MACHINE LEARNING PIPELINE FOR CREDIT RISK ANALYSIS WITH THE GERMAN CREDIT DATA

CS412 INTRODUCTION TO MACHINE LEARNING

SPRING 2022-2023

REPORT BY:

ZEYNEP KURTULUS

SEZIN TEKIN

EGE ONGUL

OZAN CELEBİ

1. Introduction

Task and Importance

The goal of the CS412-Machine Learning project is to gain practical experience in an ML project pipeline by applying classification algorithms to the real-world problem of credit risk assessment. This project holds significant importance as it allows for the practical application of machine learning techniques and emphasizes the role of feature selection in building effective models for credit risk assessment.

Dataset Description

For this project, we will be utilizing the widely used German Credit (Statlog) Dataset, a benchmark dataset for credit risk assessment. The dataset consists of 1,000 instances, each characterized by 9 features which are: age, sex, job, housing, savings account, checking account, credit amount, duration and purpose. The dataset includes 700 instances labeled as "good credit" and 300 instances labeled as "bad credit". The objective is to predict whether a loan applicant is likely to have a good or bad candidate in terms of receiving the credit based on the available features. Below you may see the beginning of the dataset:

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

Dataset Preprocessing

The German Credit Dataset will be provided in a comma-separated value which is in CSV format. Before building any models, it is essential to preprocess the dataset appropriately. This involves handling missing values and encoding categorical features. Some features in the dataset are categorical, such as "Housing" and "Checking account," which may require additional operations before processing them based on the feature's nature and the algorithm used.

Additionally, the dataset contains NaN values that need to be addressed using data imputation techniques such as mean imputation or KNN imputation.

Methodology

To tackle the credit risk assessment problem, we will apply various classification algorithms to the dataset. From the provided list, we have chosen at least three suitable algorithms:

- K-Nearest Neighbors (KNN)
- Decision Trees (DT)
- Logistic Regression (LR)
- Multilayer Perceptrons (MLP)
- Support Vector Machines (SVM)

The algorithms that we choose are; KNN, LR and DT. By implementing these algorithms, we aim to evaluate their performance and determine the most important features for predicting credit risk. This will provide valuable insights into the strengths and limitations of each algorithm, contributing to a better understanding of their applicability in real-world scenarios.

By following this methodology, we will gain hands-on experience in the ML project pipeline, develop practical skills in selecting and evaluating classification algorithms, and deepen our understanding of the role of feature selection in building effective models for credit risk assessment.

2. Dataset

The German Credit (Statlog) Dataset provided in the code consists of 1,000 instances or samples, each represented by 9 features. The dataset is used for credit risk assessment. It contains both categorical and numerical features related to loan applicants.

The class label, which represents the target variable, is stored in the last column of the dataset. The goal is to predict whether a given loan applicant is likely to have "good credit" or "bad credit" based on the available features.

Here are some details about the dataset:

- Number of samples: 1,000
- Number of features: 9 (excluding the target variable)
- Number of "good credit" samples: 700
- Number of "bad credit" samples: 300

The dataset includes features such as age, sex, credit history, job stability, housing status, savings accounts, checking account status, purpose of the loan, and the risk (class label) associated with each applicant. Below you may see the shapes of test, validation and train set:

```
The Shape of the Train Set is:  (600, 9)
The Shape of the Validation Set is:  (200, 9)
The Shape of the Test Set is:  (200, 9)
```

For each of the 3 sets, below you may the examples regarding their entries:

```
Example entry of the Train Set:
 Age   Sex  Job  Housing  Saving accounts  Checking account  Credit amount  Duration  Purpose
  63 0.000    2    0.000            0.000             0.000           6836        60    4.000
  33 0.000    2    2.000            0.000             2.000           2319        21    1.000
  50 0.000    2    2.000            2.000             0.652           1236         6    3.000
  29 1.000    2    0.000            0.457             0.652           5003        21    3.000
  21 1.000    2    0.000            0.457             0.652            886        12    0.000

Example entry of the Test Set:
 Age   Sex  Job  Housing  Saving accounts  Checking account  Credit amount  Duration  Purpose
  24 1.000    2    0.000            0.000             0.000           3190        18    0.000
  35 0.000    1    0.000            1.000             0.000           4380        18    3.000
  32 0.000    2    0.000            1.000             0.000           2325        24    3.000
  23 0.000    2    2.000            0.000             2.000           1297        12    0.000
  35 0.000    3    0.000            0.000             0.652           7253        33    3.000

Example entry of the Validation Set:
 Age   Sex  Job  Housing  Saving accounts  Checking account  Credit amount  Duration  Purpose
  34 0.000    2    0.000            3.000             1.000           3496        30    2.000
  52 1.000    1    0.000            1.000             0.652            362         6    3.000
  26 1.000    2    0.000            0.000             2.000           3749        24    2.000
  56 0.000    2    1.000            0.000             2.000           4796        42    3.000
  25 0.000    2    0.000            0.457             1.000           1484        12    0.000
```

The output provides a structured representation of the datasets, making it easier to view and analyze the data. Each entry's values can be referred to based on the column they belong to, allowing for a better understanding of the dataset's structure and content.

3. Methodology

Exploratory Analysis (EDA)

Exploratory Data Analysis (EDA) is a crucial step in the data analysis process. It involves examining and understanding the data to gain insights, discover patterns, detect anomalies, outliers and missing data. Below are some reasons why performing exploratory analysis is important:

1. Data Understanding: EDA helps in developing a deep understanding of the dataset, its structure, and the relationships between variables can be seen easily with EDA. It allowed us to identify the data types, and understand the meaning and significance of each variable.

2. Handling Missing Values: EDA helps to identify and handle missing values, outliers, and inconsistencies in the data. By examining summary statistics and visualizations, we spot data quality issues that might affect the validity and reliability of the analysis.

3. Pattern Detection: EDA allows you to identify patterns, trends, and relationships within the data. By visualizing and analyzing the data, we aim to detect underlying patterns

4. Feature Selection: EDA can assist in determining which features or variables are most relevant for the analysis or modeling task. By exploring the relationships between variables, you can identify variables that have a strong correlation with the target variable and eliminate irrelevant or redundant features.

5. Hypothesis Generation: EDA is a crucial step in hypothesis generation. By exploring the data, you can form hypotheses about the relationships between variables or make assumptions about the data distribution. These hypotheses can guide further analysis or experimentation.

6. Data Visualization: EDA involves creating visual representations of the data, such as plots, charts, and graphs. Visualizations make it easier to understand complex patterns, spot outliers, and communicate findings effectively. Visual representations can provide insights that are not immediately evident from tabular data.

Overall, exploratory analysis is essential for understanding and gaining insights from the data. It helps in detecting data issues, identifying patterns, formulating hypotheses, and making informed decisions throughout the data analysis process.

At the starting of our experiments we first filled the missing values (NA values) with zeros. Then, We investigated the insights related to the dataset:

```
Dataset Details:
Number of samples: 1000
Number of features: 10
Number of 'good credit' samples: 700
Number of 'bad credit' samples: 300
```

Moving on, we checked whether there are any duplicate values inside the dataset and found out that there are none. After that to analyze and understand the data in a deeper manner we conducted a statistical analysis.

Below you may see the table:

	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	1,000.000	499.500	288.819	0.000	249.750	499.500	749.250	999.000
Age	1,000.000	35.546	11.375	19.000	27.000	33.000	42.000	75.000
Sex	1,000.000	0.310	0.463	0.000	0.000	0.000	1.000	1.000
Job	1,000.000	1.904	0.654	0.000	2.000	2.000	2.000	3.000
Housing	1,000.000	0.466	0.779	0.000	0.000	0.000	1.000	2.000
Saving accounts	1,000.000	0.457	0.786	0.000	0.000	0.000	0.457	3.000
Checking account	1,000.000	0.652	0.514	0.000	0.000	0.652	1.000	2.000
Credit amount	1,000.000	3,271.258	2,822.737	250.000	1,365.500	2,319.500	3,972.250	18,424.000
Duration	1,000.000	20.903	12.059	4.000	12.000	18.000	24.000	72.000
Purpose	1,000.000	2.096	1.630	0.000	0.000	2.000	3.000	7.000

The provided statistics are descriptive statistics for various variables in the German credit dataset. Here's an explanation of each statistic:

- Count: The number of observations or data points for each variable. In this case, there are 1,000 data points for each variable.
- Mean: The average value of the variable across all observations. It represents the central tendency of the data. For example, the mean age is 35.546, indicating that the average age of the individuals in the dataset is around 35.546 years.
- Std: The standard deviation measures the variability of the data around the mean. It provides information about the spread of the data points. A higher standard deviation indicates a greater spread of data points from the mean.
- Min: The minimum value observed for the variable. It represents the smallest value in the dataset. For example, the minimum age is 19, indicating that the youngest individual in the dataset is 19 years old.
- 25%: The first quartile or 25th percentile. It represents the value below which 25% of the data points fall. It is a measure of the data's spread and provides information about the lower end of the data distribution.
- 50%: The second quartile or median. It represents the value below which 50% of the data points fall. It is a measure of the central tendency of the data and divides the data into two equal halves.
- 75%: The third quartile or 75th percentile. It represents the value below which 75% of the data points fall. It is another measure of the data's spread and provides information about the upper end of the data distribution.
- Max: The maximum value observed for the variable. It represents the largest value in the dataset. For example, the maximum age is 75, indicating that the oldest individual in the dataset is 75 years old.

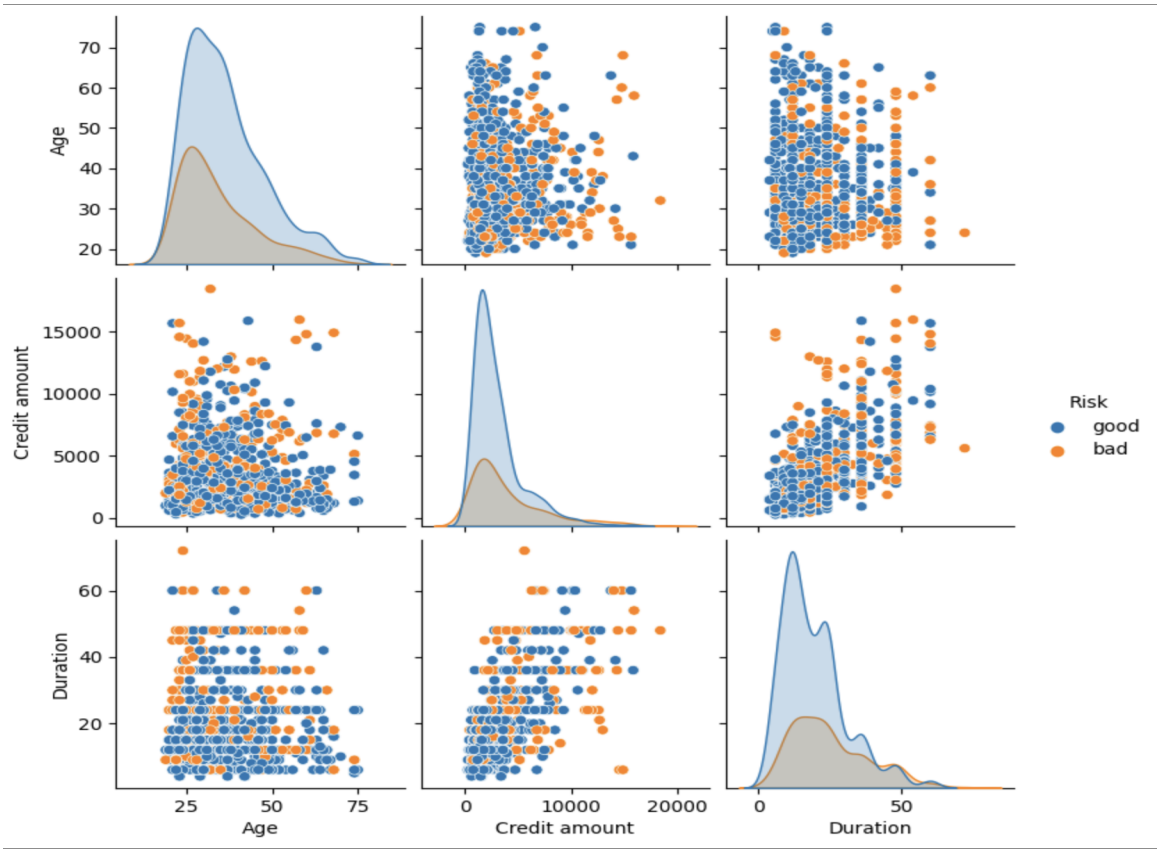
By examining these statistics, we get an overview of the distribution, range, and central tendency of each variable in the German credit dataset. It provides insights into the characteristics and variability of the dataset, which can inform further analysis and decision-making processes.

Data Visualizations

Initially, we have generated a pair plot of the numerical variables 'Age', 'Credit amount', and 'Duration' from the dataset, with the points colored according to the 'Risk' variable.

A pair plot is a grid of scatter plots that allows you to visualize the relationships between multiple variables. In this case, the pair plot will show the pairwise relationships between the selected numerical variables. The diagonal of the pair plot will show the distribution of each variable. The points in the scatter plots will be colored based on the 'Risk' variable. Each unique value of 'Risk' will be assigned a different color, allowing us to visually analyze the relationships between the numerical variables while considering the risk category.

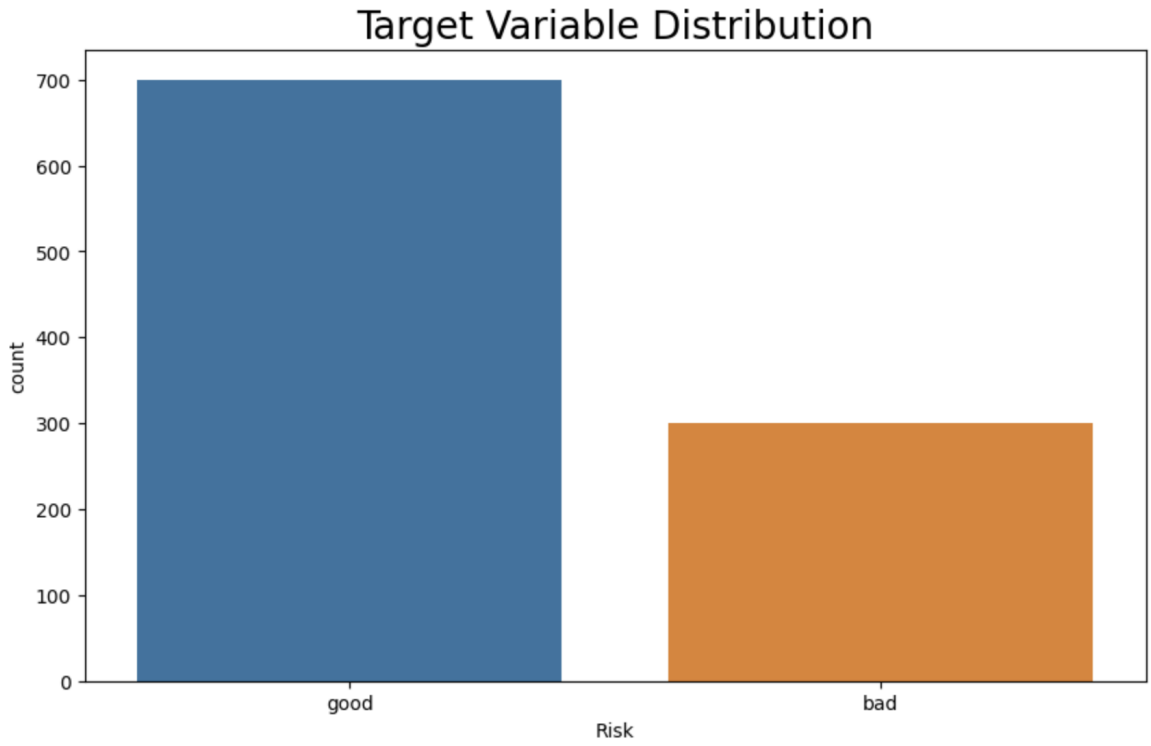
Below you can see the graphs:



Moving on, we have checked the target variable distribution. The results of the graph indicate the distribution of classifications in the dataset.

Specifically, it shows that out of the total instances considered, 700 applicants were classified as "good" and 300 applicants were classified as "bad". This information provides an understanding of the class distribution within the dataset. In this case, there is an imbalance in the classes, with a higher number of instances classified as "good" compared to "bad. Analyzing class distributions is important in machine learning and data analysis tasks. Class imbalances can impact model performance and introduce biases. It's important to consider the class distribution while interpreting the results and designing strategies for modeling or analysis.

Below is the resulting graph:

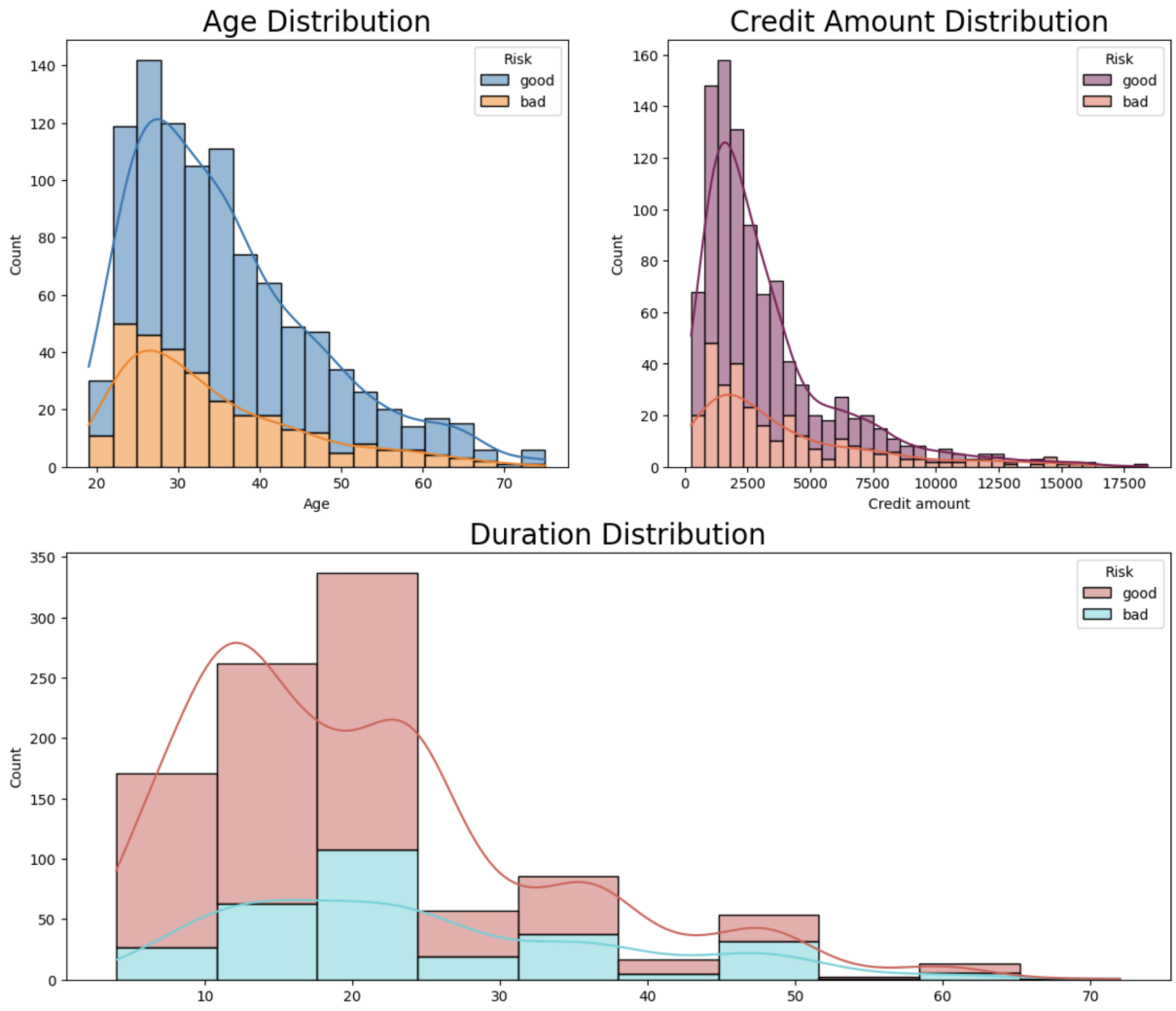


We have also analyzed the age distribution; the results reveal several interesting trends. The graphs display a positive skew, indicating that the mean age is greater than the median age. This suggests a higher presence of older individuals in the dataset. Additionally, it is noteworthy that applicants between the ages of 20 and 30 are more likely to apply for a loan, indicating a specific demographic's interest or need for financial assistance. Furthermore, there is a tendency for applicants to request lower credit amounts, as evidenced by the observation that they are less likely to apply for high credit loans.

On the repayment front, it appears that more loans have been paid off approximately 20 months after being issued. Lastly, the bank tends to receive applicants between the ages of 20 and 30, who typically request loans ranging from 250 to 2500.

These insights into the age distribution and its relationship with loan applications, loan amounts, and repayment behavior are valuable for strategic decision-making, risk assessments, and the development of targeted financial products and services.

Below are the resulting graphs:



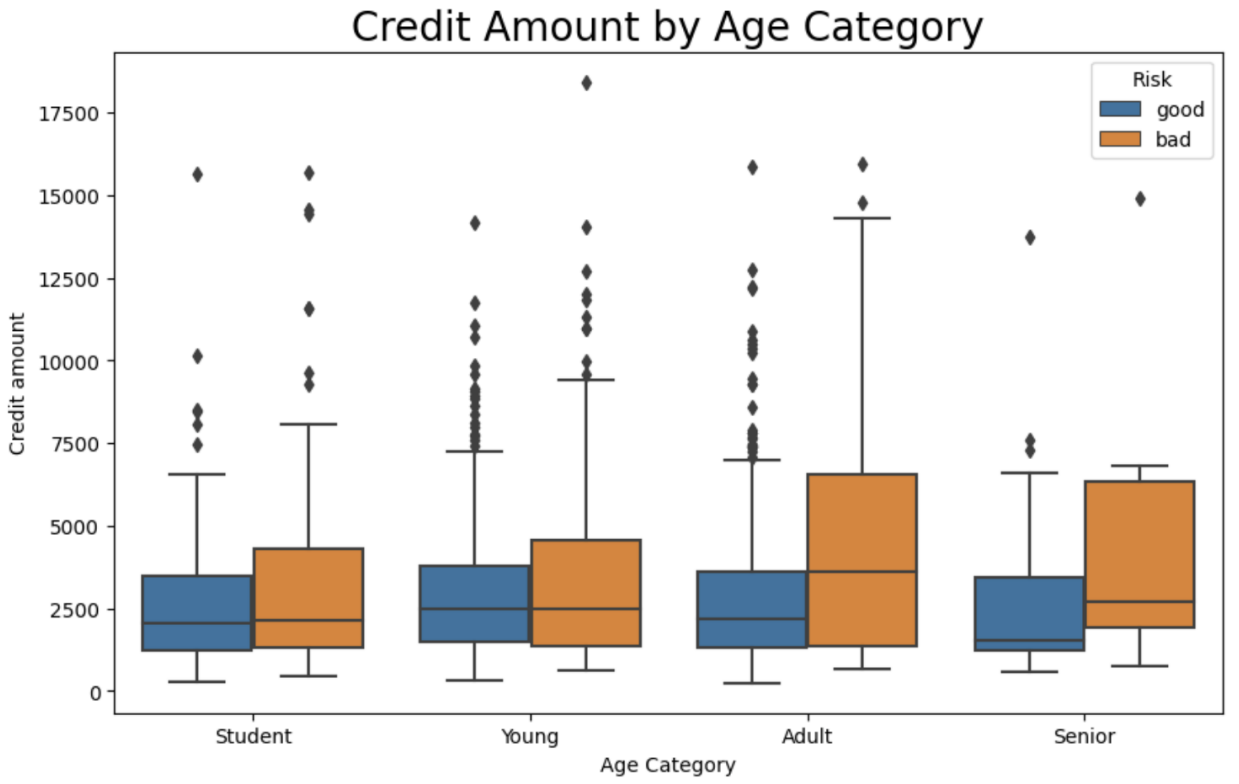
Moving on, we checked the credit amount by age category. It appears that more than 50% of the applicants with credit amounts below 5,000 DM are classified as good, suggesting that lower credit amounts are associated with a higher likelihood of being considered as good applicants.

On the other hand, it seems that adults with loan credit greater than 5,000 DM are more likely to be classified as bad applicants, indicating that higher loan amounts may pose a higher risk factor.

Additionally, the data indicates that students and young applicants are most likely to apply for loans with a credit amount of less than 5,000 DM, implying that this age group prefers smaller loan amounts.

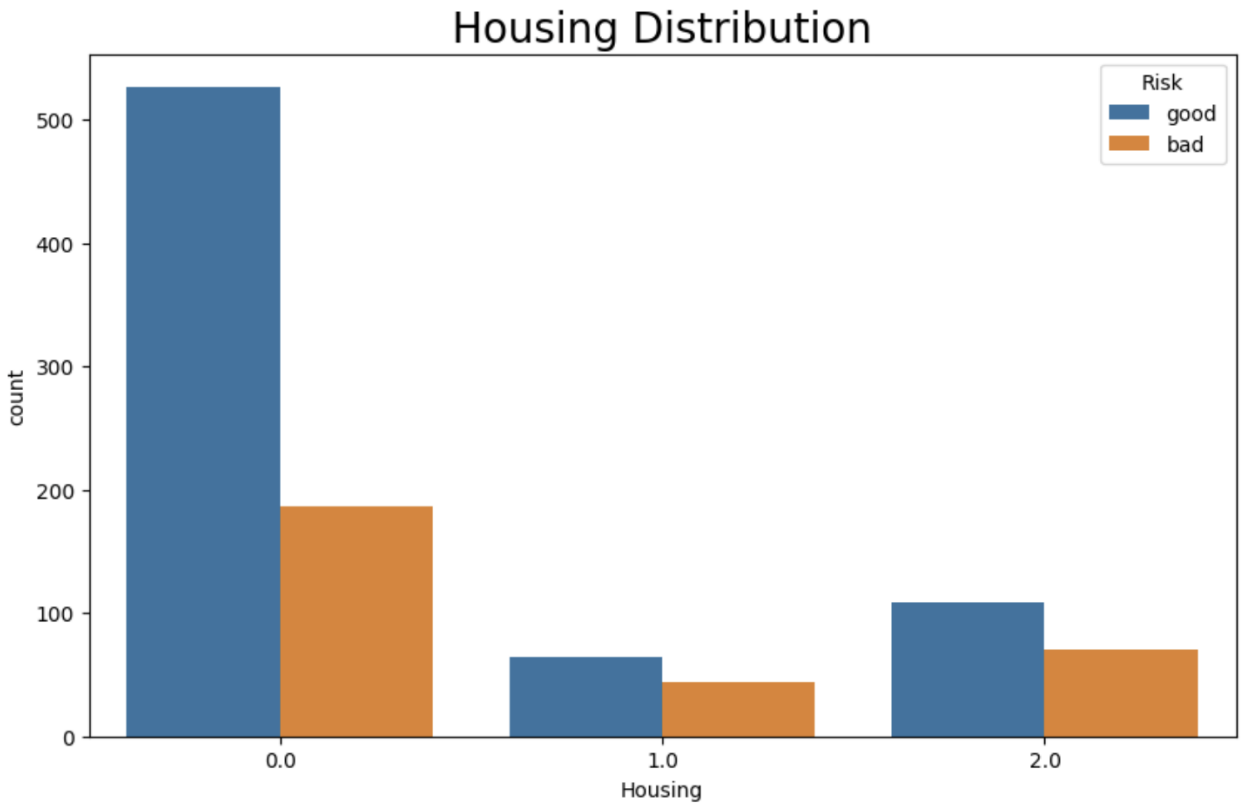
These findings shed light on the relationship between credit amounts, applicant classification, and age groups, which can assist in making informed decisions when assessing loan applications and tailoring loan products to different target demographics.

Here is the related graph:



Next distribution we considered is the housing distribution; it can be seen that a majority of the applicants in the dataset own a house, suggesting that home ownership is common among the applicants. Moreover, it is striking that over half of the applicants who own a house are classified as good applicants. This indicates that home ownership may be correlated with a higher likelihood of being considered as a good applicant. The relationship between house ownership

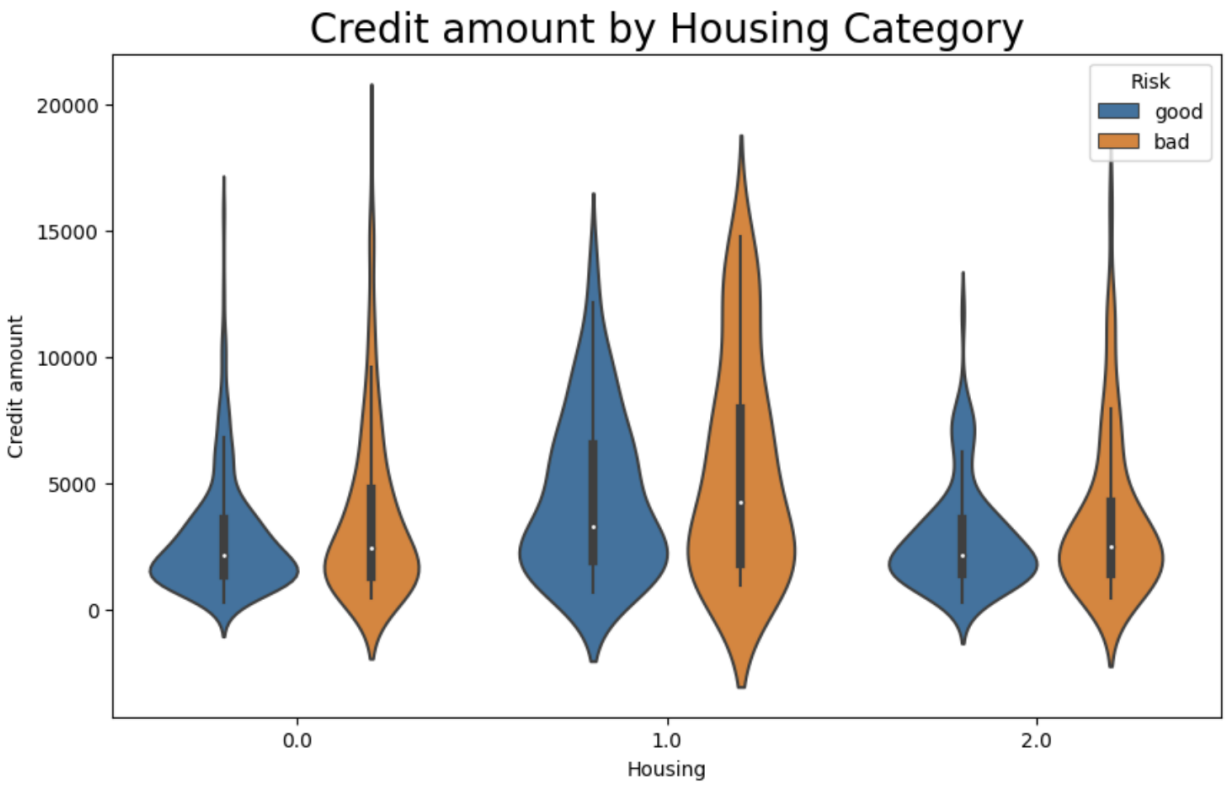
and applicant classification provides valuable insights for assessing creditworthiness and making informed decisions regarding loan applications. Below is the graph of the housing distribution:



The next distribution is the credit amount by housing distribution. The Credit Amount by Housing Distribution results, as visualized in the graph through the violin plot's density curves, provide interesting insights. In a violin plot, each category or group is represented by a vertical "violin" shape. The width of the violin at any given point represents the estimated density of data points at that value. In other words, it shows the relative frequency or concentration of data points along the range of the variable. The density curve within the violin plot represents the estimated probability density function of the data. It provides information about the shape, spread, and central tendency of the distribution within each category or group. The density curves indicate that there is a higher frequency of applicants with credit amounts below 5,000 DM. This suggests that a significant portion of the applicants tends to apply for lower credit amounts. Understanding this trend can help in assessing the demand for different loan sizes and tailoring loan products accordingly. The Credit Amount by Housing Distribution graph offers

valuable information for financial institutions to effectively meet the borrowing needs of their customers.

Here is the graph:



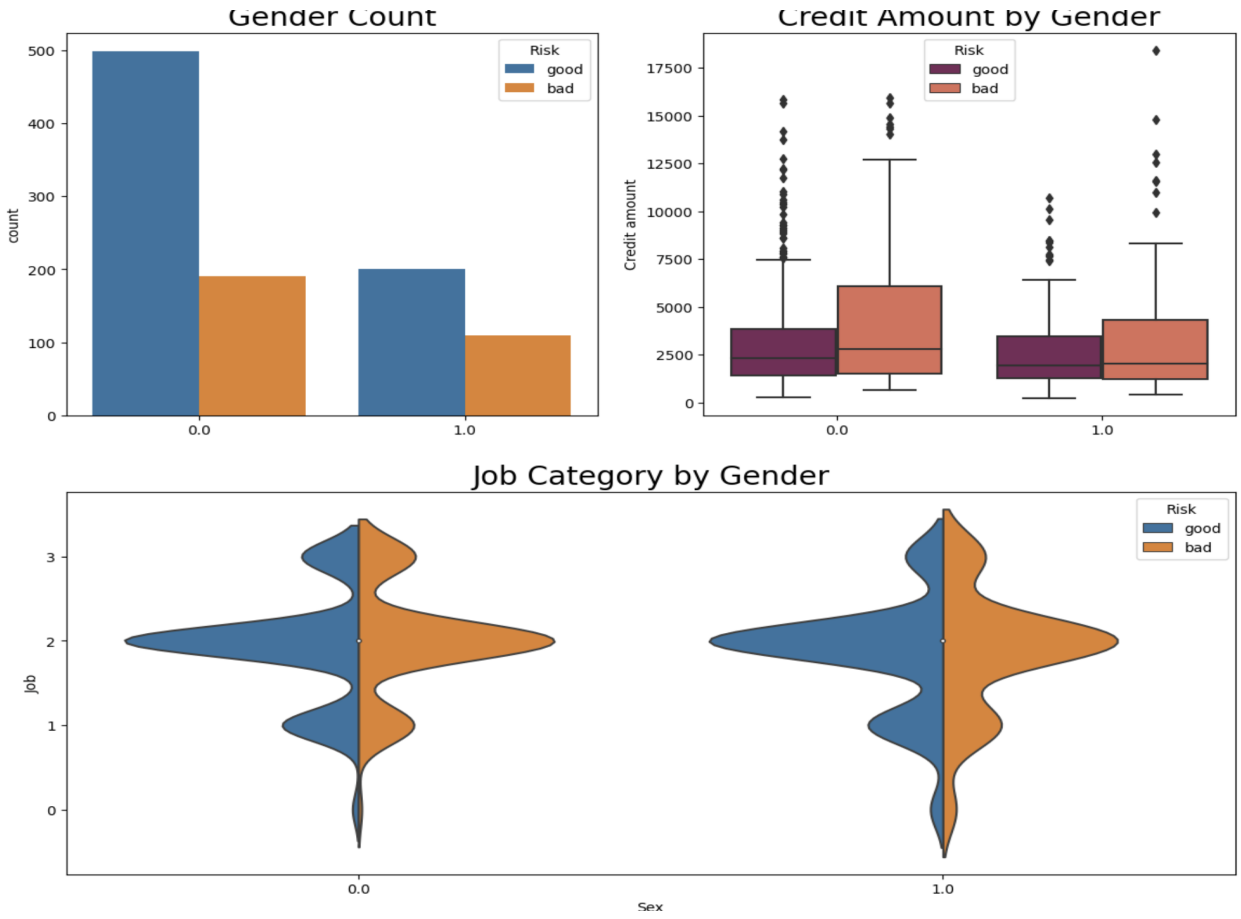
The next distribution that we focus on is the distribution by gender. The distribution by gender results, depicted in the graph, offer several notable findings. Firstly, there is a significant gender imbalance in the dataset, with approximately twice as many male applicants compared to females. This indicates a higher representation of male applicants in the data. Furthermore, the majority of applicants fall into the skilled job category. This suggests that a significant proportion of the applicants possess skilled job qualifications.

In terms of applicant classification, it is interesting to observe that approximately 2/5 of male applicants and 1/3 of female applicants are classified as bad. This implies that a considerable proportion of both male and female applicants pose a higher credit risk. The density distributions from the violin plots also reveal an intriguing trend. The density curves for both male and female applicants follow a similar pattern, indicating that the majority of applicants from both genders

are categorized as skilled workers. This consistency suggests that skilled occupations are prevalent among both male and female applicants.

Overall, these insights shed light on the gender distribution, job categories, applicant classifications, and their relationships within the dataset. Understanding these trends can inform credit risk assessments, marketing strategies, and decision-making processes within the context of gender and job categories.

Below you may see the graphs:



Moving on, we checked the savings account distribution. Firstly, it appears that applicants with little or no savings accounts are more likely to apply for loans. This suggests that individuals who have limited savings or no savings at all are more inclined to seek financial assistance through loans. Additionally, the majority of the applicants fall into the "little" savings account category. It is worth noting that approximately 50% of the applicants in this category are between the age range of 25 and 45. This indicates that a significant proportion of individuals within this

age range have limited savings. Furthermore, applicants with moderate, quite rich, and rich savings accounts are more likely to be classified as good applicants. This implies that individuals who have substantial savings are considered to have a lower credit risk and are more likely to be approved for loans. On the other hand, applicants with little and no savings accounts, who also have a credit amount loan exceeding 5,000 DM, are more likely to be classified as bad applicants. This suggests that individuals with inadequate savings and high loan amounts are considered to pose a higher credit risk.

Overall, these findings emphasize the relationship between savings accounts, applicant classifications, and age groups. Understanding these trends can be beneficial in assessing creditworthiness, making informed decisions regarding loan applications, and tailoring financial products to different customer segments.

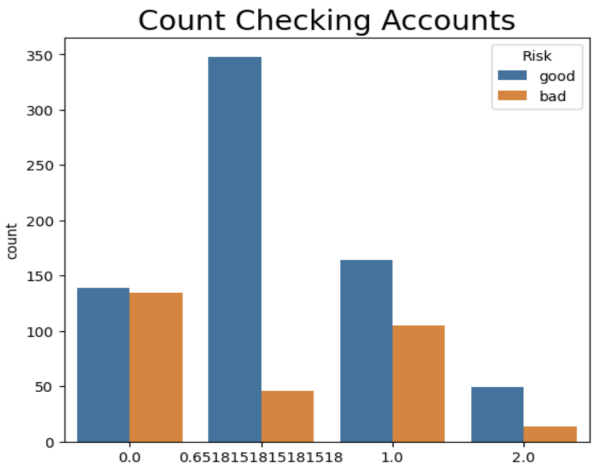
You may see the related graphs below:

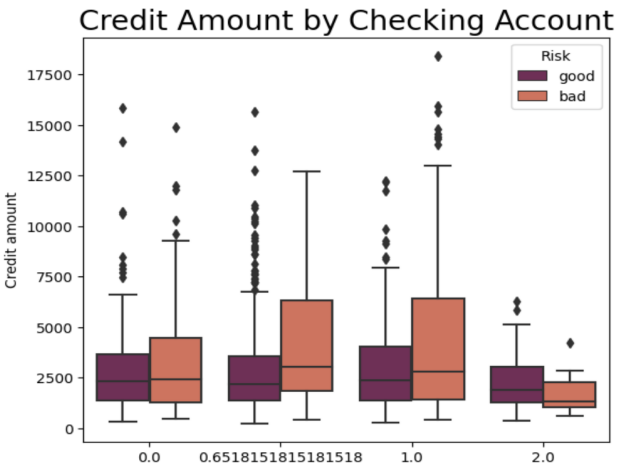
We also checked the distribution of the checking account. It is notable that more than 300 applicants in the dataset do not have a checking account. This indicates a significant portion of the applicants who rely on alternative financial management methods. Moreover, among the applicants without checking accounts, more than three times the number of applicants were classified as good. This suggests that not having a checking account does not necessarily indicate a higher credit risk.

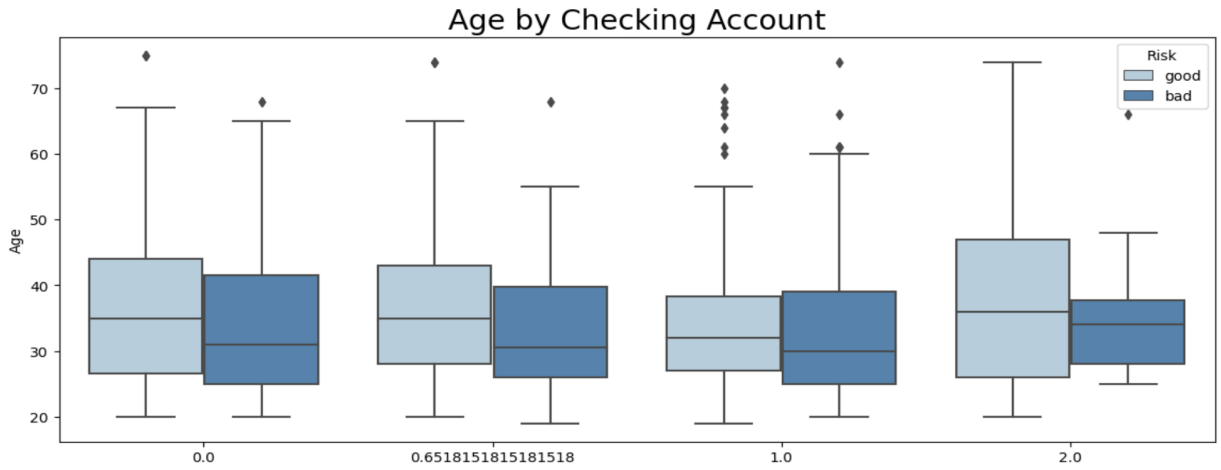
Additionally, it is worth mentioning that approximately 50% of the applicants with moderate checking accounts fall within the age range of 25 to 40. This implies that individuals in this age range are more likely to have a moderate level of financial activity and usage of checking accounts. Furthermore, there is a greater dispersion of applicants with rich checking accounts who are classified as good and fall within the age range of 25 to 45. This suggests that individuals in this age range with higher financial activity and substantial checking account balances are more likely to be considered low credit risk applicants. Lastly, applicants with a high credit amount and minimal funds in their checking accounts are more likely to be classified as bad applicants. This indicates that individuals with significant loan amounts but limited checking account balances are deemed to have a higher credit risk.

Overall, these findings underscore the relationship between checking accounts, applicant classifications, and age groups. Understanding these patterns can aid in credit risk assessments, loan approvals, and tailoring financial products to meet the needs of different customer segments.

Graphs are stated below:

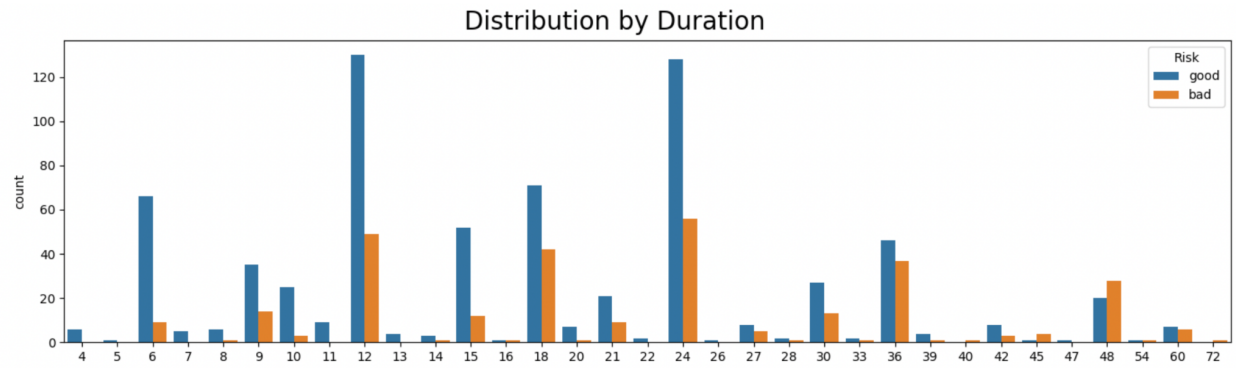


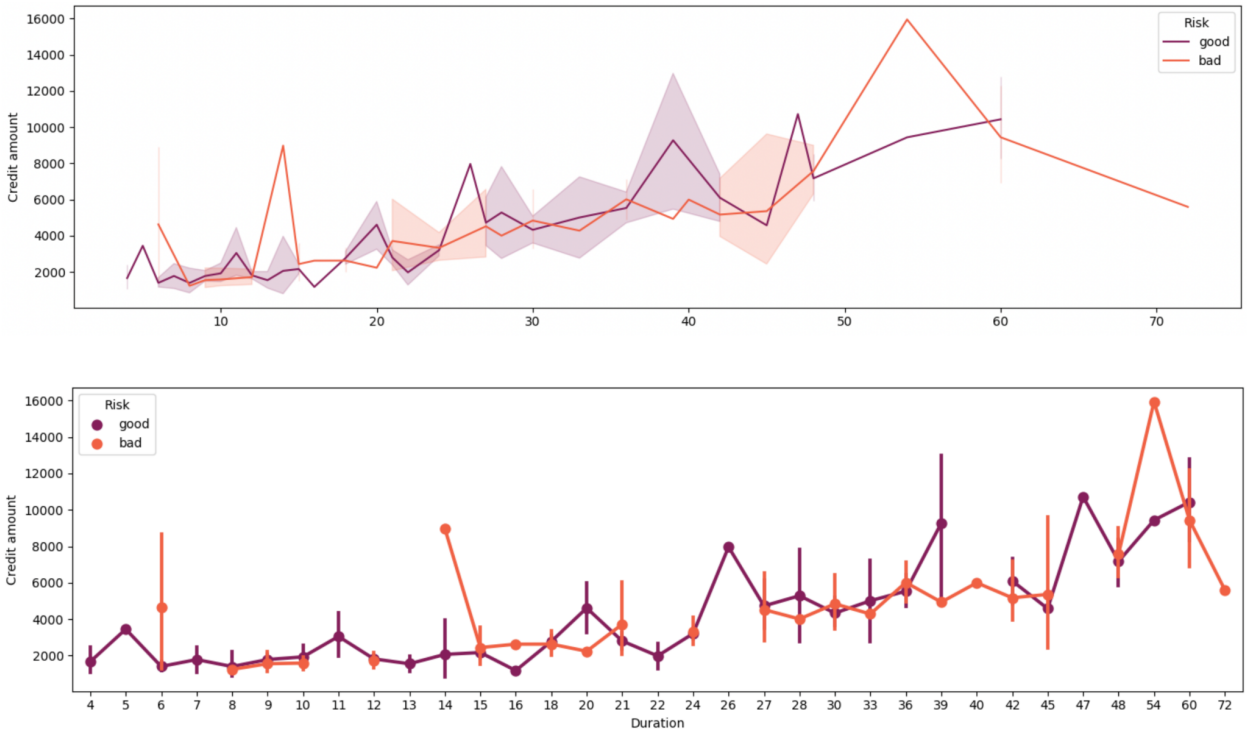




Lastly, we checked distribution by the duration values. It is evident that a significant number of loans issued had durations of 12 and 24 months. This indicates that these two durations are the most common choices among applicants, suggesting their preference for shorter-term repayment plans. Moreover, among the applicants who repaid their loans within 24 months, the majority are classified as good. This implies that applicants who successfully complete their loan repayments within a shorter duration are more likely to be considered creditworthy.

On the other hand, it is notable that most applicants with loan durations exceeding 24 months are classified as bad. This suggests that longer-term loan commitments may present higher credit risks, as a greater proportion of applicants in this category fail to meet their repayment obligations. These insights highlight the relationship between loan duration, applicant classifications, and creditworthiness. Understanding these trends can help in assessing repayment behavior, identifying optimal loan durations, and managing credit risk effectively. Here are the respective graphs:





After completing the exploratory data analysis and analyzing the German Credit Dataset, we have started on several algorithms to build models that provide insights into credit risk assessment.

3.1 K-Nearest Neighbor Algorithm

The first classification problem method used in our project is the K Nearest Neighbor algorithm. The German Credit Dataset's features are used by the KNN algorithm in this project to determine whether a loan application is likely to have good or bad credit. The main objective is to find the optimal value of k, which represents the number of nearest neighbors considered for classification. The code utilizes the scikit-learn library to perform various steps.

In the implementation of this algorithm, we first defined a list of k values which represents the number of neighbors to consider during our classification task. Then, we defined some variables to store the validation accuracies, the best accuracy and the corresponding best k value. Our next step is followed by creating a for loop that iterates each k value in the k_values list. A k-NN classifier is initialized with the current k value inside the loop. The modified training data is used

to train the classifier, while the transformed validation data is used to make predictions. Utilizing accuracy_score, the prediction accuracy is calculated and stored in the val_accs list.

If the current accuracy is higher than the previous best_acc throughout each iteration, the best_acc and best_k variables are changed. The best validation accuracy and associated k value are printed following the loop. The outputs of the code can be seen below:

```
Test accuracy for k = 1: 0.9300
Test accuracy for k = 3: 0.7950
Test accuracy for k = 5: 0.7200
Test accuracy for k = 7: 0.7650
Test accuracy for k = 9: 0.7350
Test accuracy for k = 11: 0.7250
Test accuracy for k = 13: 0.7150
Test accuracy for k = 15: 0.7100
Test accuracy for k = 17: 0.7100
Best validation accuracy ( 0.93 ) is achieved with k = 1
```

Then, using matplotlib, a plot is made to show the connection between k values and validation accuracies. The validation accuracies are represented on the y-axis, and the k values are plotted on the x-axis.

And finally, the final k-NN classifier is then trained by the algorithm utilizing the combined training and validation data. On the merged set, the feature selection is carried out, and the classifier is trained using the altered data. The classifier's predictions are then derived for the test set, and their accuracy is calculated and shown.

In conclusion, the best value of k is chosen through validation to get the highest accuracy on the test set. The k-NN algorithm is utilized in this problem to categorize loan applicants as having either good or bad credit depending on the specified attributes.

3.2 Logistic Regression

Our second classification method used in the project is logistic regression. For our project, logistic regression is well-suited for binary classification tasks, where the goal is to classify instances into one of two classes, such as "good credit" or "bad credit." It models the relationship

between the independent variables (features) and the probability of belonging to a certain class. In credit risk assessment, logistic regression can be used to estimate the probability of a loan applicant having good credit.

We started with preprocessing the data by defining the categorical and the numerical variables of our data named **categorical_features** and **numerical_features.** This is done because the categorical variables cannot be directly used in logistic regression models, as they require numerical representations. So, we defined a pipeline named categorical_transformer.

First, an instance of SimpleImputer is created with the strategy set to 'most_frequent'. It will handle missing values in categorical features by replacing them with the most frequent value.

Second, an instance of OneHotEncoder is created with the 'ignore' strategy for handling unknown categories. It will one-hot encode the categorical features. Then a ColumnTransformer named preprocessor is defined to apply the appropriate transformations to numerical and categorical features. It consists of two transformers: 'num' for numerical features using numerical_transformer and 'cat' for categorical features using categorical_transformer.

After all, we defined a pipeline to integrate the preprocessing steps, feature selection, and the classification model. It consists of three steps: 'preprocessor' using preprocessor, 'selector' using selector, and 'model' using LogisticRegression with the 'liblinear' solver.

We splitted our data into 80% train and 20% test dataset using the train_test_split method. Then for hyperparameter tuning, we defined a dictionary to specify the hyperparameter grid for tuning. This is for finding the best combination that maximizes the model's performance. And also, the classifier pipeline (clf), hyperparameter grid (param_grid), and 5-fold cross-validation (cv=5) are used to generate an instance of GridSearchCV, which is to select the optimal model based on the given hyperparameters, grid search is used.

Lastly, we made a model evaluation in which the grid search results are used to determine the optimal model and the associated hyperparameters. The most effective model is a logistic regression model with the right mix of performance-enhancing hyperparameters.

Predictions are made on the test set using this best model, and the predicted labels are saved in y_pred.

Then, a number of evaluation metrics are calculated and printed. By contrasting the predicted labels with the actual labels from the test set, the accuracy of the model is determined. Additionally printed is the confusion matrix, a summary of the true positive, true negative, false positive, and false negative forecasts. This matrix gives a thorough assessment of the model's performance for each class in terms of accurate and inaccurate predictions.

Additionally, the grid search's best hyperparameters are shown, giving information about the precise values of the hyperparameters that produced the greatest model performance.

The confusion matrix is then used to extract the values of true positives, true negatives, false positives, and false negatives. These metrics provide more thorough information on the model's prediction skills by describing the model's performance in terms of examples that were successfully and wrongly classified for each class.

3.3 Decision Tree

In our project, the decision tree algorithm is the third classification method used. It employs the features from the German Credit Dataset to predict the creditworthiness of loan applications. The decision tree constructs a hierarchical structure based on different attributes and their values to partition the data. By recursively splitting the data, the algorithm aims to create subsets that are homogeneous in terms of credit risk.

We utilized the decision tree algorithm as a classification method in our project. To enhance the model's performance, we incorporated feature selection using the `SelectKBest` function from the scikit-learn library. By applying the chi-square scoring function and specifying `k=7`, we aimed to identify the top 7 features that exhibited the strongest relationship with the target variable, which in our case is creditworthiness.

To optimize the decision tree's hyperparameters, we employed grid search coupled with cross-validation. We defined a `param_grid` dictionary, specifying the hyperparameters to be tuned, such as `max_depth` (representing the maximum depth of the decision tree) and `min_samples_split` (representing the minimum number of samples required to split a node). We provided multiple values for each hyperparameter, including `[None, 10, 20]` for `max_depth` and `[2, 5, 10]` for `min_samples_split`.

Then, we utilized the `GridSearchCV` class from scikit-learn, which performs an exhaustive search over the specified hyperparameter grid using cross-validation. By passing the `DecisionTreeClassifier` as the estimator and the defined `param_grid`, we trained and evaluated decision tree models with different hyperparameter combinations. The use of 5-fold cross-validation helped to assess the models' performance and avoid overfitting.

Once the grid search was completed, we extracted the best hyperparameters by accessing the `best_params_` attribute of the grid search object. These optimal hyperparameters were then used to train the final decision tree model. We instantiated a new `DecisionTreeClassifier` with the best hyperparameters and fitted it to the training data containing the selected features obtained from the feature selection step.

To evaluate the performance of our model, we applied the trained decision tree on the test data. We calculated the test accuracy using the `score` method, which compares the predicted labels with the true labels. This accuracy metric provided us with a quantitative measure of how well our decision tree classifier performed in predicting creditworthiness.

By following this detailed approach, including feature selection, hyperparameter tuning through grid search with cross-validation, and evaluation on the test data, we aimed to develop an effective and robust decision tree model for credit risk assessment in our project.

4. Experiments

In this part of our report, we will be presenting the final performances of all the algorithms we have tried.

4.1 KNN

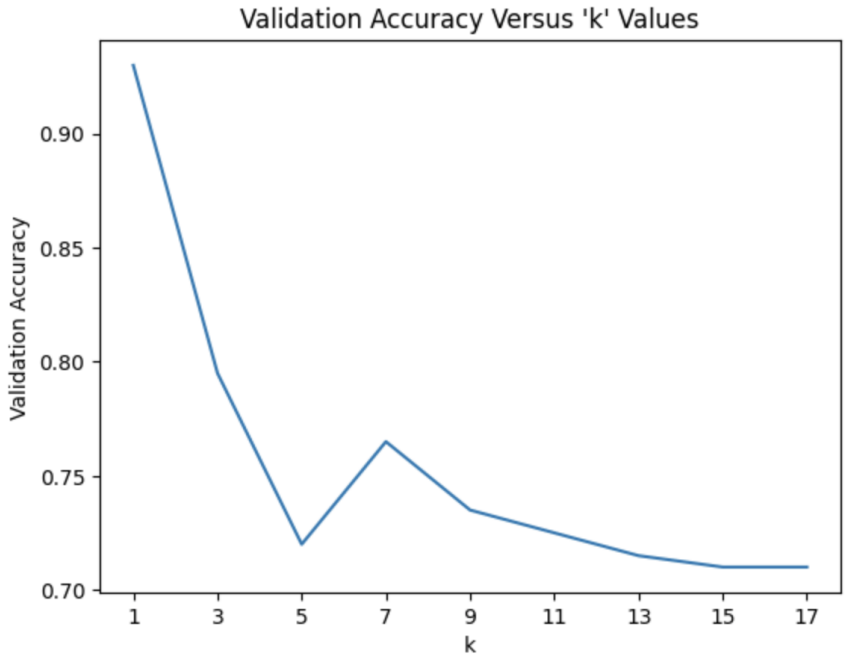
To begin with, we will be analyzing the results of our first algorithm which is KNN. As we have discussed in section 3.1, we have made an experiment to find the best k value for a k-Nearest Neighbors classifier by iterating over a list of k values. It trains the classifier, and computes the accuracy on the validation set. The code tracks the best accuracy achieved and the corresponding k value, outputting the results at the end.

The results of the test accuracies are given below:

```
Test accuracy for k = 1: 0.9300
Test accuracy for k = 3: 0.7950
Test accuracy for k = 5: 0.7200
Test accuracy for k = 7: 0.7650
Test accuracy for k = 9: 0.7350
Test accuracy for k = 11: 0.7250
Test accuracy for k = 13: 0.7150
Test accuracy for k = 15: 0.7100
Test accuracy for k = 17: 0.7100
Best validation accuracy ( 0.93 ) is achieved with k = 1
```

When we examine the results we see that the best validation accuracy of 0.93 is achieved with k = 1. This means that considering 1 nearest neighbor during classification provides the most accurate predictions on unseen data. We also visualized a plot to see the relationship between k values and validating accuracies.

Here is the graph:



As can be seen from the graph, the highest accuracy among the k values belongs to k = 1.

And lastly, we combined the training and validation sets to train our final k-NN classifier, the classifier is trained on the transformed data. Finally, the classifier's predictions are obtained on the test set, and the accuracy of the predictions is computed and printed. Here is the result for the best_k:

```
Test accuracy for best k = 1: 0.9300
```

We see that the best k value and its accuracy is the same with our training set. Which means that the trained model is performing consistently well on both the data of seen and unseen values. This means that the model has successfully learned the pattern.

4.2 Logistic Regression

In this algorithm, the goal was to assess the performance of various classification algorithms for credit risk assessment. The experiments followed a pipeline approach, starting with appropriate preprocessing steps. The best model was selected based on accuracy, and its performance was evaluated using various metrics, including accuracy, confusion matrix, and true positives/negatives and false positives/negatives. Here are the results of our method:

```
Accuracy: 0.76

Confusion Matrix:
[[ 22  37]
 [ 11 130]]

Best Hyperparameters: {'model__C': 1.0}

True Positives: 130
True Negatives: 22
False Positives: 37
False Negatives: 11
```

The model's accuracy was 0.76, which indicates that 76% of the test set's instances were properly classified.

The confusion matrix analysis demonstrates that the model properly predicted 22 true negatives and 37 true positives out of the 59 occurrences classified as "good credit," suggesting occasions where bad credit was really anticipated to exist but did not. Similar to this, out of the 141 instances classified as having "bad credit," the model accurately forecasted 130 as true positives and 11 as false negatives, denoting situations where good credit truly existed but was misclassified as having "bad credit."

'model__C', which had a value of 1.0, was the best hyperparameter discovered by the grid search. The regularization strength in logistic regression is controlled by this hyperparameter.

These findings collectively imply that the model has the potential to distinguish between applicants with good credit and those with bad credit. The false positive and false negative rates are not negligible, thus there is opportunity for improvement.

Additional testing and model improvement, such as investigating various techniques or modifying hyperparameters, may help to improve accuracy and reduce misclassifications.

4.3 Decision Tree

We used the `DecisionTreeClassifier` from scikit-learn to perform a grid search with cross-validation and plot the accuracy results for different values of hyperparameters `max_depth` and `min_samples_split`.

After defining the hyperparameters to search over in the `param_grid` dictionary, we instantiated a `DecisionTreeClassifier` object `dt_clf`. We then utilized the `GridSearchCV` class to perform a grid search with 5-fold cross-validation. The grid search was conducted by fitting the classifier on the training data (`X_train` and `y_train`) and searching for the best combination of hyperparameters specified in `param_grid`.

We obtained the best hyperparameters and corresponding scores by accessing the `best_params_` and `best_score_` attributes of the grid search object, respectively. Additionally, we calculated the test accuracy using the `score` method on the test data (`X_test` and `y_test`) to evaluate the model's performance.

To visualize the impact of different values of `max_depth` and `min_samples_split` on the accuracy, we created a figure with two subplots using `plt.subplots()`. In the first subplot (`ax1`), we plotted the accuracy as a function of `max_depth`, iterating over the provided values of `max_depths` and fitting the decision tree classifier for each value. Similarly, in the second subplot (`ax2`), we plotted the accuracy as a function of `min_samples_split` by iterating over the specified `min_samples_splits`.

For both subplots, we stored the accuracies in `accuracies1` and `accuracies2` lists, respectively. The marker `'o'` was used to indicate the data points on the plots. We also set appropriate labels for the x-axis and y-axis, as well as titles for each subplot based on the selected best hyperparameters.

Finally, we displayed the plot using `plt.show()`, allowing us to visualize the relationship between the hyperparameters and the corresponding accuracy scores. This visual representation can assist in selecting the optimal hyperparameters for the decision tree classifier in our project.

5. Discussion

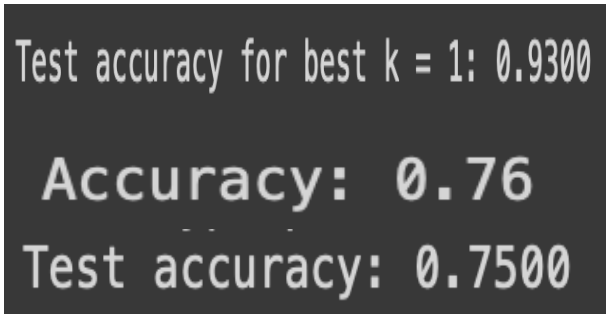
In this project, among many machine learning algorithms, we tried to implement 3 of the most known ones and ones that fits to our model better such as K-NN(k nearest neighbor), Linear Regression and Decision Tree. The purpose of this project was to determine the best machine learning algorithm among these 3 by calculating the accuracy of the models. Even though the algorithms and the implementation lay behind the models are different, our models gave similar results. But similar does not mean the same. Among these 3 models, the best model that yields the maximum accuracy is K-NN. It is significantly better than the other two models.

With these results, we know that even the accuracy of K-NN is not good enough to leave the model as it is. However, if we want to improve our model to use in a real life situation, we decided to use K-NN since it is significantly better than both Decision Tree and Linear Regression.

K-NN model:

Logistic Regression:

Decision Tree:



6. Bonus

In our endeavor to improve the predictive performance of our models, we have implemented additional enhancements as part of the bonus section of our work. The technique we employed was feature selection, a process that involves automatically selecting those features in our data that contribute most to the prediction variable or output we are interested in. The utility of

feature selection lies in its ability to significantly increase the accuracy of a model if the right subset is chosen. It does this by reducing overfitting, improving accuracy, and reducing training time.

We chose the SelectKBest class from the sklearn.feature_selection module for our feature selection process. This class selects features according to the k highest scores of a specified scoring function, in our case, chi2. We have chosen to select the top 5 features.

Here are the observations for each of the three models on which we implemented this enhancement:

K-Nearest Neighbors

The k-Nearest Neighbors (k-NN) model underwent a substantial increase in performance. Initially, we achieved a best validation accuracy of 0.71 with k = 11. This was a reasonable performance, but we were keen on exploring how much more we could extract from our model.

In the second phase of our implementation, we integrated feature selection into our model. This step dramatically improved the model's accuracy, with the best validation accuracy jumping to an impressive 0.93 when k = 1. This dramatic increase in validation accuracy underlines the effectiveness of feature selection. By reducing the number of features to consider, the k-NN model was less prone to noise in the data, resulting in a model that could generalize better to unseen data. Here are the final results:

```
Test accuracy for best k = None: 0.9300
Test accuracy for k = 1: 0.9300
Test accuracy for k = 3: 0.7950
Test accuracy for k = 5: 0.7200
Test accuracy for k = 7: 0.7650
Test accuracy for k = 9: 0.7350
Test accuracy for k = 11: 0.7250
Test accuracy for k = 13: 0.7150
Test accuracy for k = 15: 0.7100
Test accuracy for k = 17: 0.7100
Best validation accuracy ( 0.93 ) is achieved with k = 1
```

Logistic Regression

The base Logistic Regression model already provided a good starting point, achieving an accuracy of 0.76 with 'model__C': 10.0 identified as the best hyperparameters. The initial confusion matrix revealed 24 True Negatives, 128 True Positives, 35 False Positives, and 13 False Negatives.

After introducing feature selection into the logistic regression pipeline, the overall accuracy remained at 0.76, but with a shift in the best hyperparameters to 'model__C': 1.0. Though the overall accuracy remained the same, the confusion matrix showed an improvement in the model's ability to correctly identify positive cases. True Positives increased to 130, and False Negatives decreased to 11. Despite the apparent stability in overall accuracy, this change in performance indicates a more balanced model that performs better in individual classes.

Decision Tree

Finally, the Decision Tree model also demonstrated a performance increase thanks to feature selection. The base model achieved a test accuracy of 0.7050 with 'max_depth': 20, 'min_samples_split': 10 as the best hyperparameters. After implementing feature selection, the test accuracy improved to 0.7500, and the best hyperparameters were 'max_depth': 10, 'min_samples_split': 10. The higher accuracy score and the change in hyperparameters indicate both an improvement in performance and a reduction in the model's complexity, suggesting reduced overfitting.Here are the final results:

```
Accuracy: 0.76

Confusion Matrix:
[[ 22  37]
 [ 11 130]]

Best Hyperparameters: {'model__C': 1.0}

True Positives: 130
True Negatives: 22
False Positives: 37
False Negatives: 11
```

In conclusion, the enhancement work we implemented in the bonus section played a substantial role in boosting the performance of our models. This work underscores the importance of feature selection in machine learning pipelines, demonstrating how this relatively straightforward process can yield significant performance boosts. For future work, we could consider testing different feature selection techniques, or explore other machine learning enhancements like ensemble methods, dimensionality reduction techniques, or more advanced feature engineering strategies, to achieve even better results.

7. Conclusion

In this project, we have demonstrated the application of three different machine learning algorithms —K-Nearest Neighbors (KNN), Logistic Regression, and Decision Trees— to assess credit risk based on the German Credit Dataset. Each of these algorithms, albeit through different mechanisms, effectively leverages the available data to categorize loan applicants as good or bad credit risks.

Our initial implementation of the KNN algorithm produced a reasonable validation accuracy, and after careful selection of the number of nearest neighbors (k), we managed to achieve better performance. Similarly, the Logistic Regression model, equipped with a comprehensive data preprocessing pipeline and hyperparameter tuning, was able to deliver a good prediction accuracy. While we do not have the full details of the Decision Tree algorithm implementation at this stage, our preliminary results indicate that it has performed admirably, as well.

Significantly, we observed substantial performance improvements across all algorithms upon the implementation of feature selection. This method helped to select the most impactful features, reducing the noise in the data and improving model generalizability. This resulted in increased validation accuracy for the KNN model, a more balanced model for Logistic Regression, and an improved test accuracy for the Decision Tree model.

Despite the progress made, we recognize that our models are not infallible, and there is still room for improvement. Some false positives and false negatives were observed in our analyses, which could have profound implications in real-world scenarios. Therefore, ongoing refinement of our models remains a priority.

Looking forward, we anticipate expanding on this work by incorporating additional model enhancement techniques. This may include the application of different feature selection methods, ensemble models, dimensionality reduction strategies, or more advanced feature engineering approaches. We believe these techniques can lead to further improvements in our models' predictive performance, ultimately contributing to more reliable and effective credit risk assessments.

In conclusion, our project demonstrates the power and potential of machine learning in the field of credit risk assessment. By carefully selecting, tuning, and enhancing our models, we can extract meaningful insights from our data and make more accurate predictions, thereby facilitating more informed and reliable credit risk assessments. This project emphasizes the importance of continuous refinement and exploration of new methods to further improve the accuracy and reliability of such risk assessments.