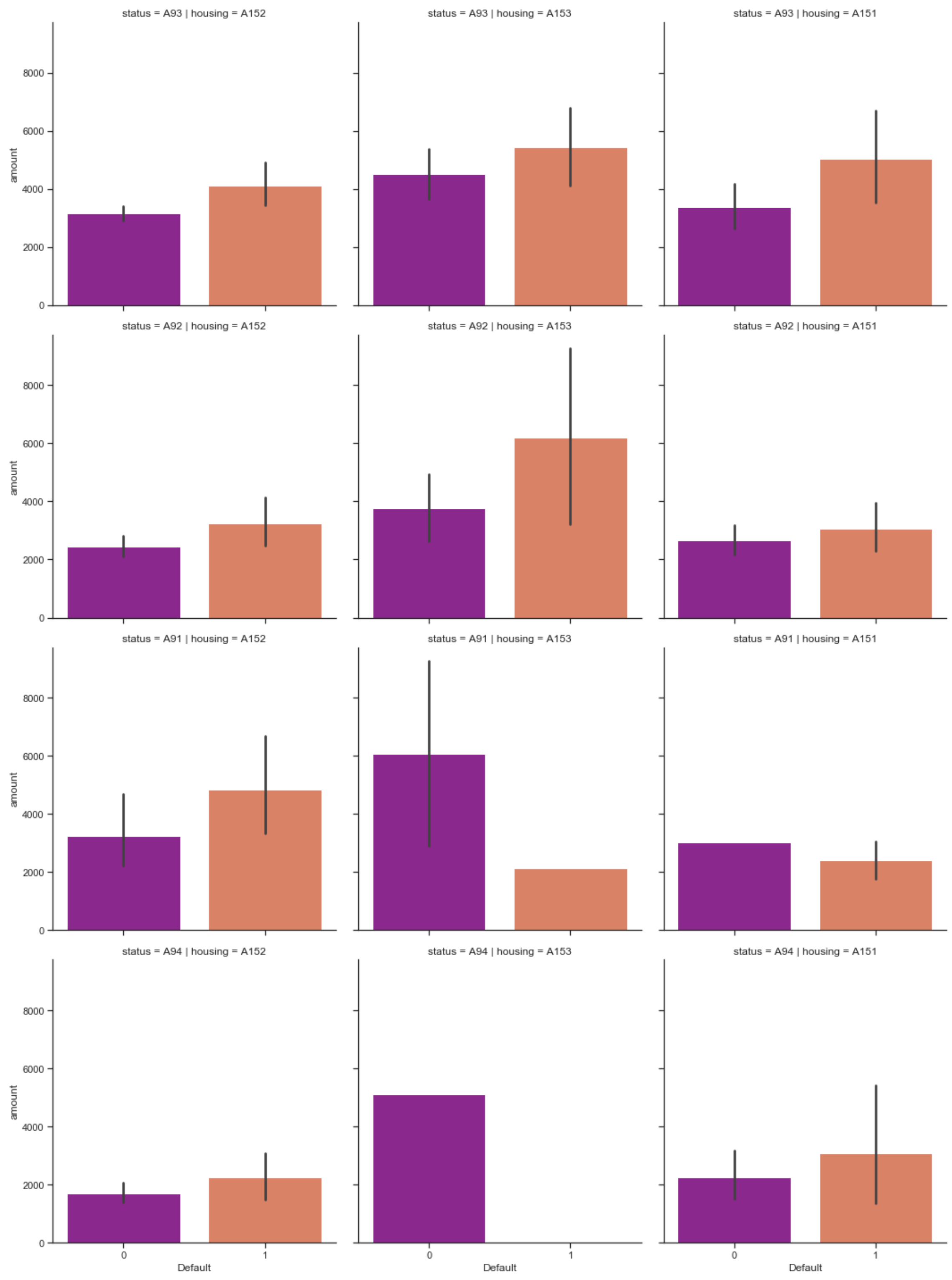
We can see that on average the monthly amount is 168 DM, the minimum amount is only 24 DM and the maximum is 100 times higher with 2483 DM.
Now let's take a look at the empirical cumulative distribution function (ECDF):



About 50% of data is comprised between 0 and 2500 DM.
If we analyze the housing part for default credit we can see:



Most people who own a house do not have a default credit compared to people who rent or have a house for free (the highest with a credit default).
We directly jump to our analysis using seaborn. We plot the different variables between each of them trying to find pattern, in this report we will only list the one we found relevant.



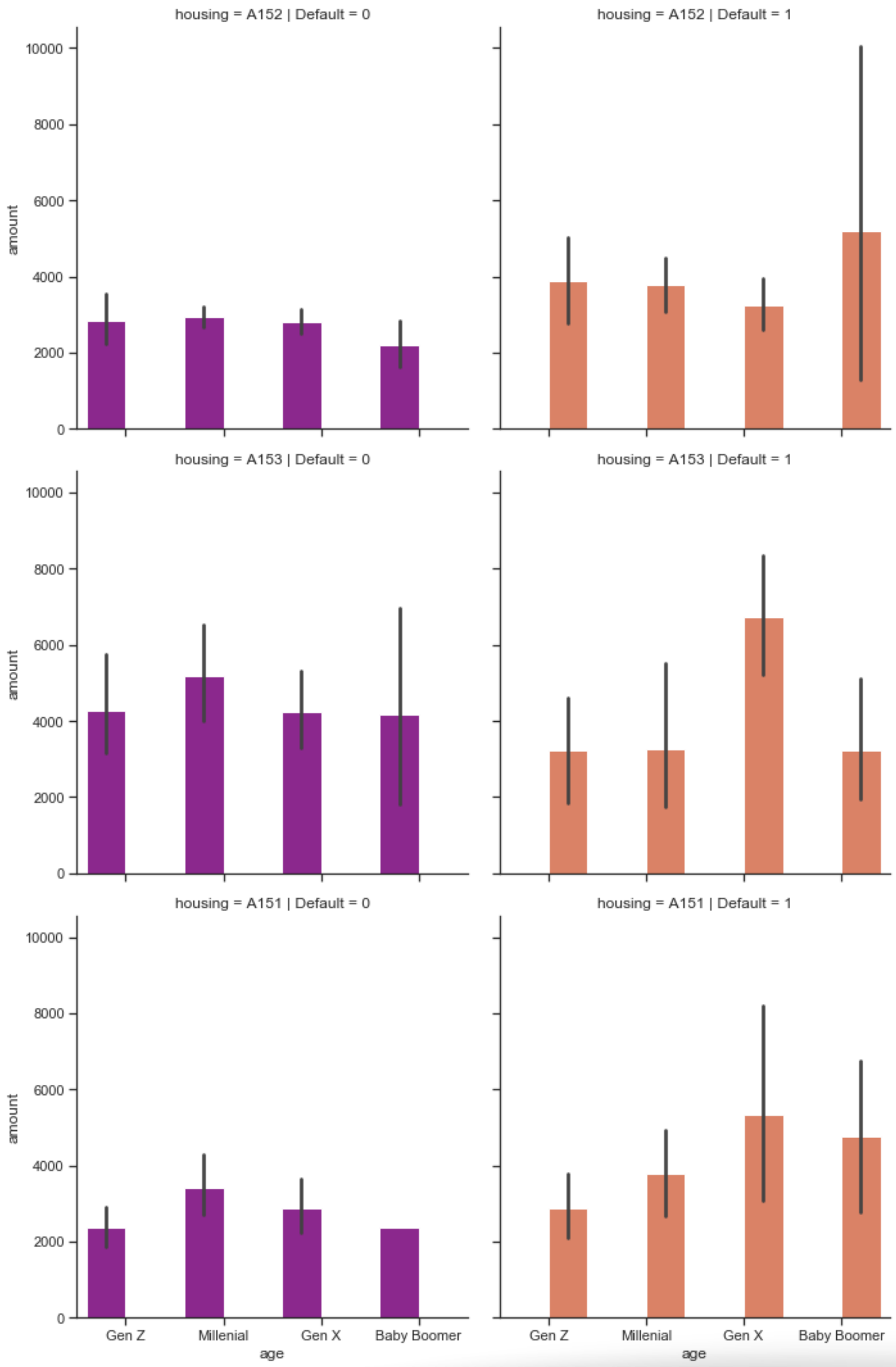
By looking at the status and sex, we can find some pattern:
- Single males have a marginally high default credit rate
- Married males who own a house do not have any default of credit

We can hardly analyze the other data because the dataset is not good enough to analyze some variables such as A92 which mix divorced female and married which are totally different population.

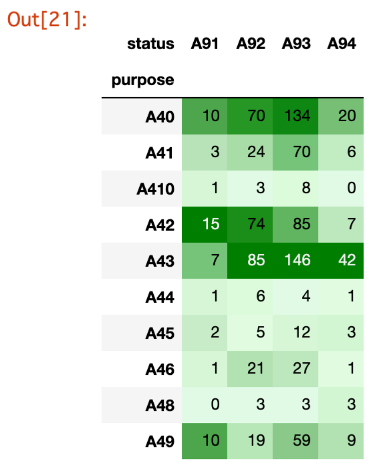
Our age category contains too many different value. Nevertheless, we can group the value under label such as generational:

In [16]:
```
interval = (18, 25, 35, 60, 120)

cats = ['Gen Z', 'Millenial', 'Gen X', 'Baby Boomer']
df["age"] = pd.cut(df.age, interval, labels=cats)
```

This results in the following analysis for housing:

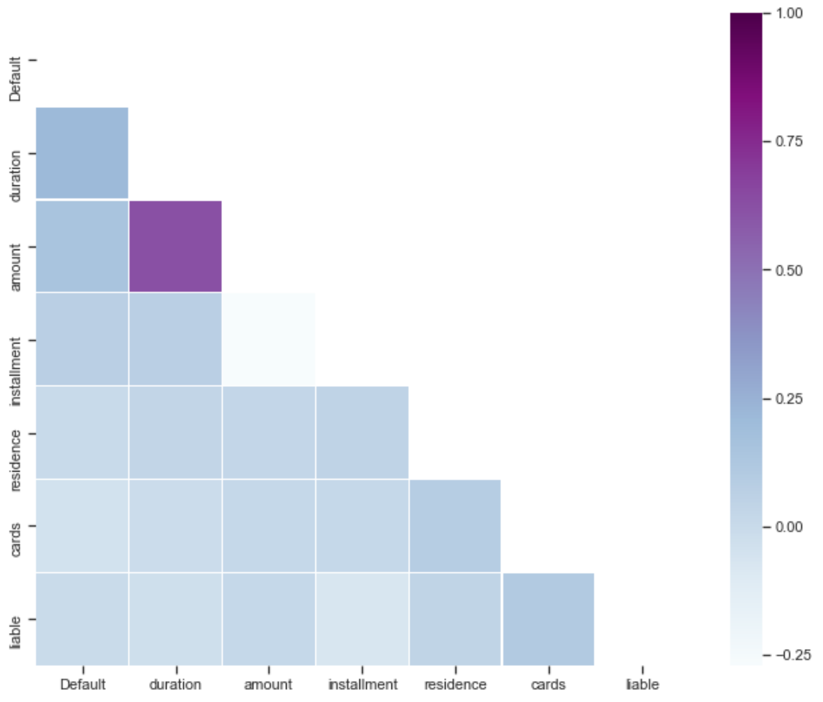


We can see that there is generational pattern in term of credit default, and that housing influence this pattern as well.
- For high amount of credit, baby boomers who own a house are more likely to have a credit default than their generational counter part.
- The same can be said for Generation X who live in a house for free or rent, they are more likely to have a credit default.
If we now examine the purpose of asking for a loan:

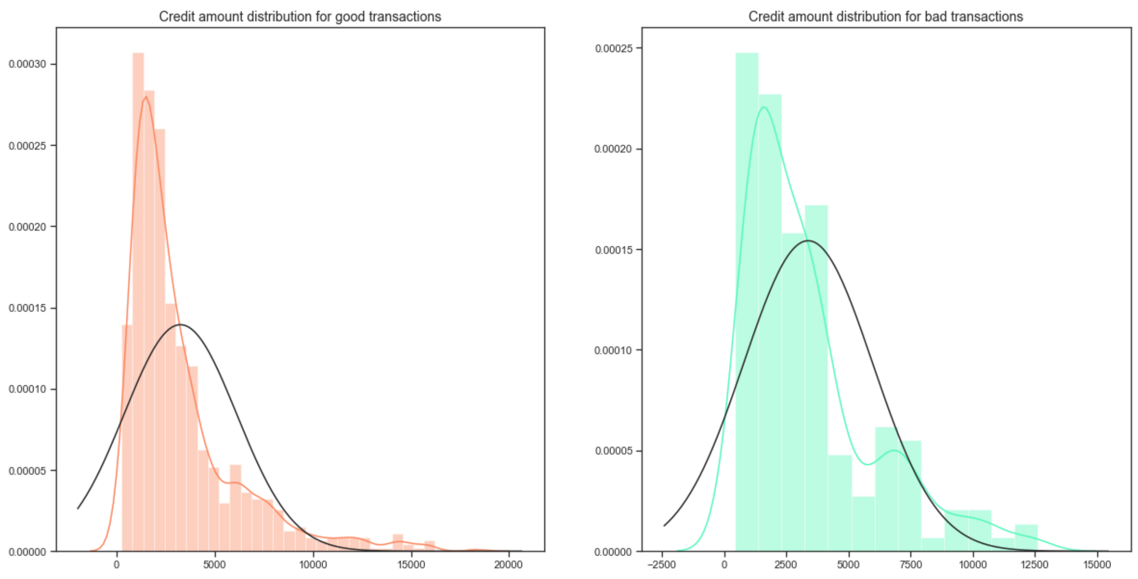

Out[21]:

status	A91	A92	A93	A94
purpose				
A40	10	70	134	20
A41	3	24	70	6
A410	1	3	8	0
A42	15	74	85	7
A43	7	85	146	42
A44	1	6	4	1
A45	2	5	12	3
A46	1	21	27	1
A48	0	3	3	3
A49	10	19	59	9

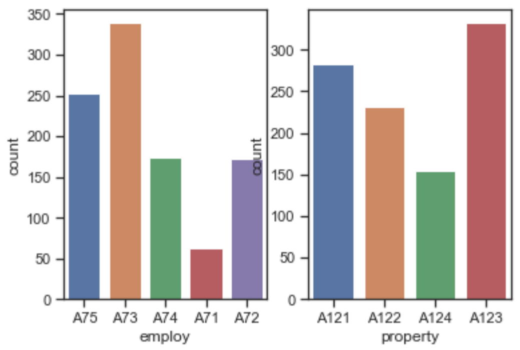
It appears that beside divorced male, the main purpose of submitting a credit is for electronics (radio/television back in the days).
As part of our EDA, we cannot skip the correlation analysis:



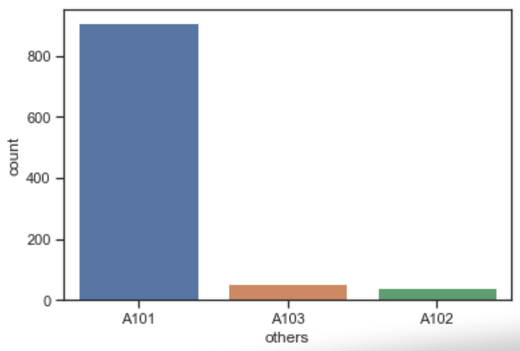
Correlation shows us only one correlation between amount and duration which totally obvious. Other than this there is no apparent correlation between variables.
Let's move to a more statistical analysis.



This figure shows us the distribution of our data set compared to the normal distribution, it appears that our data set is right skewed.
We now want to figure out if there is a trend in the data. To do so we run a multiple linear regression on our data set using all variables. The result is visible in Jupyter Notebook line 27 [1]. The multiple linear regression tells us that there variables which are not significant in our dataset which we may drop such as liable, cards, residence. These variables have a P-value > 0.05 which is the significant level we chose (95%). To make sure the variables which have a significant value are relevant, we need to take a look at the distribution:



We see it here that the data is fairly balanced which means there is an impact of the variable with a balanced level in the data. This is not the case for ’others’ which is totally unbalanced and may not be used for further analysis.



This concludes our EDA part, we will now be dealing with the main content of the report which is the machine learning part.

Machine Learning

With the advent of new computing technologies, today's machine learning helps companies to deal with large amount of data. Machine learning is to analyze this large amount of data iteratively and find out the underlying patterns. This is something that a computer is trying to realize "the same function as learning ability" that humans are doing naturally. In this way, it is possible to predict the future according to patterns by applying the results of computer analysis to new data.
The overall result of this factor is that machine learning can automatically generate in a short time a model that can analyze a large amount of complex data and provide more accurate results more quickly, and can handle very large-scale data as well. Moreover, building an accurate model gives businesses and organizations the opportunity to identify opportunities to generate revenue and avoid unknown risks.

Many coding languages are used for machine learning here is a non exhaustive list:
- Python
- R
- Octave
- C/C++
- Java
- etc...

In our case we will be using Python. We want the syntax simplicity of the language as it is to involve various tasks such as data set preprocessing and algorithm construction / tuning.
The main reason why we will be using Python instead of R is the number of libraries and frameworks in Python, the main library we will be using is scikit-learn which includes a large amount of models.

Model Choice

Which model is relevant for our analysis? Since our final objective is to have a proper ROC analysis with an optimized result be it AUC, F1-Score, Recall, Precision, Accuracy. It is difficult to choose the model which will work the best beforehand. Nevertheless, we can select among the most popular models, which one to use for our machine learning part.

Multiple Variable Regression: Multiple regression analysis, is a form of general linear modeling, and is a multivariate statistical technique used to examine the relationship between a single dependent variable and a set of independent variables. We used it for EDA but not for the machine learning part where we are dealing with a classification problem [2].

Logistic Regression: Class determination is one of the purposes of data analysis. Class distinction is to determine which group (= class) an individual belongs to from the data of the individual obtained by observation or experiment. In logistic regression analysis, "the probability that an individual x is observed and to which group it belongs". It is a method to predict the occurrence probability. The basic idea is the same as linear regression, but the formula and its assumptions are improved so that the prediction result is between 0 and 1 [3].

SVM (RBF, Linear): SVM (Support Vector Machine) is known as a method of machine learning with high classification accuracy. To obtain higher classification accuracy with SVM, hyperparameters need to be determined from training data. SVM is a method to determine hyperplanes that separate data point sets mapped to feature space. However, point sets in a feature space are not always separable. The cost parameter CC is a parameter that determines how much misclassification is permitted. The hyperplane is determined such that the larger the CC is, the smaller the CC is, the smaller the CC is. Linear SVC is a classification method based on SVM that does not use a kernel whereas RBF use a kernel (Gaussian kernel)[4].

KNN: The k-nearest neighbor method is one of the simplest of classification algorithms for supervised learning. The idea of the k-neighbor method is to find the data that is most similar to the test data in feature space. When adding new data to the training data which is already clustered, the test data will be attributed a class that depends only on the data at the closest distance, this is called KNN [5].

Decision Tree: Decision trees are one of machine learning methods that perform classification and regression using tree structures. To construct a decision tree we do the following: first, all training data is collected in the root node, then, we select the set of features and threshold values that will best divide the collected data among the variables of the data. After division by feature and threshold, division is repeated at each node again. The decision tree is built using this procedure. Entropy and Gini coefficient are often used to select this pair [6].

Random Forest: Decision tree model has a drawback which is over fitting. Indeed, when handling training data we do not consider the noise in data. Therefore, when training a Decision Tree model, you learn not only the correct data but also the noise, so the Decision Tree model obtained by this method has a high bias. In other words, the learned model has high accuracy for training data, but the accuracy decreases in actual operation. Random Forest has improves this shortcoming. The solution is to randomize the training data before building the model to create a subset of training data and subtract the bias for each subset's noise [7].

Naive Bayes: Naive Bayes algorithm is based on this Bayes theorem. It calculates the probability of all the estimates given the data, and outputs the one with the highest probability as the estimation result. In this algorithm, it is assumed that the features of the data are independent and not correlated with each other. In other words, each feature value affects the estimation result independently. This is a very strong assumption and often does not hold true with real data [8].

XGBoost: First the name of the algorithm is really cool and makes people want to use it. Second, this algorithm is an ensemble algorithm which means it combines two different algorithm to create a more powerful one. So, XGBoost is a kind of method called GBDT (Gradient Boosted Decision Tree) combining boosting with trees. Boosting refers to a method that uses the model already learned up to reduce the error between training and testing data after learning. To do that, we calculate the derivation of the difference and repeat the process over and over again. This process can also be used for regression [9].

Global - Data Preparation

Data pre-processing involves the transformations being applied to the data before feeding it to the algorithm. Since most of the variables in the dataset are categorical, various techniques need to be applied for converting the categorical to numerical variables.
The first step in our data preparation is to normalize the data, to do so we use logarithm to rescale everything then the normal scaler included in scikit-learn library [10]. The scaler transform the data using z-value for scale. This gives us the following for the variable ”amount”:

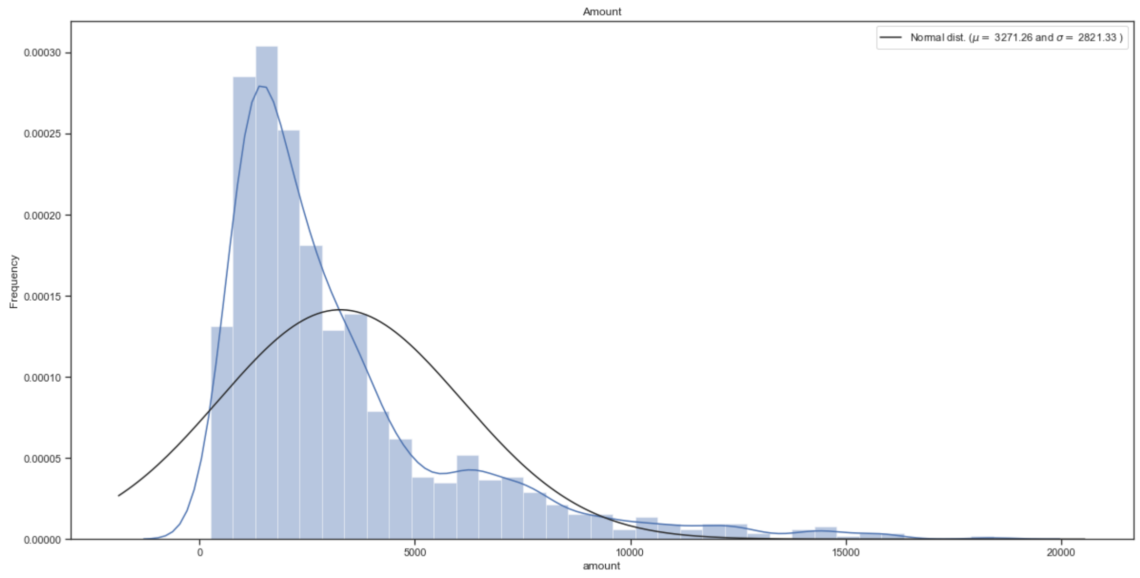


Figure 1: Before Scaling

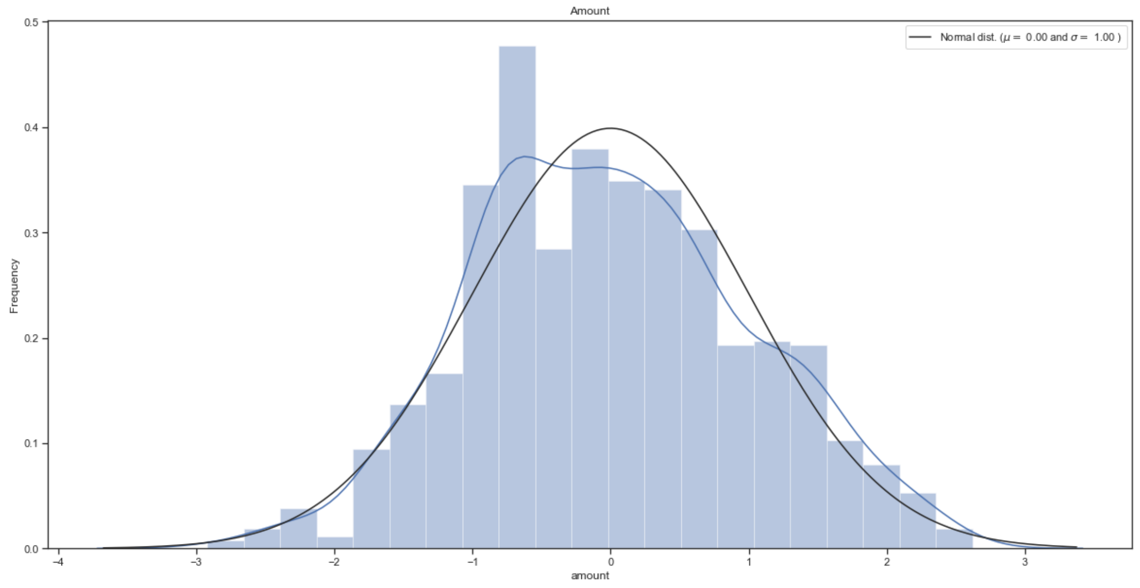


Figure 2: After Scaling

We can see that the data now is perfectly normalized. This will help us for our machine learning to avoid any outliers which may impact our training.
Our dataset contain many variables which are in their current state unusable for a machine learning training. Indeed, string values are not desirable. Consequently, we will be converting all non continuous variables (i.e. everything beside amount and monthlyamount) to dummy variables.
To make sure we do not have duplicate, we delete the previous column in our dataframe. We also remove monthlyamount column to not have redundancy in our data pattern with existing amount and duration variables.
We know have a considerable amount of different variables (70). We need to see which variables

may explain the dataset the best. To do so we use two techniques, explanation of variance (which is PCA) [11] and Multiple correspondence analysis (MCA) [12].
Multiple correspondence analysis is an extension of correspondence analysis (CA). It should be used when you have more than two categorical variables. The idea is simply to compute the one-hot encoded version of a dataset and apply CA (this computes the dependencies between categorical variables) on it.

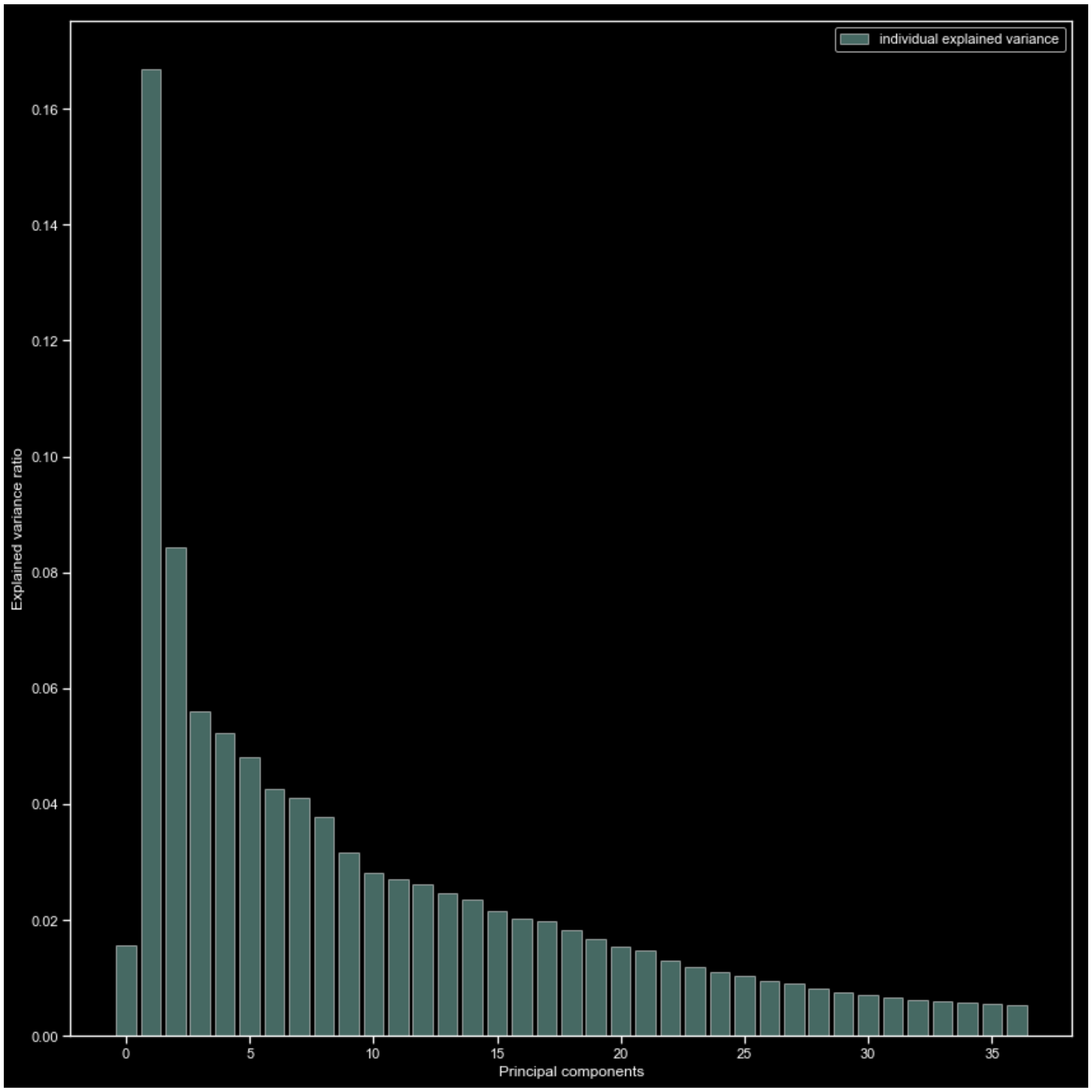


Figure 3: PCA

We can see here that we have more than 20% of our data which is explained by a single component, the rest of the variables do not explain the dataset as much as this one.

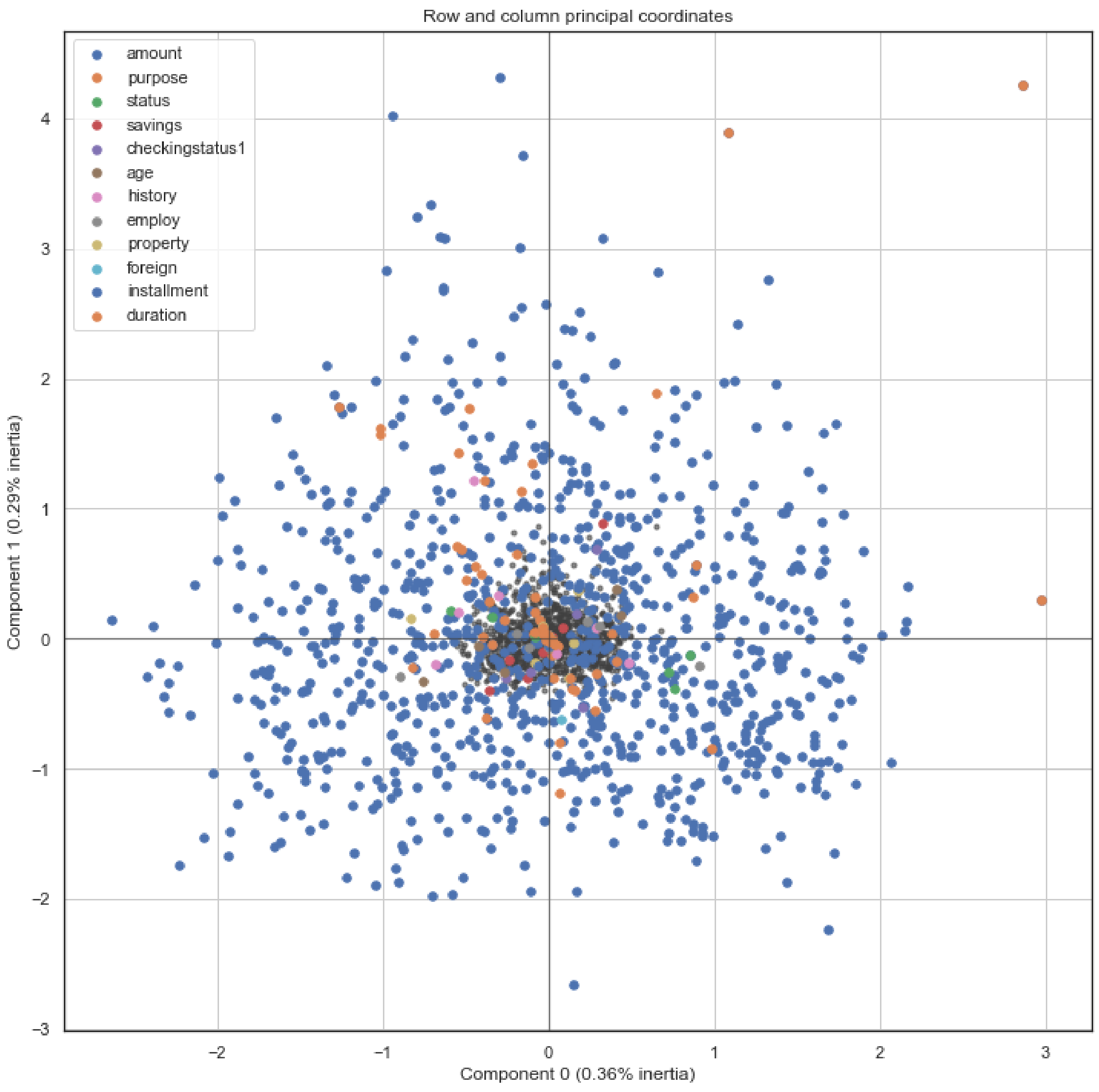


Figure 4: MCA

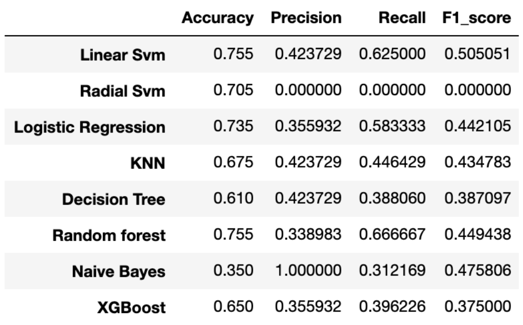
The MCA highlights this even more with amount being the most prominent variable there taking place in all dimensions.

We split our dataset randomly using a random seed for reproducible purpose. The split rate follows Pareto Law with a 80/20 split. We dropped "Default" from our training set to have it in our test set since it is our target variable.

Machine Learning - Significant Variables ONLY

In this part, we will be only working with the significant variables found thanks to multiple linear regression. The code showing the choice of variable is available in the Jupyter Notebook line 35 [1].

For the whole report and exercice we will using Random Forest and XGBoost with a depth of 100 and number of tree of 100.



	Accuracy	Precision	Recall	F1_score
Linear Svm	0.755	0.423729	0.625000	0.505051
Radial Svm	0.705	0.000000	0.000000	0.000000
Logistic Regression	0.735	0.355932	0.583333	0.442105
KNN	0.675	0.423729	0.446429	0.434783
Decision Tree	0.610	0.423729	0.388060	0.387097
Random forest	0.755	0.338983	0.666667	0.449438
Naive Bayes	0.350	1.000000	0.312169	0.475806
XGBoost	0.650	0.355932	0.396226	0.375000

To interpret this table we need to understand what does Accuracy Precision, Recall, F1 score mean [13].

Accuracy: Proportion of data that is actually true among data predicted to be positive or negative.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

There is a huge drawback with accuracy, it is only relevant when the target variables are balanced which is not the case in our dataset.

Precision: The percentage of data that is predicted to be positive that is actually positive.

$$Precision = \frac{TP}{TP + FP}$$

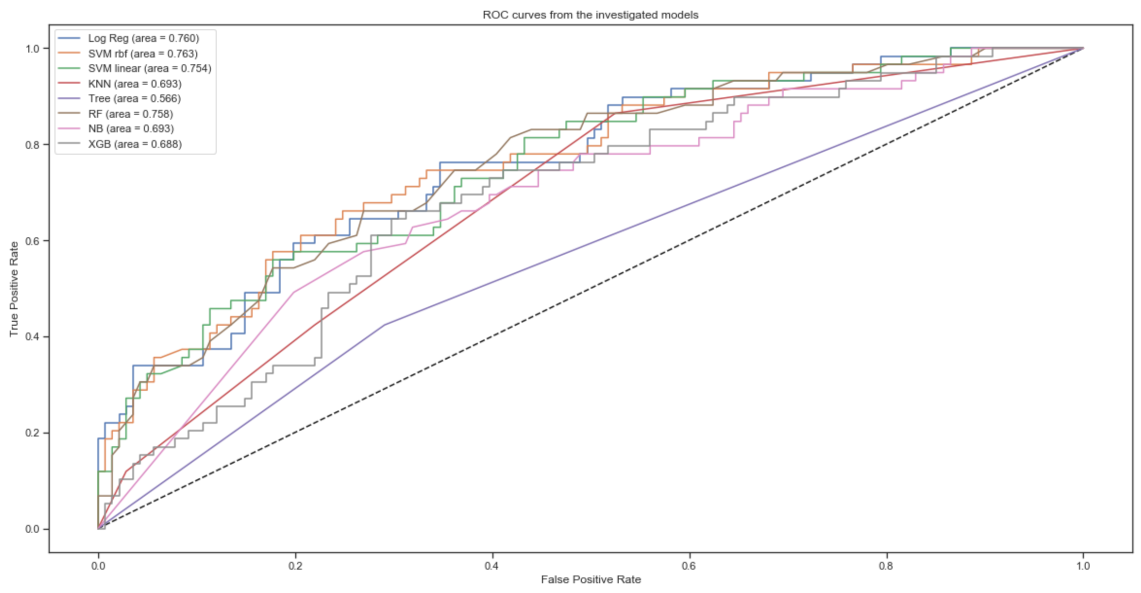
Recall: Proportion of those that are predicted to be positive among those that are actually positive

$$Recall = \frac{TP}{TP + FN}$$

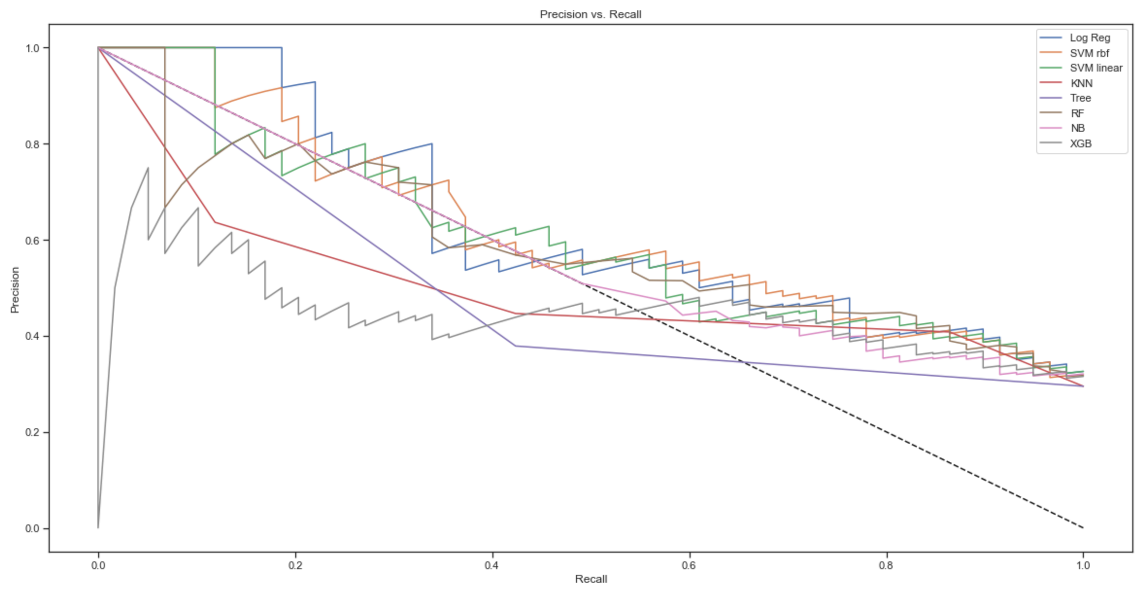
F1: Harmonic mean of accuracy and recall.

$$F1 = \frac{2Recall * Precision}{Recall + Precision}$$

In regards of our dataset the measure which interest us the most are Precision and Recall, out of both we want the model which gives us the best F1 score.
The initial results with only significant variables show that in regard of precision and recall the best models are Linear SVM, Naive Bayes and Random Forest.
If we look at the ROC curve and AUC score:



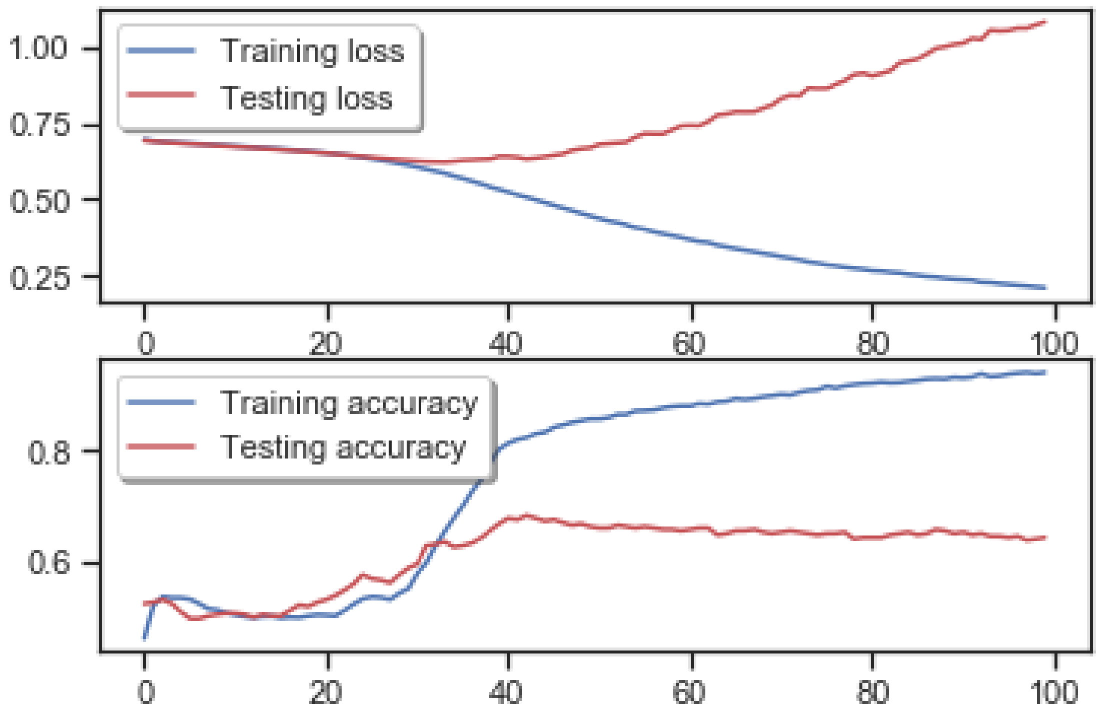
We can see that Linear SVM and Random Forest are among the best scorers. However we do not care about ROC that much. Indeed, we would use the precision-recall (PR) curve as a metric, rather than the ROC. True negatives will be the most abundant, and should probably not contribute to model performance. In this case, applying the label ”No Fraud” to all data has high level of accuracy due to imbalance in the dataset. The advantage of an ROC curve is that it is invariant to the balance or imbalance of the data. (By ”balance” we mean the fraction of positive cases.) So we don’t have to know about that imbalance in order to evaluate how well your classifier is doing, and we can have differently balanced data at different times: training and testing and fielded. So in some sense, the precision-recall curve is an interpretation of the same data as on an ROC curve, but making it very readable for one particular level of class balance. While the ROC curve abstracts away the class balance, therby making it applicable to all levels. To sum up, it is good evaluation of a model at first glance but we would rather use:



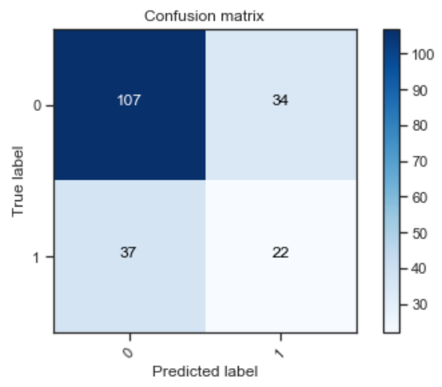
Precision versus recall. In this plot we can see how the models we chose outperform the others.
Therefore, for our analysis with significant variables only we believe that Random Forest and

Linear SVM and the best performing models.
As supplement, we also used Neural Network using Tensorflow by Google and Keras to analyze this dataset. It results the following:

```
200/200 [==============================] - 0s 46us/step
Test score: 1.084309782385826
Test accuracy: 0.6425
```



We can see that our neural network model is pretty mediocre. We see that the model accuracy hovers around 64% in the testing data which is not better than other models (accuracy may change slightly). We can try larger network with a higher number of epochs however, very likely this is the maximum prediction or performance of the model based on the dataset. We may also end of overfitting the training dataset. Again, note that the dataset is relatively small for neural network.
After about 30 epochs, testing loss starts to become higher than training loss and exactly opposite for accuracy: testing accuracy starts to get lower than training accuracy. Basically we are starting to overfit here. Therefore, we get an optimal accuracy of close to 65% using this model. It results the following confusion matrix:



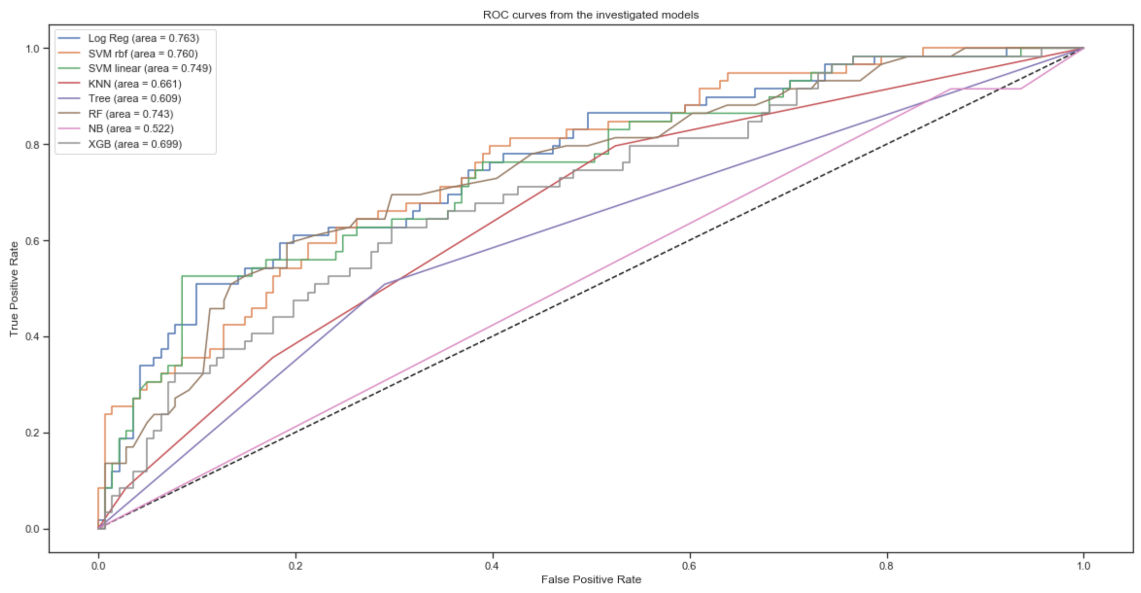
Which confirms our 60% accuracy. This confirms that Neural Network does not work well in our dataset compared to other models.

Machine Learning - ALL Variables

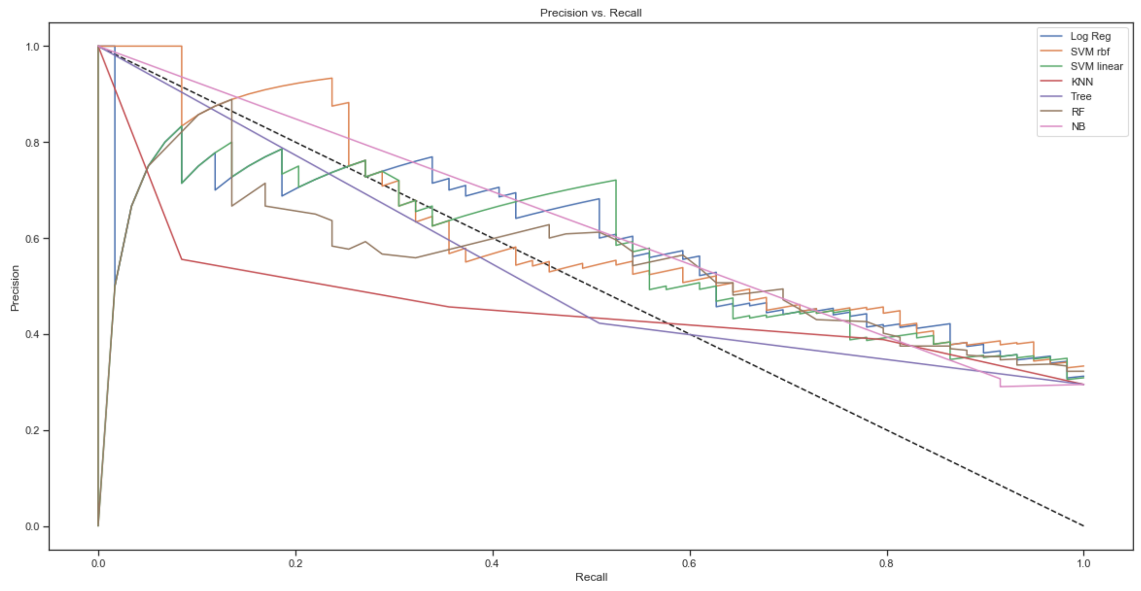
As a matter of comparison, we also decided to run the different models on a dataset which does not exclude too many variables like the previous one. So we kept the dataset intact, rescaled amount to be normalized and converted all variables to dummy variables.

	Accuracy	Precision	Recall	F1_score
Linear Svm	0.790	0.525424	0.688889	0.596154
Radial Svm	0.705	0.000000	0.000000	0.000000
Logistic Regression	0.775	0.474576	0.666667	0.554455
KNN	0.685	0.355932	0.456522	0.400000
Decision Tree	0.665	0.491525	0.424242	0.413223
Random forest	0.730	0.169492	0.666667	0.270270
Naive Bayes	0.325	0.915254	0.293478	0.444444
XGBoost	0.715	0.389831	0.522727	0.446602

Linear SVM is totally outperforming all the other models with a very high accuracy (compared to what we saw so far) and a high F1 score. It followed by Logistic Regression. Random Forest which was performing well before falls hard in precision compared to the previous dataset (significant variables only).

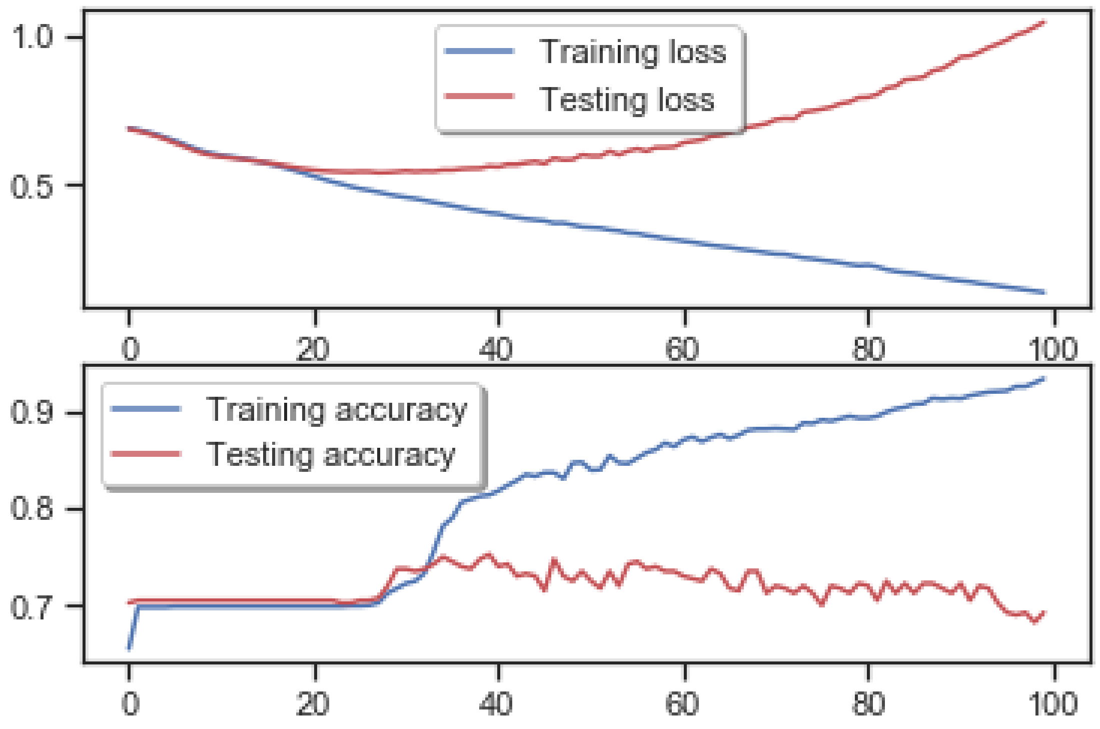


If we look at the ROC curve, we can see that the AUC for Linear SVM and Logistic Regression are among the best performing.

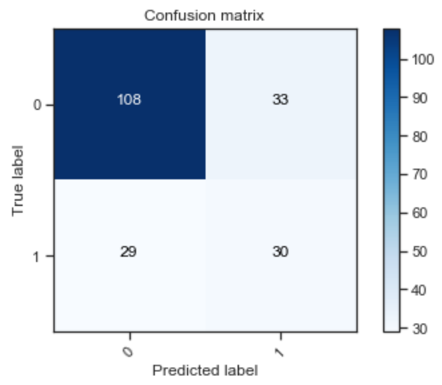


This is also confirmed by the Precision-Recall plot.

```
200/200 [==============================] - 0s 38us/step
Test score: 1.0476391124725342
Test accuracy: 0.6925
```



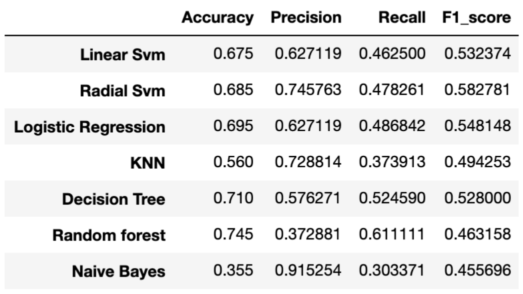
The neural networking is performing much better. As we said before, neural networking is performing well when there is more data which is the case here. We have 70% of accuracy, we notice that the number of epoch before the testing loss is lower than previously at 20 epochs.



The confusion matrix is better than 1 which is not significant. Consequently, for the over-sampling part we will be using our models on a dataset with significant variables only (we tested on both, this one is working better).

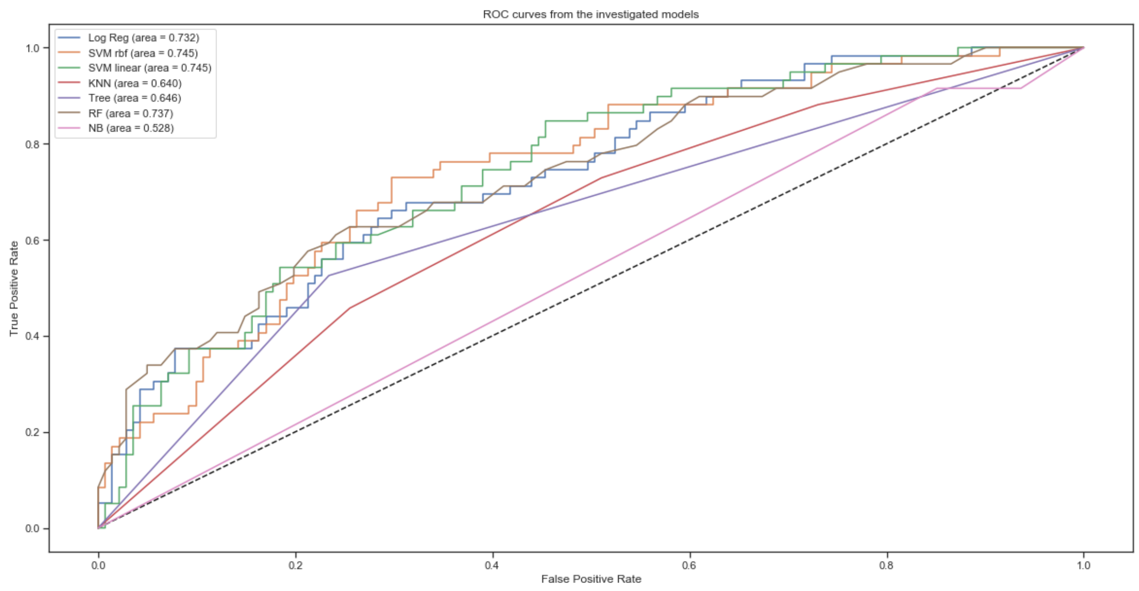
Machine Learning - SMOTE Oversampling

As highlighted previously, our dataset is highly unbalanced with a 70/30 distribution for the target variable. One of the way to solve this problem is to use SMOTE which stands for Synthetic Minority Over-sampling Technique [14]. SMOTE offers three options for generating samples. These methods focus on samples near the boundary of the optimal decision function and generate samples in the opposite direction of the nearest class. SMOTE uses KNN and calculates the distance between each vector then multiply them by a random number between 0 and 1 and add them back to the dataset .

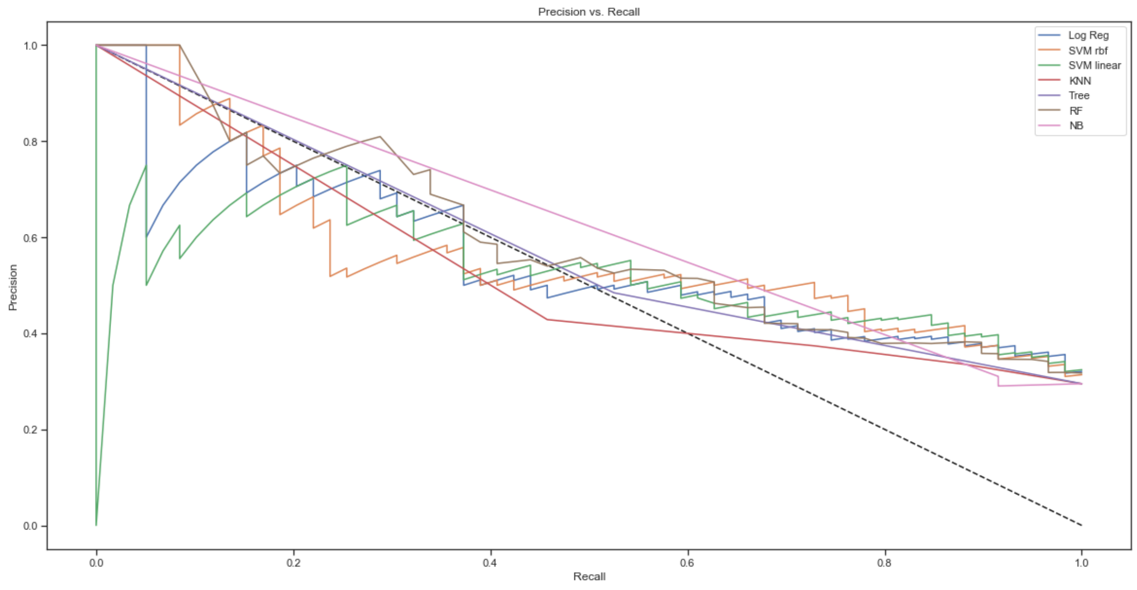


	Accuracy	Precision	Recall	F1_score
Linear Svm	0.675	0.627119	0.462500	0.532374
Radial Svm	0.685	0.745763	0.478261	0.582781
Logistic Regression	0.695	0.627119	0.486842	0.548148
KNN	0.560	0.728814	0.373913	0.494253
Decision Tree	0.710	0.576271	0.524590	0.528000
Random forest	0.745	0.372881	0.611111	0.463158
Naive Bayes	0.355	0.915254	0.303371	0.455696

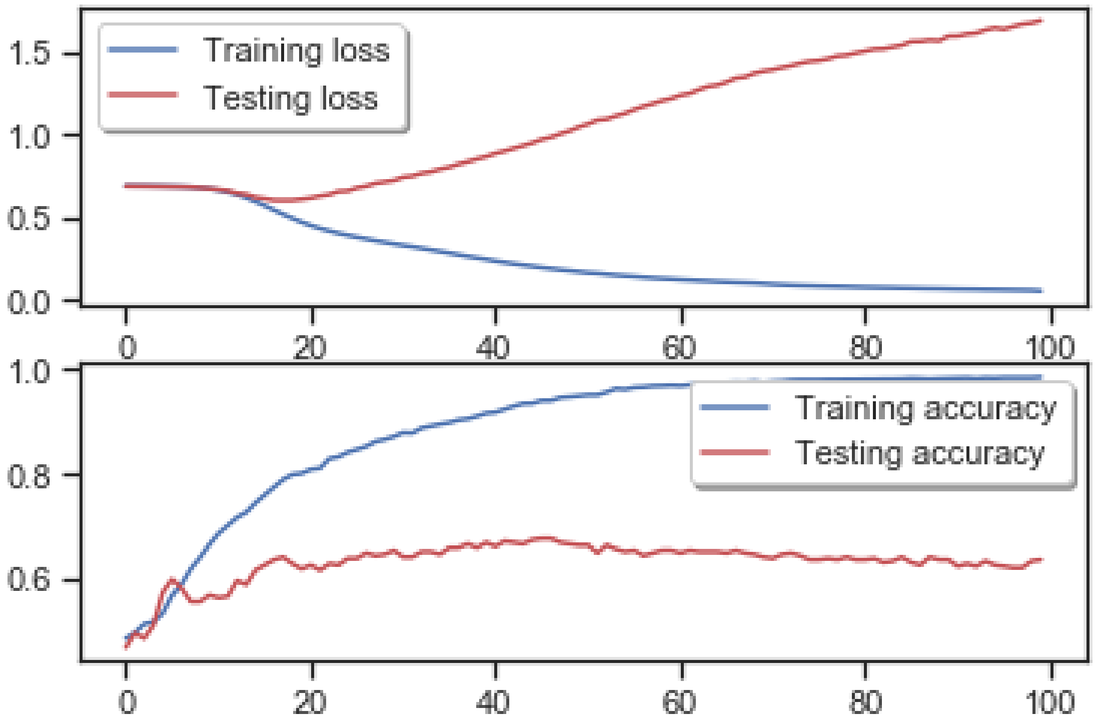
Our best models here are Radial SVM and Logistic Regression, we can now consider accuracy which will be less biased thanks to SMOTE oversampling technique, in that department the two best performing models are Decision Tree and Random Forest.



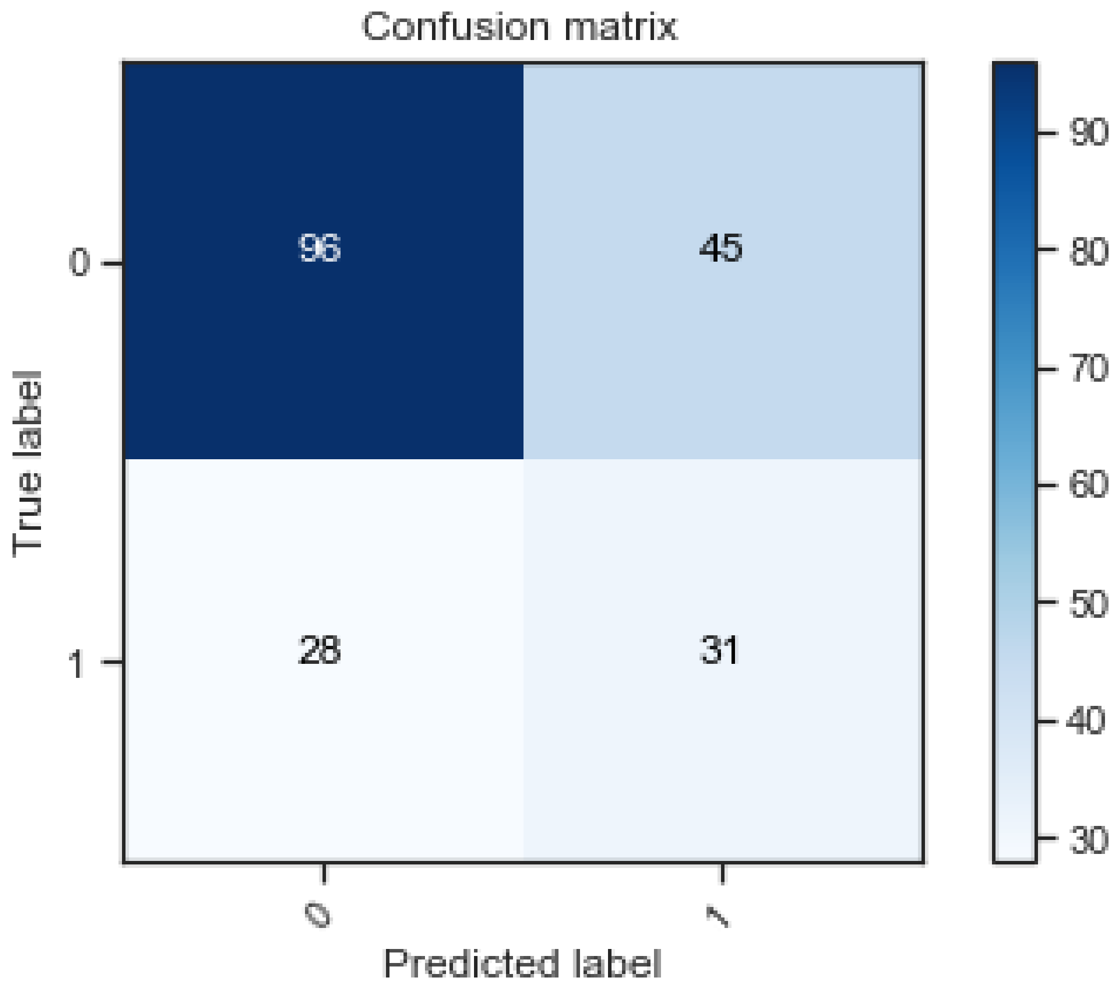
The ROC curve for the two best Recall-Precision models are still among the best. Random Forest has also a high AUC score but Decision Tree is much worse than the top performing ones.



The plot shows us that the models Radial SVM and Logistic Regression requires a somewhat decent recall value to have a pretty precision value.



We cannot compare the neural network of this dataset with the previous one (all variables). Nevertheless, by comparing it with the significant value one without oversampling, we can see that the overall performance is bad with 64% accuracy again. However, as we said before, we have more Default value, which means less bias in our accuracy number. With that fact, we have to consider the better performance of our neural network over this dataset which results in the following confusion matrix.



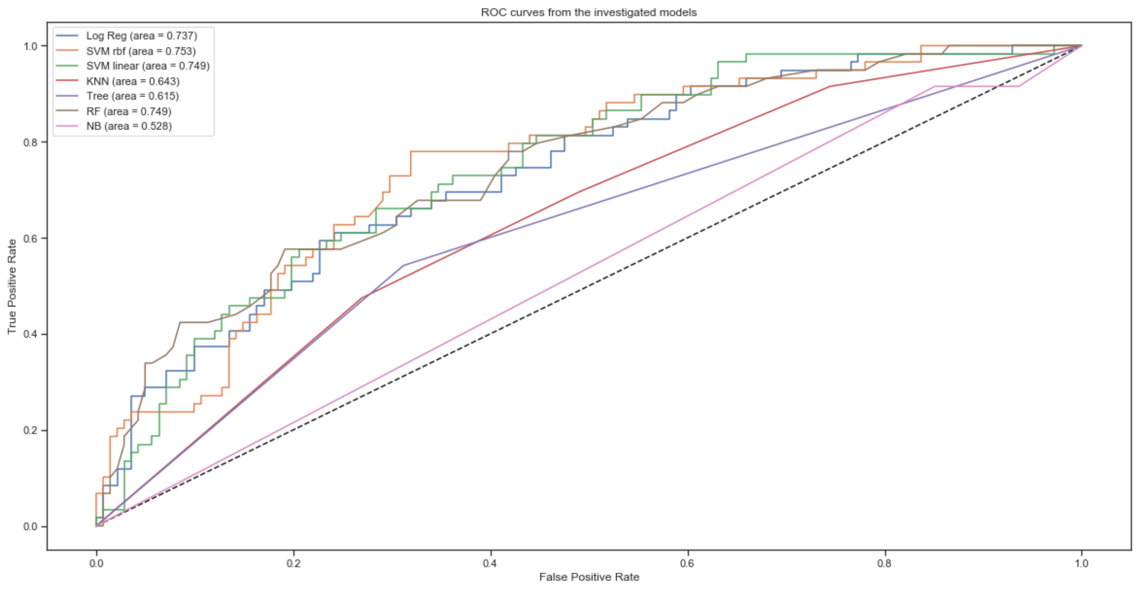
Less TP, but less bias. The performance is still not good as expected.

Machine Learning - ADASYN Oversampling

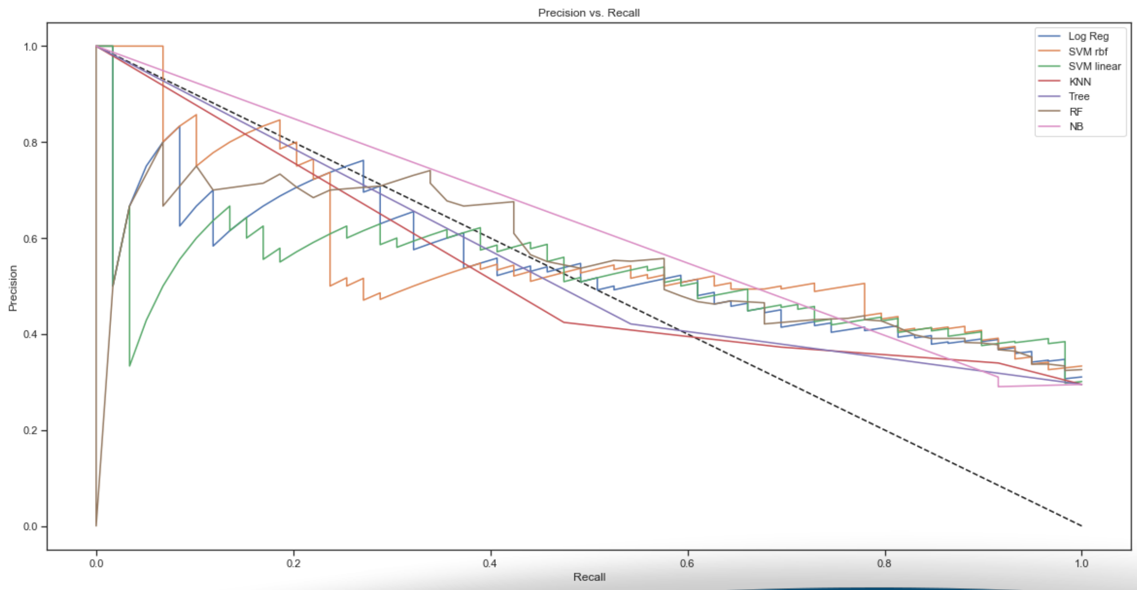
There are different oversampling techniques. We just saw how SMOTE performs. We will be using ADASYN (Adaptive Synthetic Sampling Approach For Imbalanced Learning) as oversampling technique this time on the significant variables dataset [15]. ADASYN use KNN to increase according to the distribution of minority data. ADASYN uses a density distribution, as a criterion to automatically decide the number of synthetic samples that must be generated for each minority sample.
Let's see how it performs:

	Accuracy	Precision	Recall	F1_score
Linear Svm	0.705	0.610169	0.500000	0.549618
Radial Svm	0.705	0.779661	0.500000	0.609272
Logistic Regression	0.710	0.610169	0.507042	0.553846
KNN	0.565	0.694915	0.372727	0.485207
Decision Tree	0.660	0.525424	0.410959	0.447761
Random forest	0.760	0.355932	0.677419	0.466667
Naive Bayes	0.355	0.915254	0.303371	0.455696

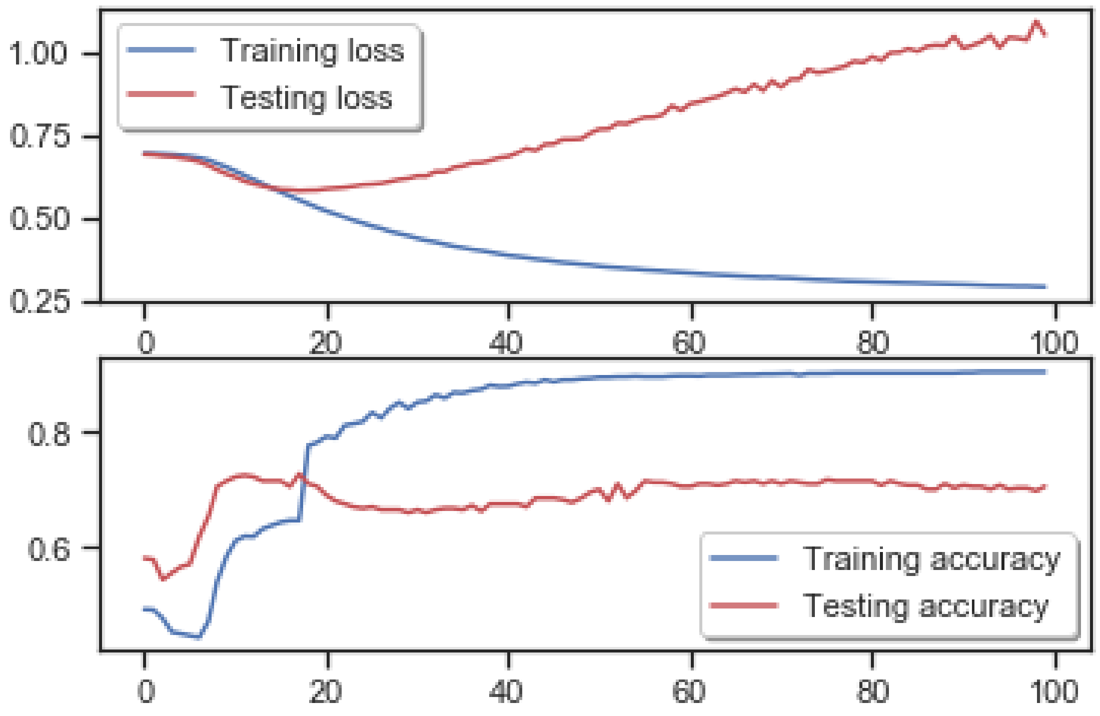
We see a larger large discrepancy between high accuracy models and high F1 score. For accuracy, Random Forest is by far the best model but its F1 is low. If we stick to F1 score we can see that Radial, Linear SVM and Logistic Regression are out performing the others.



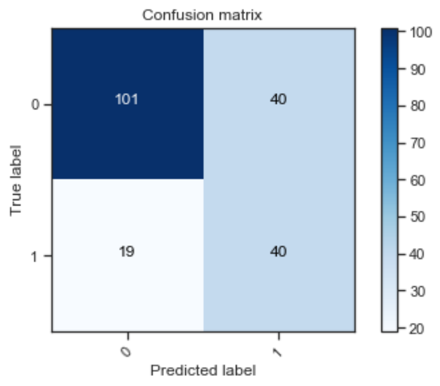
These performances are confirmed by the high AUC score of all these models compared to the others.



The precision-recall plot highlights us well the best performing F1 score models and confirm their high recall and precision values compared to the others.



Surprisingly, ADASYN increases the accuracy scoring of our neural network by a lot. We now have 70% accuracy which is the highest so far. We also notice that the number of epoch before testing loss is less than what we had so far with a loss at around 18 epochs.



The confusion matrix contributes to show us that for neural network ADASYN is performing the best compared to SMOTE in regard to accuracy.

Discussion and Results

We compared all different models on different datasets. We found out that overall the best performing models regardless of the dataset are SVM (linear mostly) and Logistic Regression.

We believe that due to the small amount of data, more complicated models such as XGBoost (ensemble model) or Random Forest are not in their best environment to generate relevant insights.
Consequently, the models to use to optimize Precision and Recall regardless of the dataset settings for this exercice are:
- Linear SVM
- Logistic Regression
For accuracy, we have to give credits to Random Forest which performed well overall.
All these models have a high AUC score compared to the others and therefore, this also leads to the same conclusion for ROC analysis

Business Point

There are some key takeaway for the company. First, they need to know what they want to optimize, if they want to limit the number of wrong data analysis they should use accuracy as main metrics. If they want to be sure they have given true data for the one who are in default they should use F1 score (which balances both recall and precision).
From this, if they want to limit their loss they need to have a scope in mind and a roadmap which includes what they want to do with the data and not going blindly with data analysis.

Limitations

The first limitation is the number of data available, 1000 rows to work with is very low and is totally not exploitable in the use of a neural network (which was not part of the exam though). The second limitation was time constraint, we were not able to add a cross validation in our code to improve the model scoring [16].

Conclusion

Credit fraud is a serious problem, which should be taken lightly. Therefore companies need to know which information they want to highlight before considering working with different models.
Indeed, with the data provided, models behave distinctively and have their niche use.

If the company wants to optimize accuracy, they should use Random Forest.

If the company wants to work with recall and precision they should use SVM and logistic regression.

From these insights we can state that business understanding is the key information in interpreting results for data analysis.

Code (text is an URL)

Jupyter Notebook

Github

References

[1] Khieu. Analytical modeling. 2019.

[2] Raymond H Myers and Raymond H Myers. *Classical and modern regression with applications*, volume 2. Duxbury press Belmont, CA, 1990.

[3] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

[4] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[5] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.

[6] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[7] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[8] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[9] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[10] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, et al. Api design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*, 2013.

[11] Lindsay I Smith. A tutorial on principal components analysis. Technical report, 2002.

[12] Hervé Abdi and Dominique Valentin. Multiple correspondence analysis. *Encyclopedia of measurement and statistics*, 2:651–66, 2007.

[13] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[14] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[15] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328. IEEE, 2008.

[16] Mustakim Al Helal, Mohammad Salman Haydar, and Seraj Al Mahmud Mostafa. Algorithms efficiency measurement on imbalanced data using geometric mean and cross validation. In *2016 International Workshop on Computational Intelligence (IWCI)*, pages 110–114. IEEE, 2016.