Loan Status Prediction

- AI Homework 2 Report -

# Contents

# 1. Introduction

# 2. Exploratory Data Analysis

# 3. Data Preprocessing

# 4. Machine Learning Models

# 5. Conclusions and Improvements

# 1. Introduction

For the scope of this document, we are working with this [Loan Approval Classification Dataset](https://www.kaggle.com/datasets/taweilo/loan-approval-classification-data?resource=download). It is a synthetic dataset, inspired by the original [Credit Risk Dataset](https://www.kaggle.com/datasets/laotse/credit-risk-dataset) and enriched with additional variables based on the [Financial Risk for Loan Approval](https://www.kaggle.com/datasets/lorenzozoppelletto/financial-risk-for-loan-approval) dataset.

Our goal is to create multiple types of classifiers & getting in the end a model that is as accurately as possible in classifying whether or not a person will get their loan approved.

# 2. Exploratory Data Analysis

Our dataset is contained in a CSV (Comma Separated Values) file. It is containing 45.000 entries, each with 14 variables, as follows:

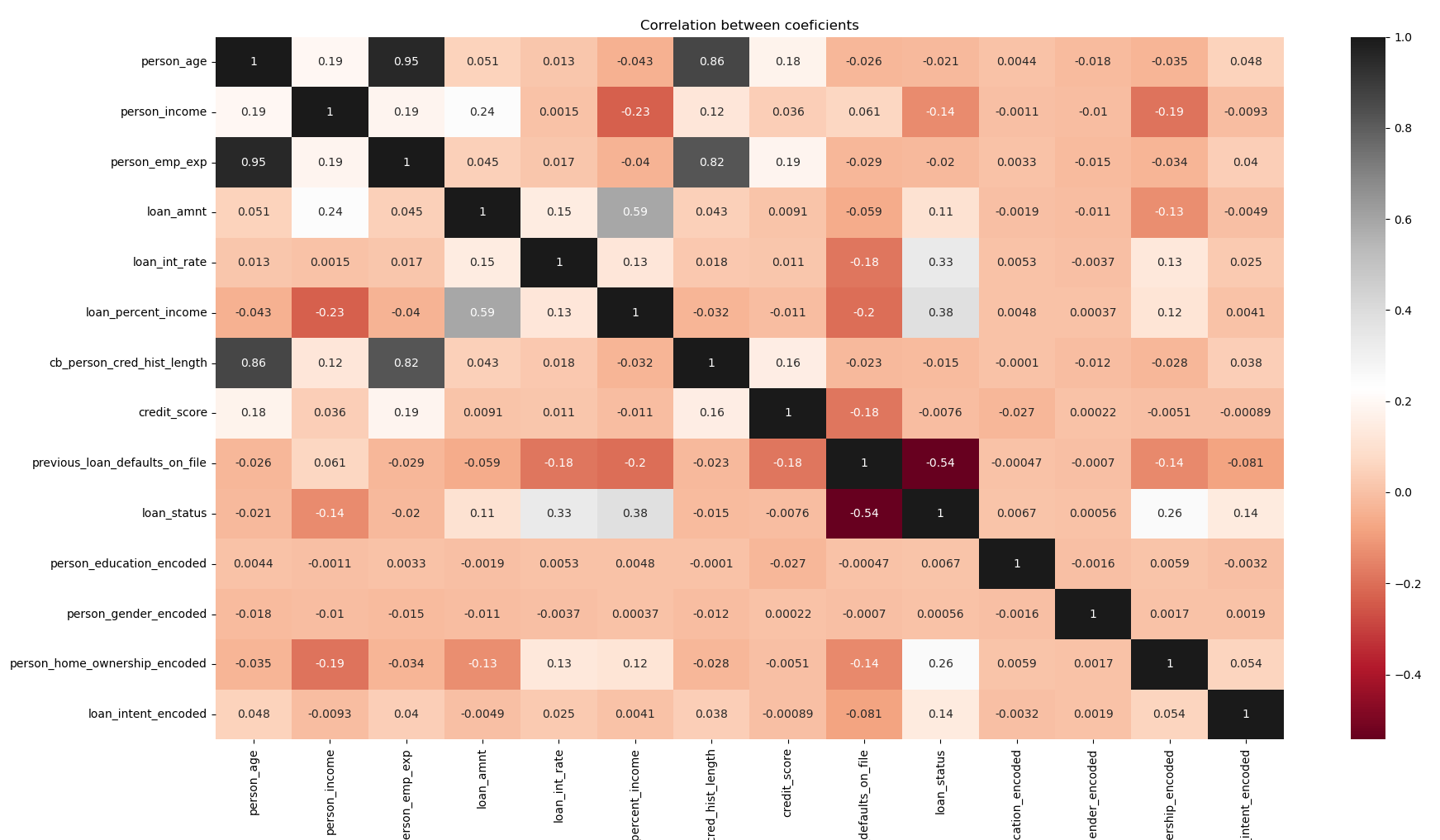
|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Variable Description** | **Variable Type** |
| person\_age | Age of the person | Flot |
| person\_gender | Gender of the person | Categorical |
| person\_education | Highest education level | Categorical |
| person\_income | Annual income | Float |
| person\_emp\_exp | Years of employment experience | Integer |
| person\_home\_ownership | Home ownership status | Categorical |
| loan\_amnt | Loan amount requested | Float |
| loan\_intent | Purpose of the loan | Categorical |
| loan\_int\_rate | Loan interest rate | Float |
| loan\_percent\_income | Loan amount as percentage of annual income | Float |
| cb\_person\_cred\_hist\_length | Length of credit history in years | Float |
| credit\_score | Credit Score of the person | Integer |
| previous\_loan\_defaults\_on\_file | Indicator of previous loan defaults | Categorical |
| loan\_status | Loan approval staus: 1 = approved; 0 = rejected | Integer |

Before building our machine learning models, we will analyze the data that we are getting.

Firstly, by observing the correlation matrix in figure 1, we can better see the correlation between our data points. If between two data points we have a positive score, there is a directly proportional correlation, while a negative score indicates an inverse correlation. As such, a score of 0 means that there is no correlation between the two data points.

From this matrix, we can clearly see that there direct correlations between:

* higher experience and credit length directly proportional with the age of a person
* loan amount directly proportional with the income of a person
* previous loan inversely proportional with the status of the loan

Figure 1: Corellation Matrix

Since the age of a person is corelating with multiple data points, I’ve decided do create a histogram of the *person\_age* variable, in order to get a better perspective.

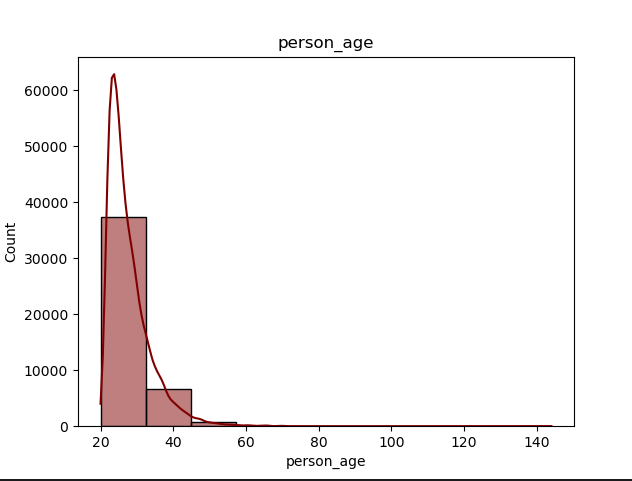
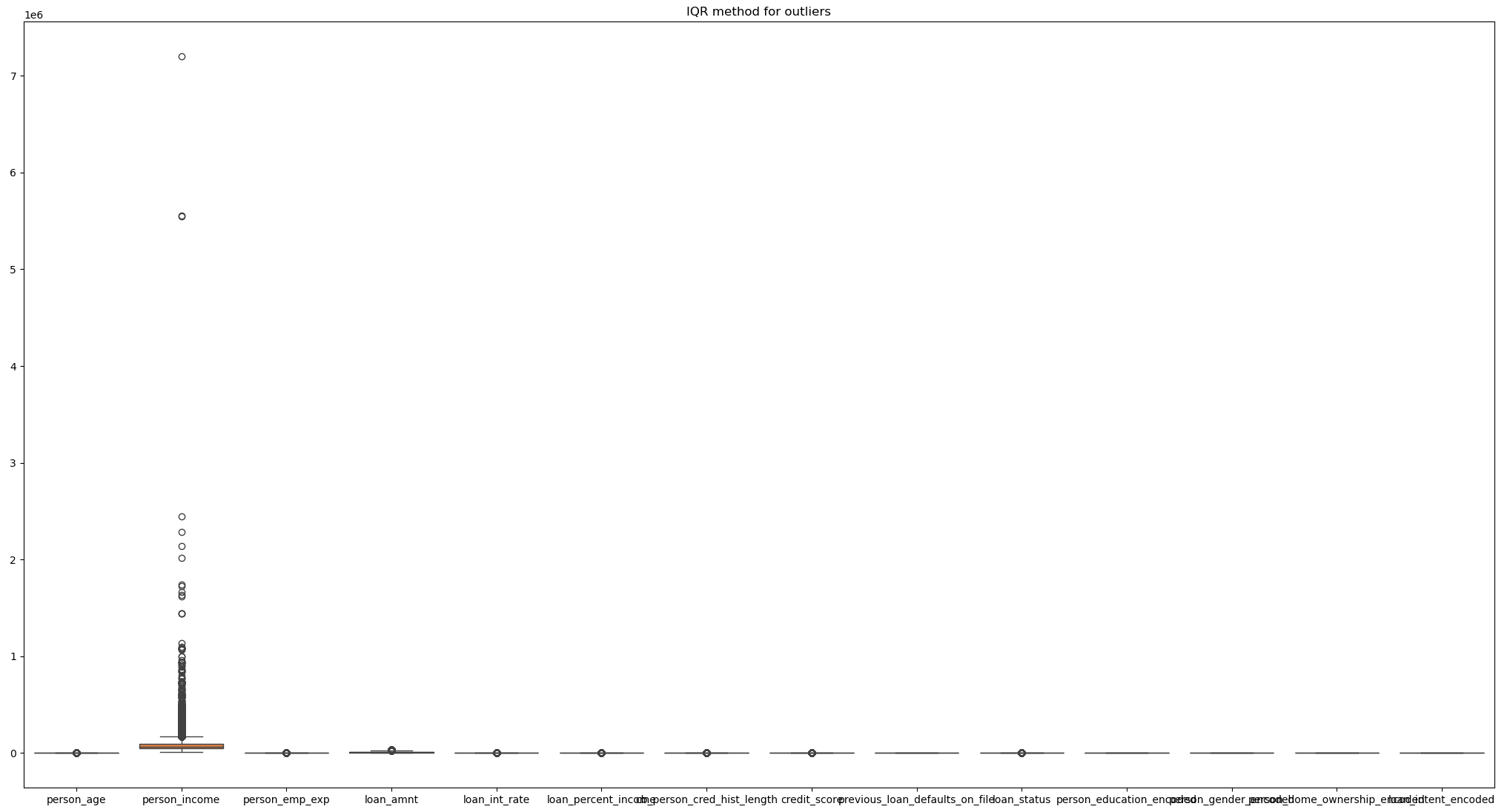


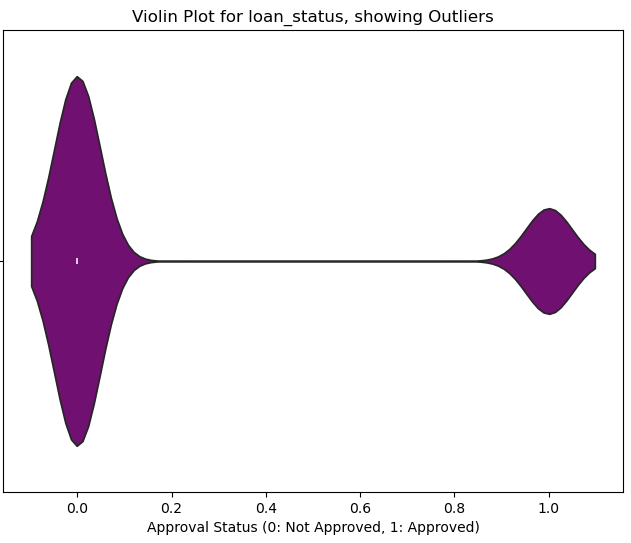
Figure 2: Age histogram

Based on this histogram we can see that the majority of persons that are seeking out a loan are between the age of 20 to 30 years old, around 38.000 out of our sample size of 45.000. The rest of 8.000 applicants are around the age of 40 years old, with a very small sample of people in their 50s. This imbalance introduces a sampling bias, favoring predictions for younger applicants while potentially reducing accuracy when an older applicant appears.

Besides these two, I’ve also done an outlier analysis, by using a box plot, which can be found at figure 2 and the Interquartile Range method (or IQR for short). By using IQR, we are focusing on the middle 50% of the data, making sure it is variable. The IQR is calculated as: I***QR = Q3 (upper quartile, 75th%) – Q1 (lower quartile, 25th%)***. As such, the outliers are identified as points that are outside of the range of ***[Q1 – 1.5 \* IQR, Q3 + 1.5 \* IQR]***. This method helps to flag extreme values that may influence our analysis or indicate important anomalies.

Figure 3: Outliers graph via IQR

As we can observe, the second variable, *person\_income*, is visibly presenting a lot of outliers, while the loan\_status has the most outliers, at around 10.000. The rest of the variables appear to not have any outliers.

In order to better visualise the outliers of the *loan\_status* variable, I used the plot on the right. It shows that the value of 0 is the “normal” status of the loan, meaning that most people do not get their loans approved. In consequence, this makes every approved loan an outlier in our case.

This can create a class imbalance and lead to balanced predictions leaning toward denied loans, which can be solved by undersampling denied cases, alongside using SMOTE.

# 3. Data Preprocessing

In order for us to analyze the data and train our models, we first have to encode the non-numerical data, i.e. the Categorical data points.

For this task, we can do it in two ways,

1. One Hot Encoding method: this method is based on representing categorical data as binary vectors, a format that can be interpreted by the algorithms of our models.
2. Label Encoding method: for each categorical data type, we assign an unique integer to each category.

Each methods has its own advantages and disadvantages, but I decided to go with the Label Encoding method, as it is simpler and more compact than the other option. Since our dataset has a lot of non-numerical data, another column is added to the data frame when we use the One Hot method, which leads to a huge loss in the performance of the model.

Another method that could be used on the dataset is PCA, or Principal Component Analysis. It is a statistical technique commonly used in order to reduce the dimensions of our data while retainging as much information as possible. PCA would transform our dataset into a set of uncorrelated variables called *principal components*. However, I did not use PCA on this particular dataset due to the decline in model accuracy, and the fact that our training porocess is already fast.

If we were to address the problem of outliers, as stated before, SMOTE would be the best performing method. Synthetic Minority Over-sampling Technique is an advanced technique for dealing with class imbalace. When you have a minority class vs. a majority class (such as our case in regards to the *loan\_status* variable, ML algorithms tend to get biased towards predicting the majority class. SMOTE would help us in this regard by generating synthetic samples for the minority class instead of undersampling the existing minority class. This leads to both a more balanced dataset and an improved performance.

Lastly, another problem which would normally be tackled in the preprocessing step would be missing values that would appear in each row. Since our dataset has no holes present, we can skip over this step, but if we were to tackle this challenge, interpolation would be the best method of going ahead.

# 4. Machine Learning Models

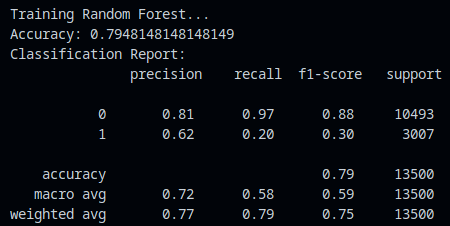
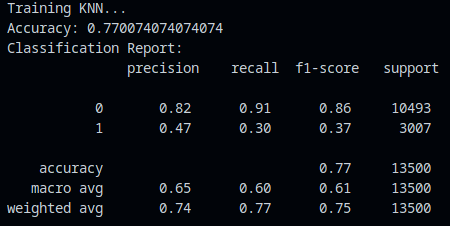
The models that were chosen for this experiment were chosen due to my familiarity with them. In no particular order, we have:

* **Random Forest** is an ensemble learning technique that builds multiple decision trees and combines their predictions to enhance accuracy and reduce the risk of overfitting, making it robust for various datasets.
* **K-Nearest Neighbors (KNN)** is a straightforward, instance-based algorithm used for classification and regression. It predicts outcomes by examining the closest neighboring data points based on a specified distance metric.
* **Decision Tree** is a versatile model that uses a tree-like structure to split data into subsets based on feature-based conditions at each node, making it suitable for both classification and regression.
* **XGBoost** is a cutting-edge algorithm based on the Gradient Boosting framework, optimized for high performance in predictive modeling tasks with superior speed and accuracy.
* **Logistic Regression** is a classification algorithm tailored for binary outcomes (e.g., 0 or 1). Despite its name, it focuses on classification rather than regression tasks.

For all of these models, I used a 70% / 30% split between the training data and the testing one, except for the Random Forest method, which had it’s best estimator as a parameter.

# 5. Training results

Now that we have went through both data preprocessing & choosing our models, it’s time to train them.  
 Here are the results:



# 

# 

# 

# 

As it can be observed, the accuracy of our models is orbiting around the values of 77%-79%, with the Decision Tree model underperforming when compared to the rest, at an accuracy of 71%.

As for our best model, we have the XGBoost and the Random Forest models tied for the best performance, both seeing an accuracy of around 79.4%.

# 6. Conclusions and Improvements

In order to train our models efficiently, we had to preprocess our dataset by using SMOTE and the label encoding method, although the raw dataset itself is a quality one.

For future improvements, we would have to choose between our two top performers, XGBoost and Random Forest, and make specific adjusments from there on, depending of our choice.