Vaasan yliopisto
UNIVERSITY OF VAASA

Jere Korhonen c115062, Reyver Serna d119652,
Emmanuel Seakomo e123612

Applying Machine Learning Models to Credit Risk Data

Performance comparison between Logistic Regression, Artificial Neural Network & Random Forest

School of Technology and Innovations
Applied Machine Learning ICAT3210
Project Work

Vaasa 2021

Sisällys

1 Introduction

Banks and other financial institutions often receive plenty of loan applications from their customers. The high degree of uncertainty present in these applications represents a challenge for these organizations because they must put a lot of resources to predict if the customer will be able to pay the loan back. For that reason, banks and other lenders base their decisions on the customer's aspects such as past transactions, income, savings, assets… A customer is a low risk if the customer has a history of paying back previous loans. On the contrary, she is labeled as a high-risk if she has defaulted some of her duties.

When analyzing the data, these organizations need to learn the type of class customer to take the best decision when a new application arrives. Thanks to the development of Machine Learning, ML, in banking and financial services, nowadays these organizations can build credit risk models that predict the credibility of a customer automatically based on observable data. According to Alpaydin (2014) financial institutions find themselves in a need for accurate and effective models that lead them to make decisions to increase the potential gain when dealing with different loan applications. This need comes from past mistakes that lead to global financial in 2007- 2008.

In the following pages, we will provide a Machine Learning application that seeks to predict if a customer applying for a loan is the default (bad) or good. The data flow explained in this work predicts has been popular in many other works for predicting good and bad customers. However, in this case, we seek to point out the differences in accuracy of using different models to the data set. Particularly, we will show the differences in accuracy between Logistic Regression, Decision Tree and Artificial Neural Networks for making predictions. Besides, we want to know the relative importance of features contribution to the accuracy of the best model.

2 Previous related works

In the following paragraphs, we will comment on different academic works about machine learning models used for credit risk assessment. These works either used the same dataset used or similar data from other countries or institutions. Nevertheless, the most important is to show the type of selected algorithms, and how the models were built around them to obtain a good performance and accuracy.

In their work, Qiang Liu, Zhaocheng Liu, Haoli Zhang, Yuntian Chen and Jun Zhu, built a cross feature model to analyze 4 datasets from two investment banks and two independent businesses. For that, they explicitly searched for cross feature candidates that best interpreted the data. This selection made using deep neural networks, DNN, and the applied to the base classifier Logistic Regression, LR.

For this project it is important to understand the limitations of white and black-box models. White-box models refer to simple machine learning models like LR. On the contrary, black-box models refer to those complex models difficult to interpret by humans, such as neural networks, NN. Liu *et al*. (2021) assert that simple white-box models such as Logistic Regression are not powerful enough to model complex nonlinear interactions among features. On the other hand, the complex black-box models with more capacity for modeling. may lack interpretability, especially global interpretability. For that reason, the use of automatic feature crossing offers a solution to find cross features that make the classification more accurate without heavy feature engineering. However, the downside is that crossing methods might be less efficient on credit risk assessment when the data contains hundreds of feature fields.

The solution to the problem mentioned before was using the cross feature in DNN to build a more accurate set of cross features filed based on interpretation inconsistency. This help to search for the final set of cross features to train a final LR model. Namely, they used cross-feature to increase the accuracy of a white-box model like LR. This accelerates the speed from x10 to 40x. In a word, the cross-feature model improved the efficiency of white-box models for the task of credit risk assessment to make them more reliable.

A.I. Marqués, V. García, J.S. Sánchez. (2012) experimenting on the same German Credit Data dataset as well as with similar ones from Japan, Australia, Poland, and Iran. In their work, they used ensemble methods with base classifiers and compared their accuracies ones the combination took place across the different datasets. Marqués, García, and Sánchez (2012) define ensemble methods as a set of individually trained classifiers (base classifiers) whose decisions are combined in some way, typically by weighted or unweighted voting. The ensemble methods used in this work were Bagging, Boosting, Random Subspace, DECORARE, and rotation forest. With this work, the authors discovered that decision trees, DT, ANN, or support vector machines, SVM, can be fully used for credit evaluation. However, they can only be fully successful if used with ensemble methods. From these, DT performed the best, although SVM and multilayer perceptron, MLP, was not that far from.

In 2017, Mingrui Chen, Yann Dautais, LiGuo Huang, and Jidong Ge, applied machine learning methods to analyze credit data from the Bank of Bulgaria. As a result of this work, they arrived to the conclusion that "developing an effective scoring process depends on two factors 1) appropriate selection of key attributes from historical data, and 2) accurate estimation of defaulting probability produced by the predictive model. (Chen et al., 2017). For this project, the authors selected an SVM model to make predictions. For them SVM can solve both linear and nonlinear classification problems and yield promising classification performance on multidimensional data with the capability to avoid the curse of dimensionality problems. In addition, SVM is well developed for probability estimation task

In this project, all attributes were considered to avoid subjectiveness in the ML learning algorithm. 2) It eliminates the need to pick attributes and create rules manually so that they can be easily and effectively adapted to different datasets with different attributes.

3 Experiment.: German Credit Risk Data.

For this project, we have used the German Credit Data provided by Dr. Hans Hofmann from the University of Hamburg. This data is available at the UCI repository: https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data). The original dataset contains 1000 data points with 20 categorical attributes. These attributes help to classify if a person has a good or a bad credit risk according to the distribution of these attributes. Nevertheless, for the sake of simplicity, we decided to operate only 9 of the attributes to avoid falling into the trap of over-complication of the project. This data has been presented in CSV format to facilitate its readability, exploration, and processing.

The selected attributes were:

1. Age (int)
2. Sex (str: male, female)
3. Job (int: 0- unskilled and non-resident, 1 – unskilled and resided, 2 – skilled, 3 – highly skilled)
4. 4 Housing (str: own, rent, free)
5. Saving accounts (str: little, moderate, quite rich, rich)
6. Checking account (int)
7. Credit amount (int)
8. Duration (int, months)
9. Purpose (str: car, furniture/equipment, radio /TV, domestic appliances, repairs, education, business, vacation/others)

3.1 Importing the necessary libraries and loading the dataset

First, we imported the necessary libraries for exploring and manipulating the dataset.

```python
#Loading the libraries

import pandas as pd # To work with the dataset
import numpy as np #Math library
import seaborn as sns #Graph library that use matplot in background
import matplotlib.pyplot as plt #to plot some parameters in seaborn
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler  #To scale the data
import os # accessing directory structure


#Importing the data
df_credit = pd.read_csv("german_credit_target.csv",index_col=0)
```

First, look at the data

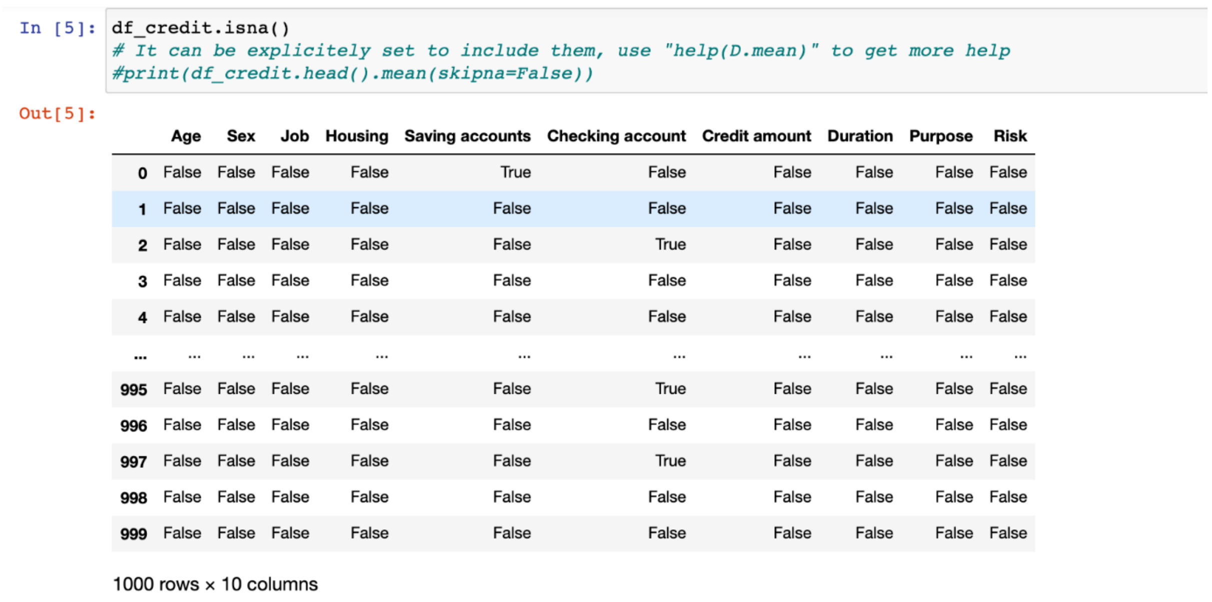
```python
# Have a look to the data

print(df_credit.info())
print()

#Looking unique values
print(df_credit.nunique())
print()

#Looking the data
print(df_credit.head())
print()

df_credit.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Age               1000 non-null   int64 
 1   Sex               1000 non-null   object
 2   Job               1000 non-null   int64 
 3   Housing           1000 non-null   object
 4   Saving accounts   817 non-null    object
 5   Checking account  606 non-null    object
 6   Credit amount     1000 non-null   int64 
 7   Duration          1000 non-null   int64 
 8   Purpose           1000 non-null   object
 9   Risk              1000 non-null   object
dtypes: int64(4), object(6)
memory usage: 85.9+ KB
None

Age                  53
Sex                   2
Job                   4
Housing               3
Saving accounts       4
Checking account      3
Credit amount       921
Duration             33
Purpose               8
Risk                  2
dtype: int64
```

```
   Age     Sex  Job Housing Saving accounts Checking account  Credit amount  \
0   67    male    2     own             NaN           little           1169   
1   22  female    2     own          little         moderate           5951   
2   49    male    1     own          little              NaN           2096   
3   45    male    2    free          little           little           7882   
4   53    male    2    free          little           little           4870   

   Duration              Purpose  Risk  
0         6             radio/TV  good  
1        48             radio/TV   bad  
2        12            education  good  
3        42  furniture/equipment  good  
4        24                  car   bad  
```

	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose	Risk
0	67	male	2	own	NaN	little	1169	6	radio/TV	good
1	22	female	2	own	little	moderate	5951	48	radio/TV	bad
2	49	male	1	own	little	NaN	2096	12	education	good
3	45	male	2	free	little	little	7882	42	furniture/equipment	good
4	53	male	2	free	little	little	4870	24	car	bad

We checked the NaN values to know how to clean the dataset.



```
In [5]: df_credit.isna()
        # It can be explicitely set to include them, use "help(D.mean)" to get more help
        #print(df_credit.head().mean(skipna=False))
```

Out[5]:

	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose	Risk
0	False	False	False	False	True	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	True	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
995	False	False	False	False	False	True	False	False	False	False
996	False	False	False	False	False	False	False	False	False	False
997	False	False	False	False	False	True	False	False	False	False
998	False	False	False	False	False	False	False	False	False	False
999	False	False	False	False	False	False	False	False	False	False

1000 rows × 10 columns

3.2 Pre-processing and Feature extraction

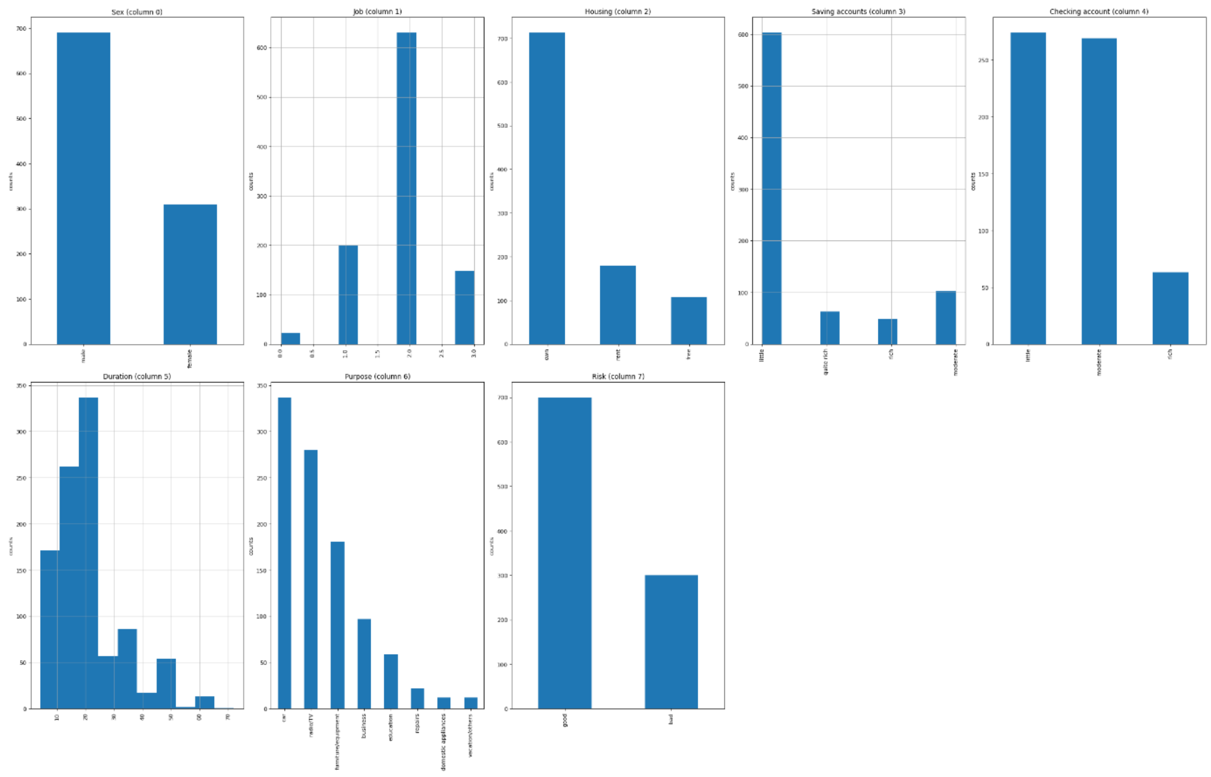
Once the data has been explored, we plot the different features to see each individual distribution. Our feature selection criteria were based on a normal distribution. This is that each feature was seen as a random variable that had the lumps on one side of the X-y axis. These lumps were a special significance to check the worthiness of the feature.

The bar charts represent the frequency of each category in the X and Y-axis. Those features that had too little bars or very low numbers were discarded because they may not be very helpful for the ML models. As a result, we saw in Age, Credit amount, Job, and Duration as interesting dimension to further our analysis. Their skewed distribution made them acceptable to continue our experiment.

For the sake of simplicity and conciseness of the code, we decided to define three functions in python where we clean the data, define the task of the function and define the plot parameters for showing the feature distribution, correlation matrix and scatter and density plots. These functions where defined as: *plotPerColumnDistribution(df,* *nGraphShown, nGraphPerShow)* for the feature distribution; *plotCorrelationMatrix(df,graphWidth)* for the correlation matrix; and *plotScatterMatrix(df,PlotSize,TextSize*) for the scatter and density plots.

```
# Distribution graphs (histogram/bar graph) of column data
def plotPerColumnDistribution(df, nGraphShown, nGraphPerRow):
    nunique = df.nunique()
    df = df[[col for col in df if nunique[col] > 1 and nunique[col] < 50]] # For displaying purposes, pick columns that
    nRow, nCol = df.shape
    columnNames = list(df)
    nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow
    plt.figure(num = None, figsize = (6 * nGraphPerRow, 8 * nGraphRow), dpi = 80, facecolor = 'w', edgecolor = 'k')
    for i in range(min(nCol, nGraphShown)):
        plt.subplot(nGraphRow, nGraphPerRow, i + 1)
        columnDf = df.iloc[:, i]
        if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):
            valueCounts = columnDf.value_counts()
            valueCounts.plot.bar()
        else:
            columnDf.hist()
        plt.ylabel('counts')
        plt.xticks(rotation = 90)
        plt.title(f'{columnNames[i]} (column {i})')
    plt.tight_layout(pad = 1.0, w_pad = 1.0, h_pad = 1.0)
    plt.show()
```

```
plotPerColumnDistribution(df_credit, 10, 5)
```



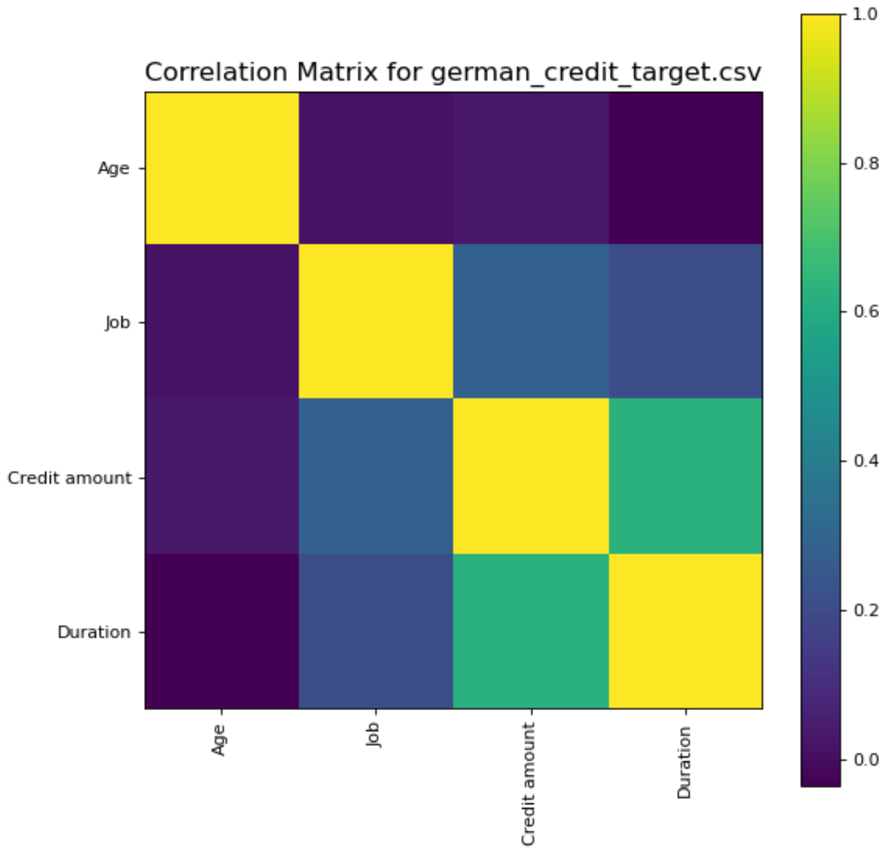
Correlation matrix.

After selecting those features with the best distributions. We plotted a correlation matrix. And checked their density in scatter and density plots. A correlation matrix is a table that shows the correlation coefficients for different variables. This matrix depicts the correlation between the possible pairs of values in a table. This is a very effective way to summarize a large dataset and visualize patterns in the given data.

We saw that variables Credit Amount and Duration have a positive correlation with Job, while age has a negative impact. Higher Credit amount and Duration suggest that the value of Job will be higher (close to 1.0). This increases the chance of high credit risk (or possibility for default).

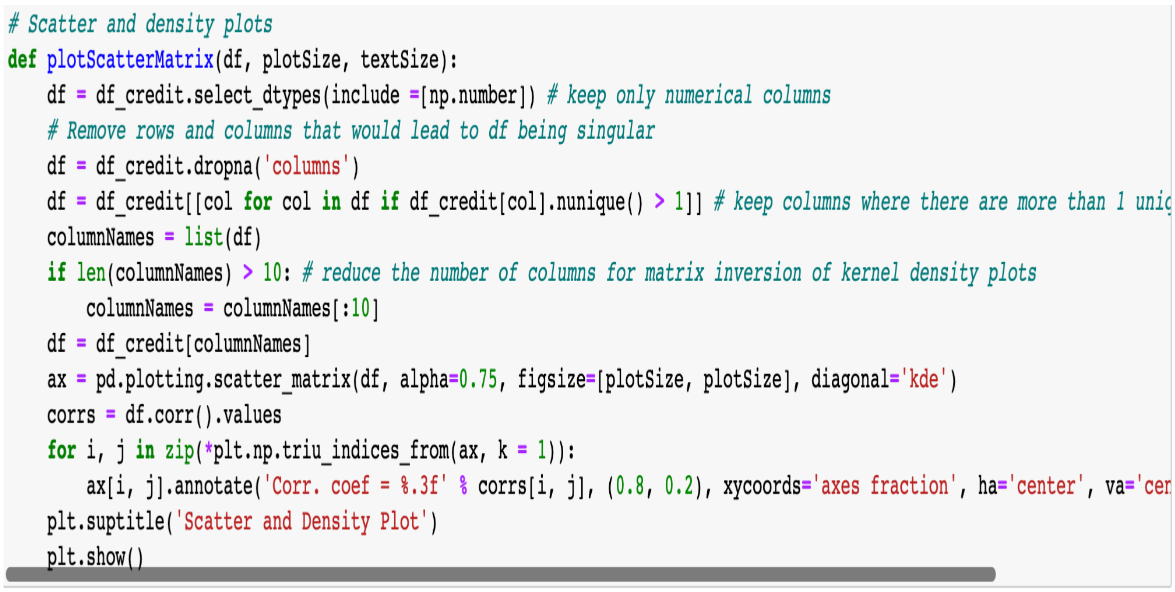
```
# Correlation matrix
def plotCorrelationMatrix(df, graphWidth):
    df_credit.dataframeName = 'german_credit_target.csv'
    filename = df_credit.dataframeName
    df = df_credit.dropna('columns') # drop columns with NaN
    df = df_credit[[col for col in df if df_credit[col].nunique() > 1]] # keep columns where there are more than 1 uniq
    if df_credit.shape[1] < 2:
        print(f'No correlation plots shown: The number of non-NaN or constant columns ({df.shape[1]}) is less than 2')
        return
    corr = df_credit.corr()
    plt.figure(num=None, figsize=(graphWidth, graphWidth), dpi=80, facecolor='w', edgecolor='k')
    corrMat = plt.matshow(corr, fignum = 1)
    plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)
    plt.yticks(range(len(corr.columns)), corr.columns)
    plt.gca().xaxis.tick_bottom()
    plt.colorbar(corrMat)
    plt.title(f'Correlation Matrix for {filename}', fontsize=15)
    plt.show()
```

```
plotCorrelationMatrix(df_credit, 8)
```

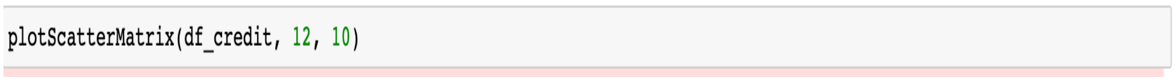


Scatter Plot

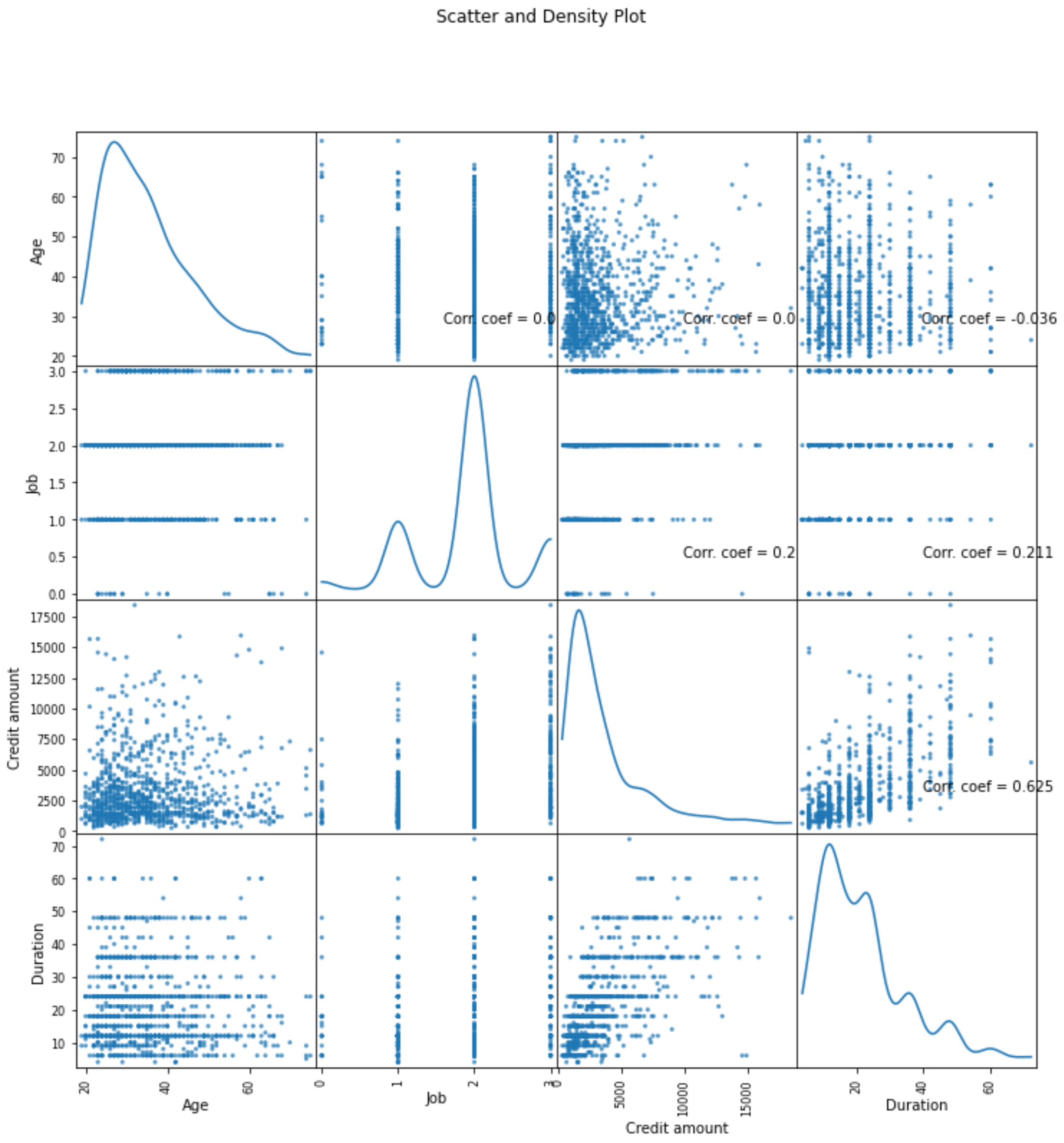
Below, we showcase the scatter plot of the features selected in the correlation matrix. This plot shows the density of the value pairs proposed in the previous matrix. We can see that the data spread is wider in the Y-axis than on the X-axis.



```
# Scatter and density plots
def plotScatterMatrix(df, plotSize, textSize):
    df = df_credit.select_dtypes(include =[np.number]) # keep only numerical columns
    # Remove rows and columns that would lead to df being singular
    df = df_credit.dropna('columns')
    df = df_credit[[col for col in df if df_credit[col].nunique() > 1]] # keep columns where there are more than 1 uniq
    columnNames = list(df)
    if len(columnNames) > 10: # reduce the number of columns for matrix inversion of kernel density plots
        columnNames = columnNames[:10]
    df = df_credit[columnNames]
    ax = pd.plotting.scatter_matrix(df, alpha=0.75, figsize=[plotSize, plotSize], diagonal='kde')
    corrs = df.corr().values
    for i, j in zip(*plt.np.triu_indices_from(ax, k = 1)):
        ax[i, j].annotate('Corr. coef = %.3f' % corrs[i, j], (0.8, 0.2), xycoords='axes fraction', ha='center', va='cen
    plt.suptitle('Scatter and Density Plot')
    plt.show()
```



```
plotScatterMatrix(df_credit, 12, 10)
```


Scatter and Density Plot
Age
Job
Credit amount
Duration
Corr. coef = 0.0
Corr. coef = 0.0
Corr. coef = -0.036
Corr. coef = 0.2
Corr. coef = 0.211
Corr. coef = 0.625

4 Model Building

4.1 Logistic regression

Linear models make predictions using a linear function of the input features, and they can be also used for classification following a similar formula to the one used for normal linear regression:

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \ldots + w[p] * x[p] + b > 0$$

A Binary Linear classifier like LR is a classifier that separates two classes using a line, a plane, or a hyperplane. However, the difference is that linear models for classification mark a threshold to the predicted value to zero instead of returning the weighted sum of the features (Müller and Guido, 2017). If the result is smaller than zero, the prediction is class -1. If it is greater than 0, the prediction belongs to class +1. Namely, in linear models for classification, the decision boundary is a linear function of the input.

It is important to keep in mind that LR is a classification algorithm and not a regression algorithm. Therefore, it should not be confused with Linear Regression. Also, LR applies a L2 regularization by default for interpretability reasons and are very fast to train and to predict. Another strength is that they make it easy to understand how a prediction is made. LR often performs well when the number of features is large compared to the number of datapoints. it is often used on very large datasets although they may not yield a good generalization performance when working in low-dimensional spaces.

For the logistic regression we divided the data in train set (75%) and test set (25%).

```
# Building the model for Logistic regression
from sklearn.model_selection import train_test_split, KFold, cross_val_score # to split the data
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, fbeta_score
from sklearn.linear_model import LogisticRegression

df = df_credit.select_dtypes(include =[np.number]) # keep only numerical columns
X = df
y = df_credit["Risk"]
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.25,random_state=42)
logre = LogisticRegression().fit(X,y)
y_pred = logre.predict(X_train)

logre.score(X_test, y_test)
print()
#confusion_matrix(y_test, y_pred)

RsquaredCV=cross_val_score(logre, X, y, cv=5).mean()
RsquaredTR=logre.score(X,y)
print("Logistic Regression score:", logre.score(X_test, y_test))
print("CV score..........", RsquaredCV)
print("Training score....", RsquaredTR)
```

```
Logistic Regression score: 0.74
CV score.......... 0.709
Training score.... 0.712
```

4.2 Cross validation

To evaluate the effectiveness of our LR and Decision Tree models, we used a cross-validation. This is a statistical method that assess and compare the learning algorithms by dividing the data into two segments: one for learning a model, and other for validate the model. This technique involves reserving a particular sample of a dataset where the model is not trained. Then you test the model before finalizing it. Cross-validation is used to prevent overfitting in a predictive model. This technique is commonly used in case where the data is limited. With cross-validation, a fixed number of folds are created out of the data, each of them is analyzed, and then the overall average error is estimated.

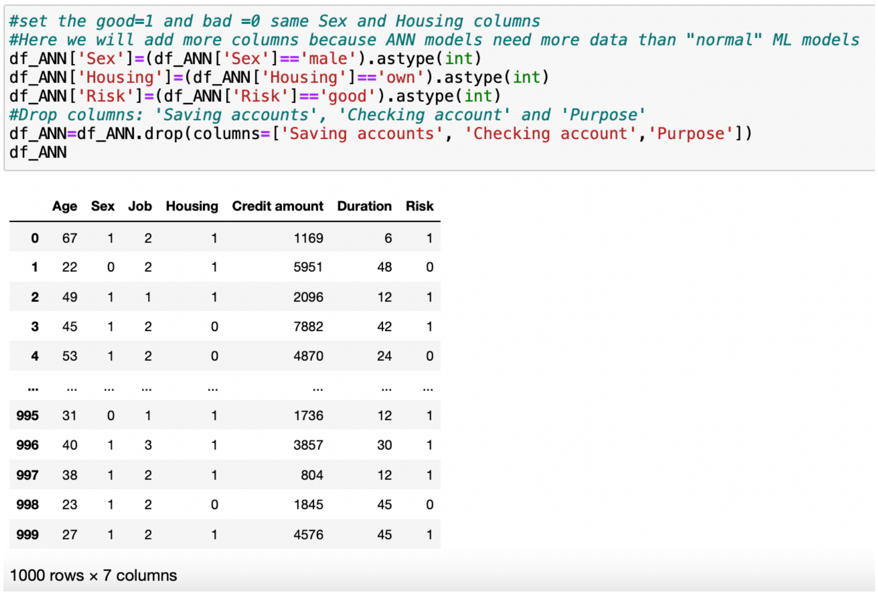
With cross validation we obtained a more accurate performance of the data. This helped us to have a more efficient use of the data. In the case of LR, the Cross-validation score, CV score, was 0.76, which reflects an acceptable predictive capability of the model. The CV score for the Random Forest was 0.67, which was not far away from our LR model. Despite of these results, the predictability capacity of these models can be possible if some other methods are included when preprocessing the data, and if we include more dimensions and a more elaborated feature engineering

5 Artificial Neural Network

We assumed that Neural network models do not fit in purposes of this data before implementing MLPRegressor and ExtraTreesRegressor models to the credit risk. The assumption was correct. The data was too simple and small to get the relations between parameters to predict the credit risk.

5.1 Preprocessing the data

The data is the same as other models however we tried to make it a little bit more complex by adding Sex and Housing columns. Male data is changed to 1 and female to 0. Same processing was made to housing; own was set to 1 and free to 0. Risk profile was set to good = 1 and bad = 0. All the columns in the data are: Age, Sex, Job, Housing, Credit amount, Duration & Risk. We did not remove the NaN values because there were many of them.

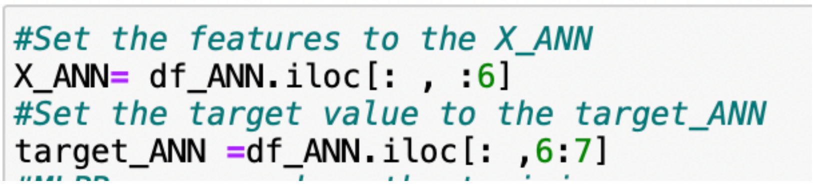


```
#set the good=1 and bad =0 same Sex and Housing columns
#Here we will add more columns because ANN models need more data than "normal" ML models
df_ANN['Sex']=(df_ANN['Sex']=='male').astype(int)
df_ANN['Housing']=(df_ANN['Housing']=='own').astype(int)
df_ANN['Risk']=(df_ANN['Risk']=='good').astype(int)
#Drop columns: 'Saving accounts', 'Checking account' and 'Purpose'
df_ANN=df_ANN.drop(columns=['Saving accounts', 'Checking account','Purpose'])
df_ANN
```

	Age	Sex	Job	Housing	Credit amount	Duration	Risk
0	67	1	2	1	1169	6	1
1	22	0	2	1	5951	48	0
2	49	1	1	1	2096	12	1
3	45	1	2	0	7882	42	1
4	53	1	2	0	4870	24	0
...	...	...	...	...	...	...	...
995	31	0	1	1	1736	12	1
996	40	1	3	1	3857	30	1
997	38	1	2	1	804	12	1
998	23	1	2	0	1845	45	0
999	27	1	2	1	4576	45	1

1000 rows × 7 columns

Now we can set the features and target values to different arrays.

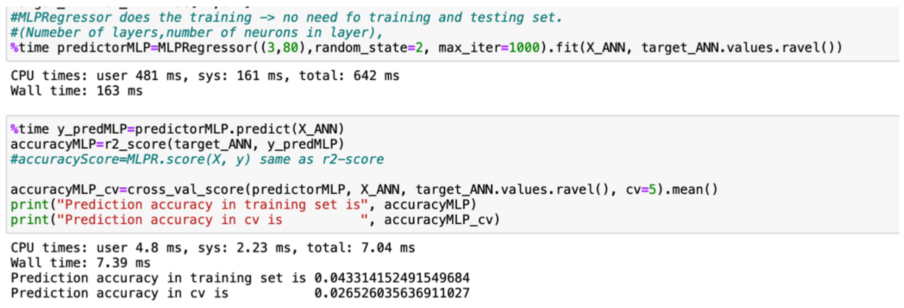


```
#Set the features to the X_ANN
X_ANN= df_ANN.iloc[: , :6]
#Set the target value to the target_ANN
target_ANN =df_ANN.iloc[: ,6:7]
```

5.2 MLP Regressor

The first ANN model is based on a Multi-layer Perceptron regressor.

Accuracy shows that the model is not a good fit for our purposes because it is very small. The accuracy should be larger than 60%, so we could even say it can get the importance of the different parameters.



```
#MLPRegressor does the training -> no need fo training and testing set.
#(Numeber of layers,number of neurons in layer),
%time predictorMLP=MLPRegressor((3,80),random_state=2, max_iter=1000).fit(X_ANN, target_ANN.values.ravel())
```

```
CPU times: user 481 ms, sys: 161 ms, total: 642 ms
Wall time: 163 ms
```

```
%time y_predMLP=predictorMLP.predict(X_ANN)
accuracyMLP=r2_score(target_ANN, y_predMLP)
#accuracyScore=MLPR.score(X, y) same as r2-score

accuracyMLP_cv=cross_val_score(predictorMLP, X_ANN, target_ANN.values.ravel(), cv=5).mean()
print("Prediction accuracy in training set is", accuracyMLP)
print("Prediction accuracy in cv is          ", accuracyMLP_cv)
```

```
CPU times: user 4.8 ms, sys: 2.23 ms, total: 7.04 ms
Wall time: 7.39 ms
Prediction accuracy in training set is 0.043314152491549684
Prediction accuracy in cv is           0.026526035636911027
```

5.3 Extra Trees Regressor

The extra Trees Regressor algorithm is related to the random forest regression algorithm (Brownlee, 2021). They usually get similar results however Extra Trees is a little more straightforward. MLPRegressor provided better results than Extra Trees, yet both models are not good because accuracy was under 5 % in both models.

```
#Let's try ExtraTreesRegressor
from sklearn.ensemble import ExtraTreesRegressor
```

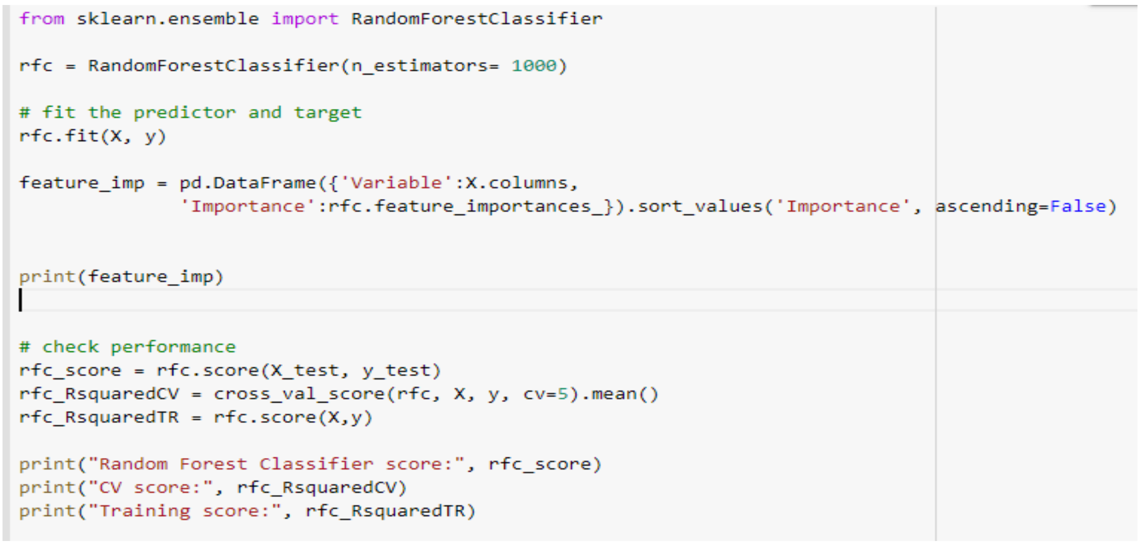
```
e same target group.
edictorETR=ExtraTreesRegressor(n_estimators=1000,max_depth=1, random_state=1).fit(X_ANN, target_ANN.values.ravel())
hatETR=predictorETR.predict(X_ANN)
uracy=r2_score(target_ANN, y_hatETR)
core=cross_val_score(predictorETR, X_ANN, target_ANN.values.ravel(), cv=5).mean()
rediction accuracy in training set is", ExtraAccuracy)
rediction accuracy in cv is          ", ExtraCVScore)
```

```
CPU times: user 374 ms, sys: 2.43 ms, total: 377 ms
Wall time: 379 ms
CPU times: user 42.5 ms, sys: 653 µs, total: 43.2 ms
Wall time: 42.6 ms
Prediction accuracy in training set is 0.037765073761232726
Prediction accuracy in cv is           0.015493488976662561
```

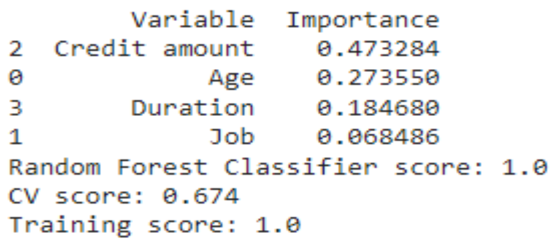
MLPRegressor model performed better than ExtraTreessRegressor yet the model is not good for this purposes. The data set is too small and simple.

6 Random Forest

Generally, the random forest classifier or classification in random forest employs an ensemble method to attain the outcome. The training data is fed to train various decision trees; thus, we can say the forest relies on various decision trees. Subsequently, every decision tree consists of decision nodes, leaf nodes, and a root node. And the leaf node of each tree is the final output produced by specific decision tree. The selection of the final output follows the majority-voting system. In this case, the output chosen by most of the decision trees becomes the final output of the rain forest system.
Similarly, our model establishes the outcome based on the predictions of the decision trees within the forests and it predicts by taking the average or mean of the output from these various trees. Our random forest classifier model produced an accuracy score 100% and with the cross-validation score of 67.4%. When you check the feature importance of the features used in generating the model, you could see that credit amount feature contribute 47.3% (highest) in explaining the quality score of the model, followed by age feature 27.4%, duration feature 18.5% and with job feature with the least contribution of 6.8% in explaining the quality model.



```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators= 1000)

# fit the predictor and target
rfc.fit(X, y)

feature_imp = pd.DataFrame({'Variable':X.columns,
              'Importance':rfc.feature_importances_}).sort_values('Importance', ascending=False)


print(feature_imp)


# check performance
rfc_score = rfc.score(X_test, y_test)
rfc_RsquaredCV = cross_val_score(rfc, X, y, cv=5).mean()
rfc_RsquaredTR = rfc.score(X,y)

print("Random Forest Classifier score:", rfc_score)
print("CV score:", rfc_RsquaredCV)
print("Training score:", rfc_RsquaredTR)
```



```
         Variable  Importance
2   Credit amount    0.473284
0             Age    0.273550
3        Duration    0.184680
1             Job    0.068486
Random Forest Classifier score: 1.0
CV score: 0.674
Training score: 1.0
```

7 Conclusion

In the previous pages we used ML algorithms to analyze credit risk data for making predictions if a client is a good or bad lender. However, the aim of this project was to compare the prediction accuracy between Logistic Regression, Artificial Neural Networks, and Random Forest.

Logistic Regression was an acceptable model for making predictions. Its good score in the Cross-validation, 0.70, makes this model as good as Random Forest. However, it is important to keep in mind that this model is not completely efficient when working with low-dimensional data. Namely, to increase the prediction capabilities of the model it would be necessary to introduce more dimensions to the data. In addition, ensemble methods would be necessary to increase the effectiveness of the model.

Artificial Neural Network models in this project based on Extra Trees Regressor and Multi-layer Perceptron regressor did not fit in our purposes. We could have compared Extra Trees Classifier and Multi-layer Perceptron Classifier as well. However, we decided to leave them out. MLP Regressor model's accuracy was 4,3% and Extra Trees Regressor model's 3,7% which were not food with low cross-validation scores in both cases. Other models, we built succeeded better to capture the relations between features and risk.

Comparing the quality score of all the models generated, the random forest has the best quality score, therefore we conclude that it is the best model for establishing the risk profile or creditworthiness of individuals based on the dataset that we used for this project. Further, we observed that the features considered in developing the models, the most important feature that the banks should consider essential in determining the risk profile of their loan clientele is the credit amount variable (47.3% importance) and age variable (27.4% importance).

References

Alpaydin, Ethem (2014), Introduction to Machine Learning. 3rd edition. MIT press; Cambridge, Massachusetts, London, England. ISBN 978-0-262-02818-9 (hardcover : alk. paper)

A.I. Marqués, V. García, J.S. Sánchez. (2012). Exploring the behaviour of base classifiers in credit score ensembles. University Jaume I. Spain. Elsevier Ltd. Expert Systems with Applications 39 (2012) 10244–10250

Brownlee, J. (2021). How to Develop an Extra Trees Ensemble with Python. Machine Learning Mastery. Retrieved 2021-12-20 from https://machinelearningmastery.com/extra-trees-ensemble-with-python/

Data: Dr. Hans Hofmann, (NA). *German Credit Data*. UCI repository: https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

Mingrui Chen, Yann Dautais, LiGuo Huang, and Jidong Ge. 2017. Data Driven Credit Risk Management Process: A Machine Learning Approach. In Proceedings of 2017 International Conference on Software and Systems Process, Paris, France, July 2017 (ICSSP'17), 5 pages. DOI: 10.1145/3084100.3084113

Müller A. And Guido S., 2017. Introduction to Machine Learning with Python. Third Edition. O'Reilly, CA, USA. ISBN: 978-1-449-36941-5

Qiang Liu, Zhaocheng Liu, Haoli Zhang, Yuntian Chen and Jun Zhu. 2021. Mining Cross Features for Financial Credit Risk Assessment. In Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482371