Project: Creditworthiness

Step 1: Business and Data Understanding

A small bank office has received a great increase in the number of loan petitions. Usually, they manage 200 per week, but they are expecting to receive 500 next week. The manager sees a great opportunity to start automating the process and wants to create a model capable of differentiating between those that will be approved and those that will be rejected.

Therefore, a model will be trained using previous loan classification data, meaning that we will use a predictive model. It will be binary as it will decide if a petition will be admitted or dismissed.

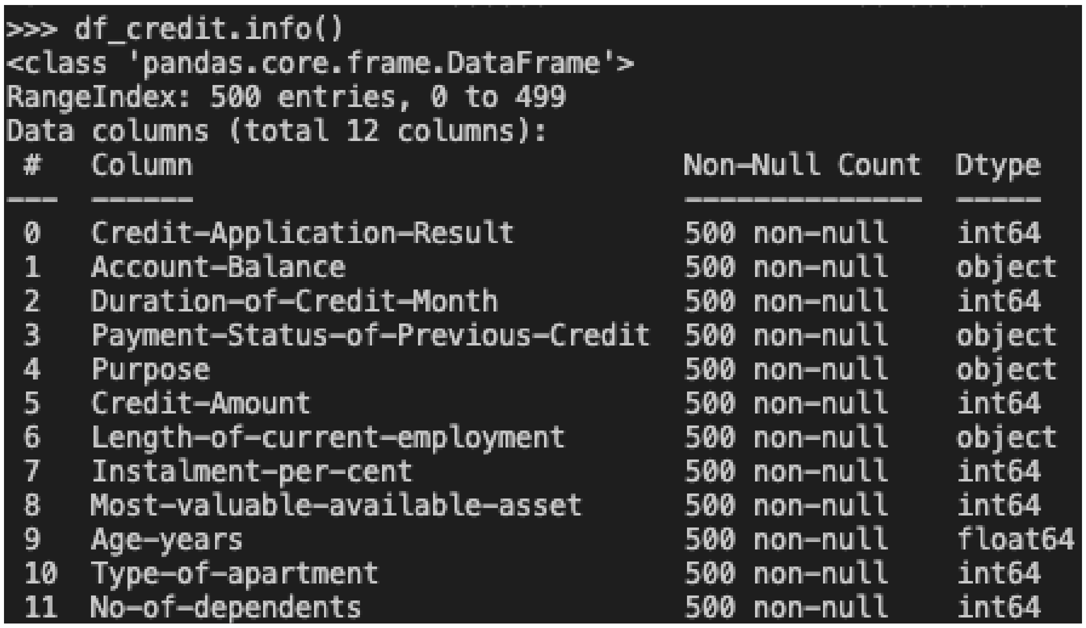
- **What decisions do they need to be made?**
  It has to be decided if a loan petition should be approved or declined.

- **What data is needed to inform those decisions?**
  The model needs data from previous loan petitions in order to understand which are the most important parameters for making the decision

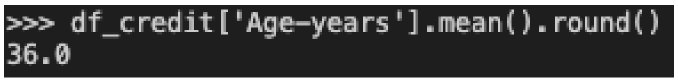
- **What kind of model (Continuous, Binary, Non-Binary, Time-Series) do we need to use to help make these decisions?**
  The model must be a **binary model** as there are only 2 options, approved or declined.

Step 2: Building the Training Set

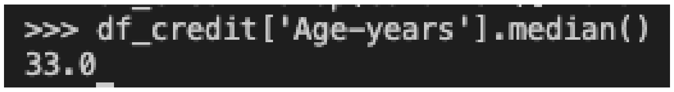
The cleaning process is defined in the python script used for making this project. However, below are screenshots to show the fields selected and their values:



```
>>> df_credit.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 12 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   Credit-Application-Result          500 non-null    int64
 1   Account-Balance                    500 non-null    object
 2   Duration-of-Credit-Month           500 non-null    int64
 3   Payment-Status-of-Previous-Credit  500 non-null    object
 4   Purpose                            500 non-null    object
 5   Credit-Amount                      500 non-null    int64
 6   Length-of-current-employment       500 non-null    object
 7   Instalment-per-cent                500 non-null    int64
 8   Most-valuable-available-asset      500 non-null    int64
 9   Age-years                          500 non-null    float64
 10  Type-of-apartment                  500 non-null    int64
 11  No-of-dependents                   500 non-null    int64
```



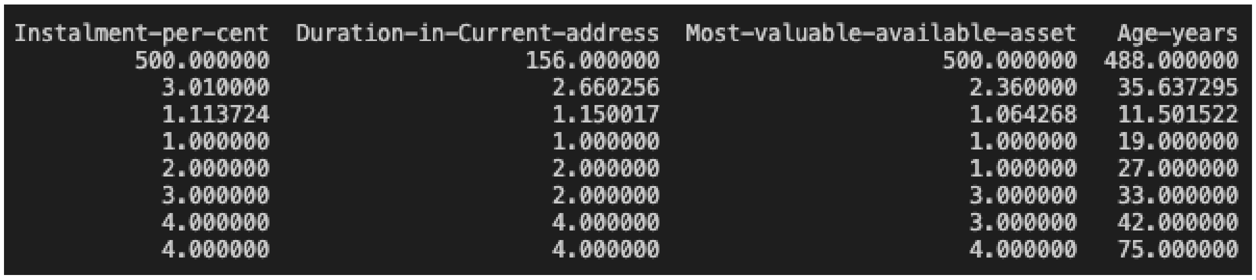
```
>>> df_credit['Age-years'].mean().round()
36.0
```



```
>>> df_credit['Age-years'].median()
33.0
```

How do we end up with these columns?

The process started exploring the columns in the dataset. First thing we realized is that 2 of the fields have Null values:

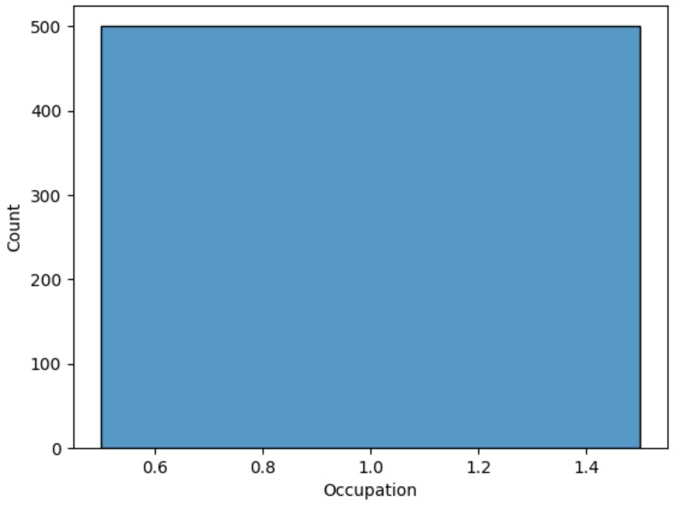


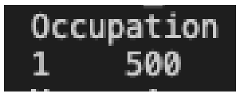
```
Instalment-per-cent  Duration-in-Current-address  Most-valuable-available-asset   Age-years
         500.000000                   156.000000                     500.000000  488.000000
           3.010000                     2.660256                       2.360000   35.637295
           1.113724                     1.150017                       1.064268   11.501522
           1.000000                     1.000000                       1.000000   19.000000
           2.000000                     2.000000                       1.000000   27.000000
           3.000000                     2.000000                       3.000000   33.000000
           4.000000                     4.000000                       3.000000   42.000000
           4.000000                     4.000000                       4.000000   75.000000
```

“***Duration-in-Current-address***” contains less than half of the values, total is 500, so the decision was to directly remove it. However, “***Age-years***” only misses 12 rows out of 500 and looks like a relevant one. Thus, we fill them using the column median value, in this case 33.

The next step was to identify categorical parameters which only have 1 option or that are really unbalanced. We explore the data frame grouping different columns and count the number of rows per each column value. The result was that these columns were dropped:

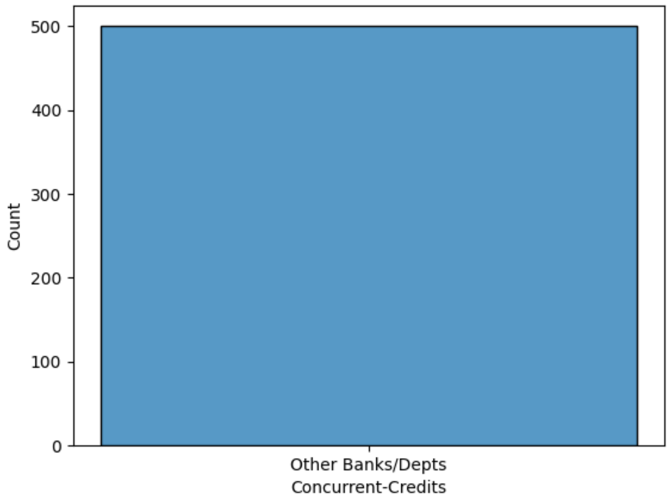
- **Occupation:**

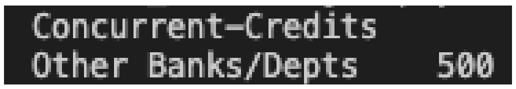




```
Occupation
1    500
```

- **Concurrent-Credits:**

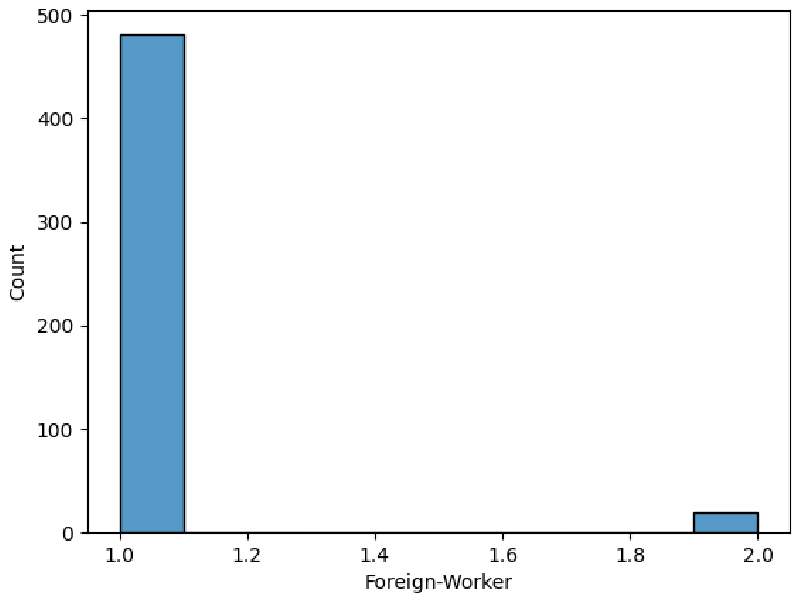


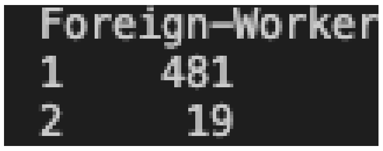


```
Concurrent-Credits
Other Banks/Depts    500
```

We also found other categorical columns that must be dropped. These contain 2 values, but they are really unbalanced so they will bias our model. As before, we attach some charts to prove this point:

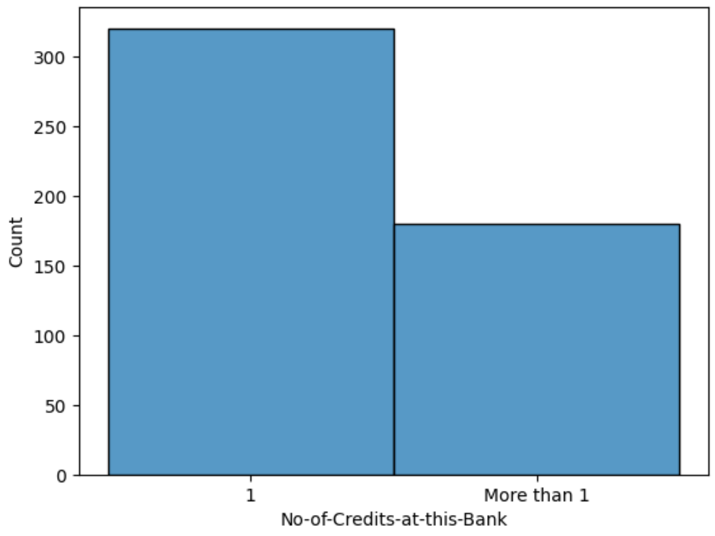
- **Foreign-Worker**

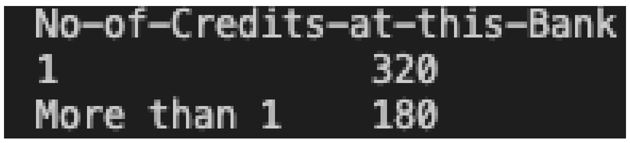




```
Foreign-Worker
1    481
2     19
```

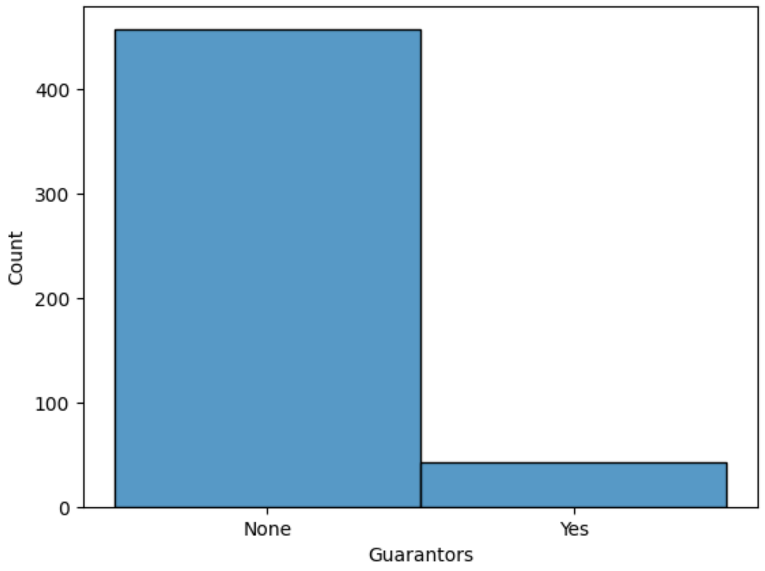
- **No-of-Credits-at-this-Bank:**

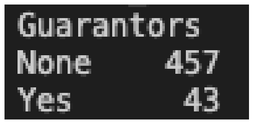




```
No-of-Credits-at-this-Bank
1                 320
More than 1       180
```

- **Guarantors:**

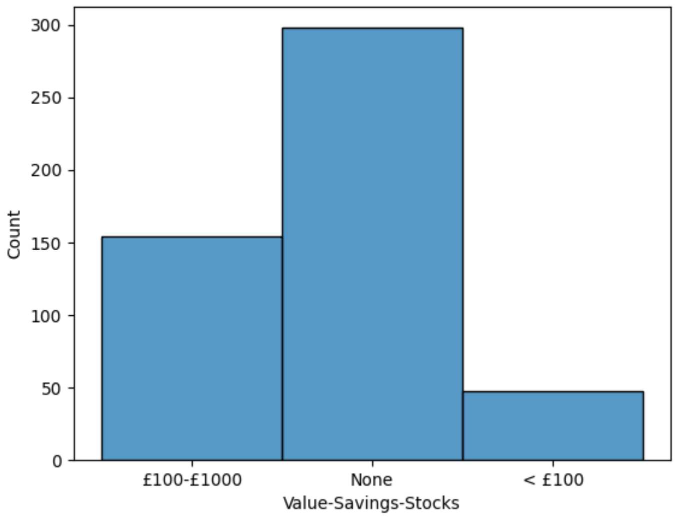


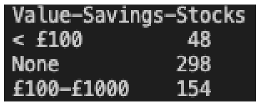


```
Guarantors
None    457
Yes      43
```

One interesting fact that happened when exploring the rest of the columns is that we found another column with a lot of empty values. However, these were not Null values, they were strings with values as "None".
Of course, even when it's a proper value, it's not providing information and, because of that, they were also removed.

- Value-Savings-Stocks:





```
Value-Savings-Stocks
< £100         48
None          298
£100-£1000    154
```

Step 3: Train your Classification Models

For training the model we transform the data to be all integer parameters. This means that we will create dummy variables from all categorical columns and drop one of each. This is all done with pandas ".get_dummies()" function and dropping the following values/dummy columns:

- From **Payment-Status-of-Previous-Credit**, the value dropped: "*No Problems (in this bank)*"
- From **Length-of-current-employment**, the value dropped: : "*< 1yr*"
- From **Account-Balance**, the value dropped: "*No Account*"

Next, we set our **x** and **y** datasets, being x the whole dataset minus the dependent variable ("Credit-Application-Result") and **y** is just this column. Finally, using the function train_test_split() from the sklearn module we create the datasets for training and testing our models, being 70 and 30% respectively.

**For the logistic regression model we use the statsmodels package as it allows us to see parameters of p-values.*
***The column "*Credit-Application-Result" is another categorical column and we need to transform it into numeric values, 1s and 0s. There are 2 ways:

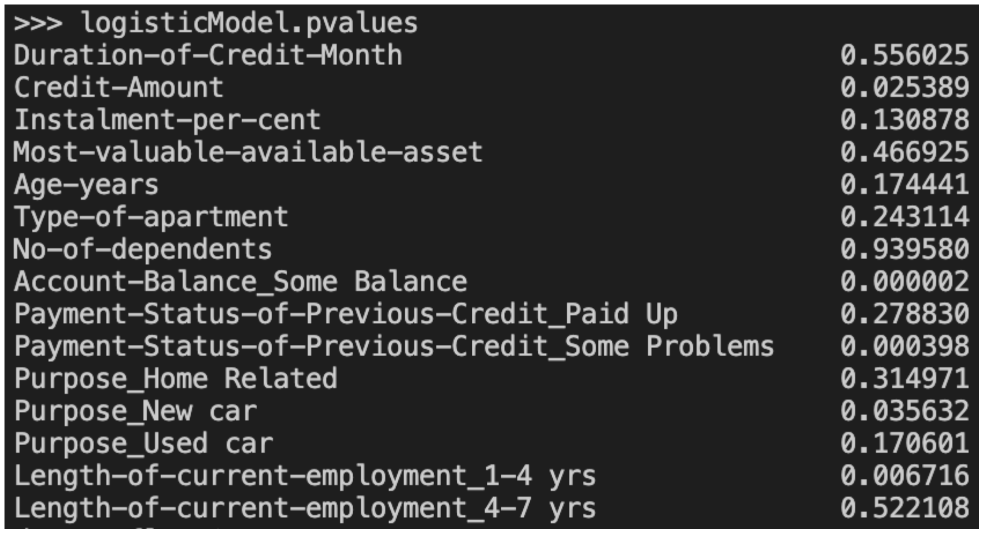
- Create dummy variables and drop one, as we did before-
- Replace values to 1s and 0s.

Finally, we create and fit the model with our data.

For all the models, we have answered the following questions:

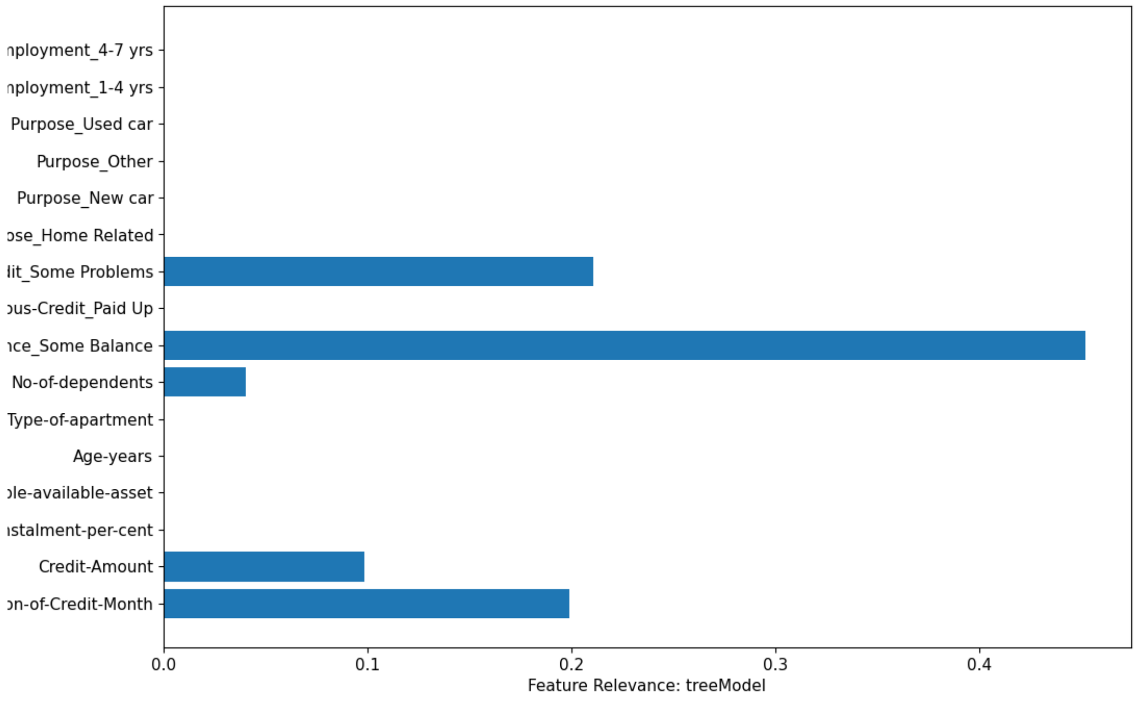
Which predictor variables are significant or the most important? Please show the p-values or variable importance charts for all of your predictor variables.

Logistic Regression:

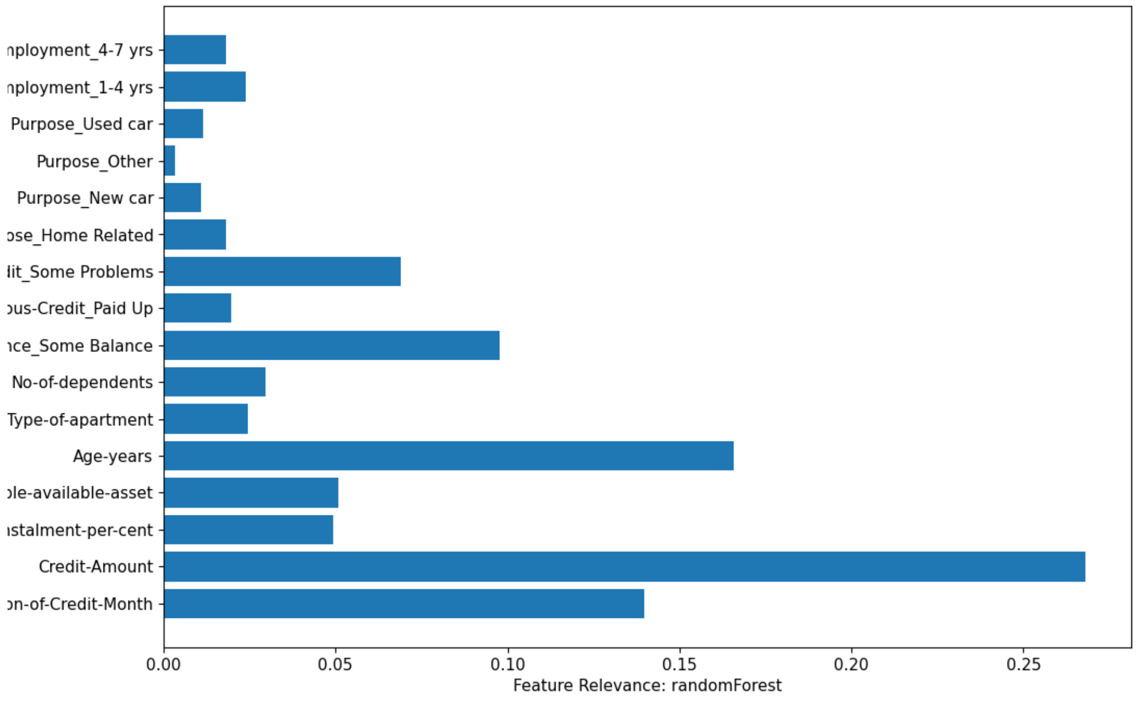


```
>>> logisticModel.pvalues
Duration-of-Credit-Month                              0.556025
Credit-Amount                                         0.025389
Instalment-per-cent                                   0.130878
Most-valuable-available-asset                         0.466925
Age-years                                             0.174441
Type-of-apartment                                     0.243114
No-of-dependents                                      0.939580
Account-Balance_Some Balance                          0.000002
Payment-Status-of-Previous-Credit_Paid Up             0.278830
Payment-Status-of-Previous-Credit_Some Problems       0.000398
Purpose_Home Related                                  0.314971
Purpose_New car                                       0.035632
Purpose_Used car                                      0.170601
Length-of-current-employment_1-4 yrs                  0.006716
Length-of-current-employment_4-7 yrs                  0.522108
```

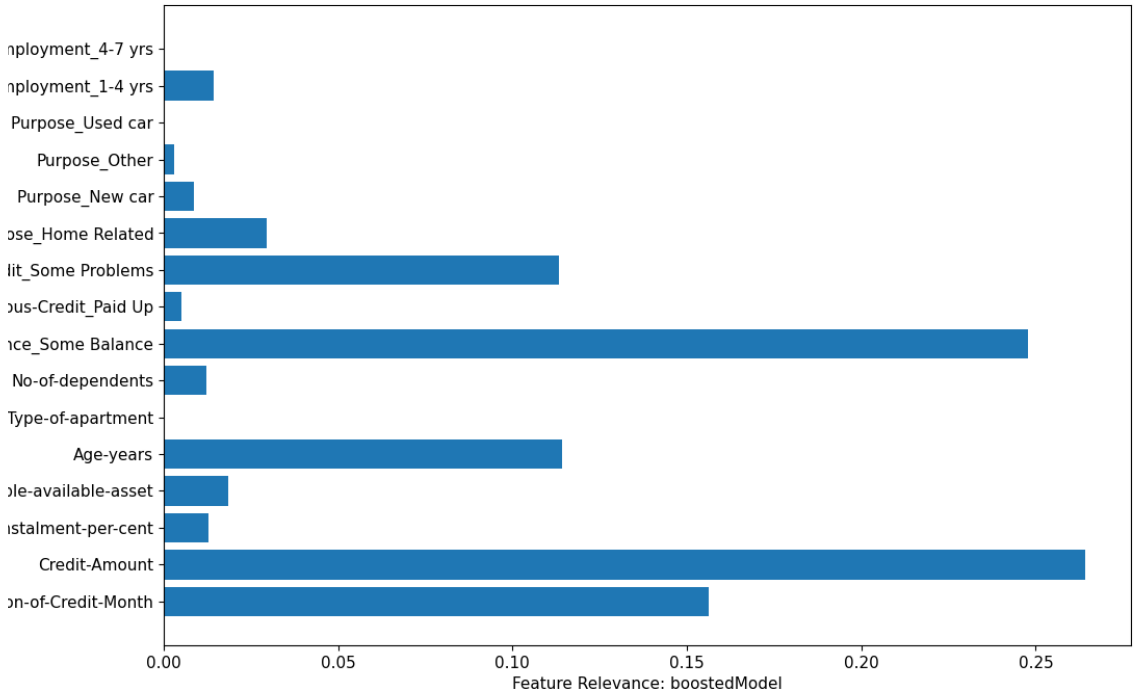
- Tree classifier:



- Random Forest:



- Boosted Model:



Each model gives more relevance to different columns, but there are some of them that all of them give some relevance:

- ***Credit Amount.***
- ***Duration of credit month.***
- **Payment Status of previous Credit_Some_problems**
- **Account Balance_Some_balance**

Also, it's interesting to check that the *p-values* less than 0.05 in the logistic regression model are:

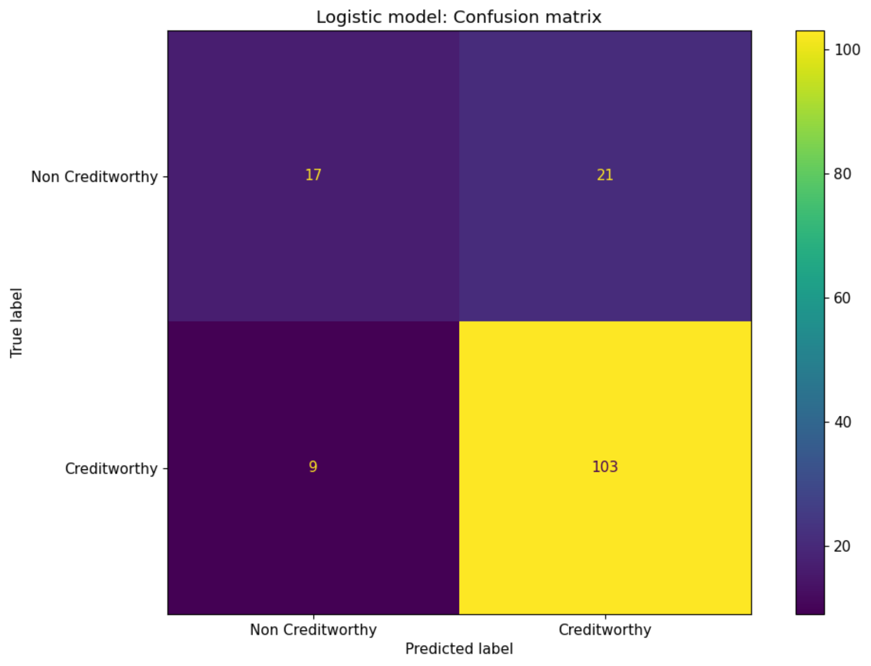
- **Account Balance**: Some_balance
- **Payment Status of previous Credit**: All of the dummy variables
- **Purpose**: All of the dummy variables

We now have a better understanding of the features in our dataset. The **Credit Amount** has shown a strong relevance in most of the models, except in the logistic regression where the p-value is over 0.05. In addition, **Account Balance** and **Payment Status of Previous Credit** have strong relevance in some of the dummy variables, but, again, differences between models is really huge.

With this in mind, I think it is better to choose which is the best model for the task and, later, select the features for optimyze the dataset. So, let's compare the 4 models using a confusion matrix and accuracy after validating it against our test datasets.

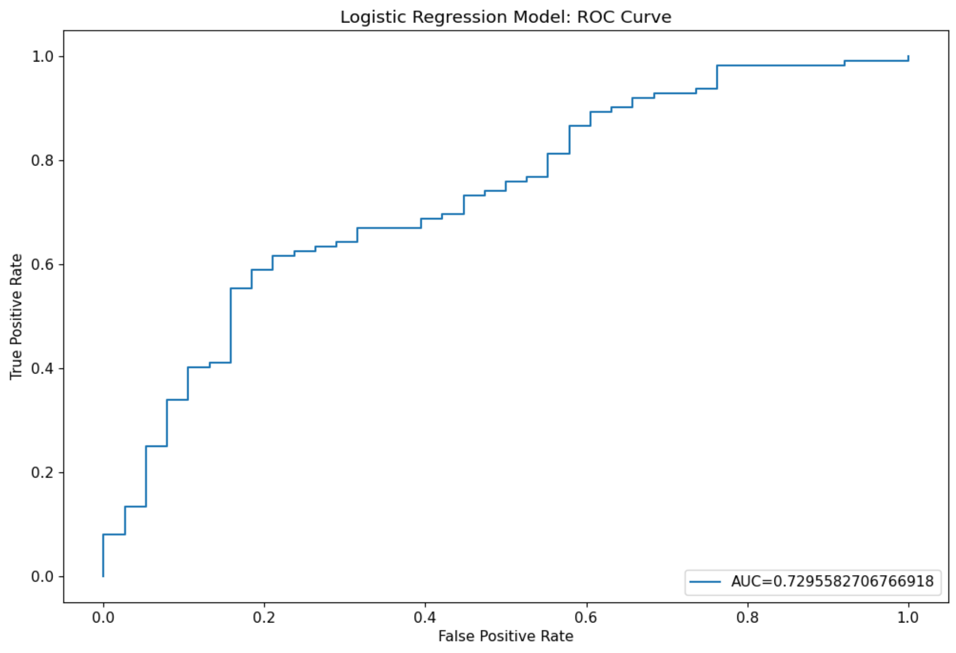
Logistic Regression Model

- **Confusion Matrix**



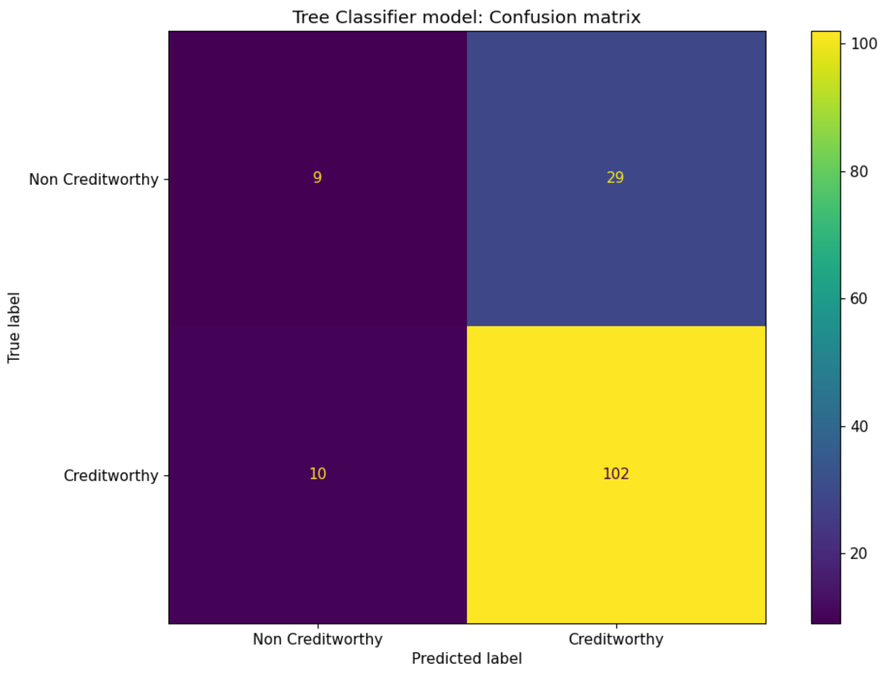
- **Accuracy score**
  - Overall accuracy percentage: **76,67%**
  - Non-Creditworthy accuracy percentage:**44,74%**
  - Creditworthy accuracy percentage: **91,96%**

- **ROC Curve**



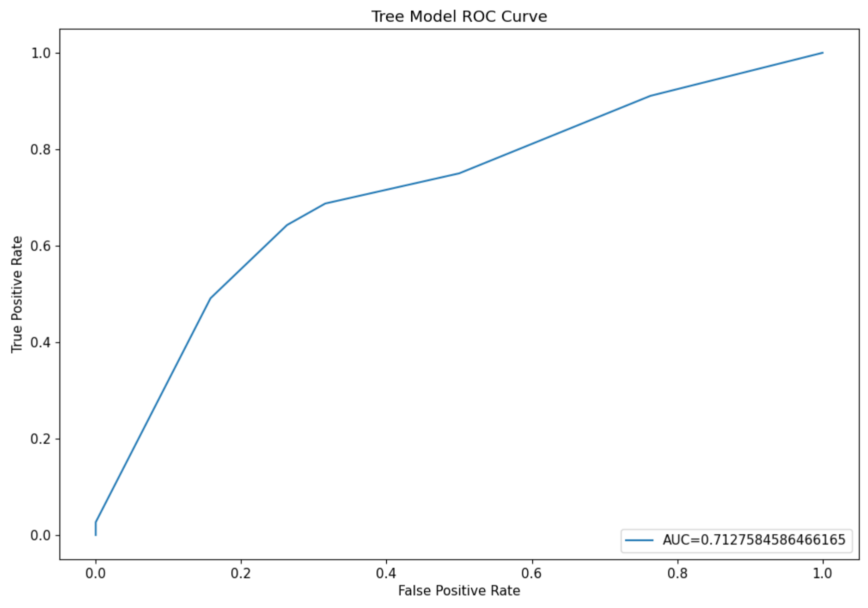
Tree Classifier Model

- **Confusion Matrix**



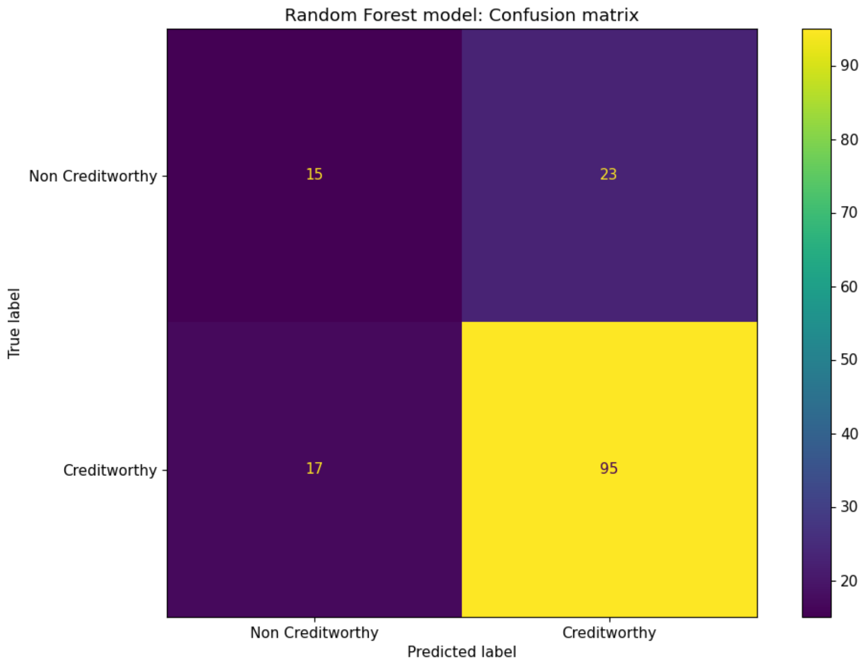
- **Accuracy score**
  - Overall accuracy percentage: **74,00%**
  - Non-Creditworthy accuracy percentage:**23,68%**
  - Creditworthy accuracy percentage: **91,07%**

- **ROC Curve**



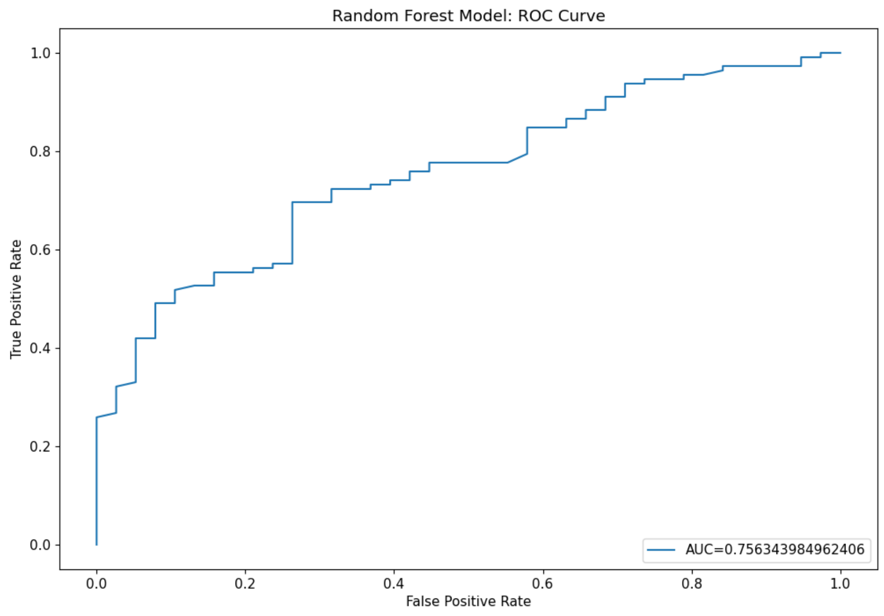
Random Forest Model

- **Confusion Matrix**



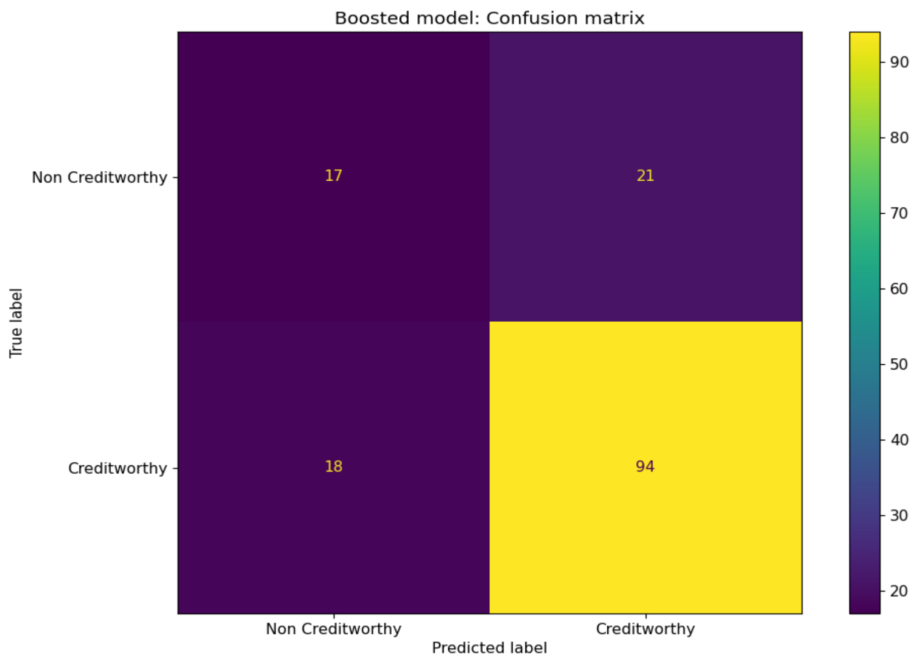
- **Accuracy score**
  - Overall accuracy percentage: **73,33%**
  - Non-Creditworthy accuracy percentage: **39,47%**
  - Creditworthy accuracy percentage: **84,82%**

- **ROC Curve**

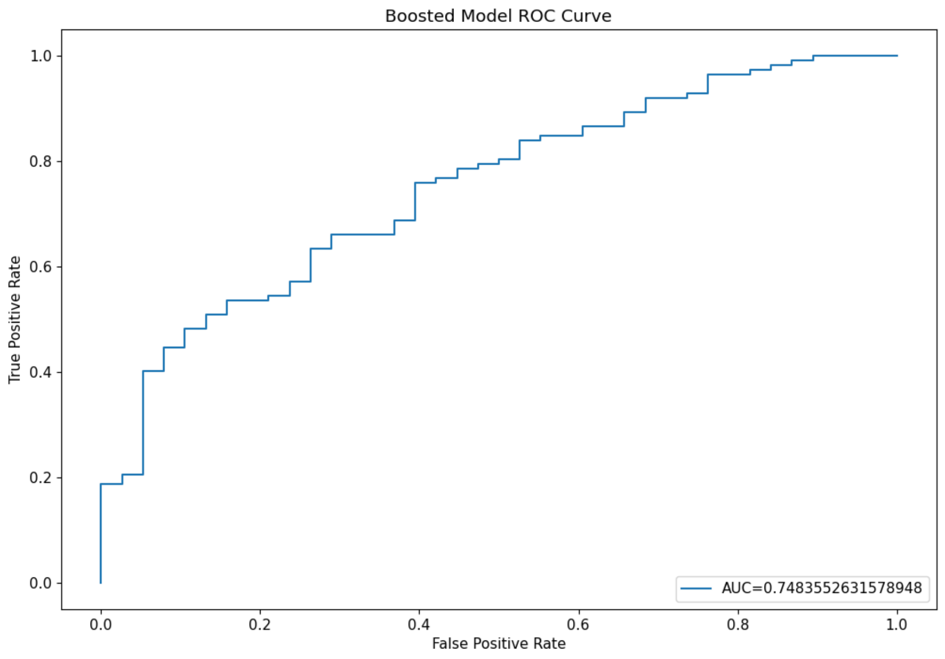


Boosted Model

- **Confusion Matrix**



- **Accuracy score**
  - Overall accuracy percentage: **74%**
  - Non-Creditworthy accuracy percentage: **44,74%**
  - Creditworthy accuracy percentage: **83,93%**
- **ROC Curve**



Step 4: Writeup

The model with the best global accuracy is the **Logistic Regression**, with a **76.67%.** This is also true when we review the accuracy within each segment, having **44.74%** and **91.96%** for Non-Creditworthy and Creditworthy respectively. The other models are not that far from these values, but it's true that some of them have great gaps when talking about the Non-Creditworthy segment.

When we analyse the ROC Curve, the Area Under the Curve (AUC) for the Logistic Regression Model is the lowest. This indicates that this model could not be the best one based on this technique, although all models are about 0,7 and not big differences between any them.

Based on the confusion matrix, it's clear that we have a bias towards the Creditworthy segment. If we sum up the number, there are 112 creditworthy users and only 38 non-creditworthy. This a significant difference that clearly has affected the models when they have to predict if a loan is non-creditworthy and, on the other hand, could also mean that we could overestimate the number of creditworthy users. ***This is a risk if using any of the models and the dataset must be balanced in order to look for improvements.***

With all of this in mind and considering the business case priorities (our boss needs), accuracy is the most relevant; even more if we talk about accuracy within the segments. Therefore, the model we chose it's the **Logistic Regression Model.**

Using this model over the new users data, we obtained the following results, after being rounded**:

- Number of Creditworthy users: **399**
- Number of Non-Creditworthy users: **111**

***The model calculates the probability to be Creditworthy [P(Creditworthy)]. As said, when this priority is 0,5 or over, it's labelled as Creditworthy. The logic applied for using this method is that the probability of being Non-Creditworthy [P(Non-Creditworthy)] is 1- P(Creditworthy), so, when the prediction is 0.6, the P(Non-Creditworthy) is 0.4.*

Considerations

As this is a training project I did not iterate over the features or adjusted the dataset, but there are a few things