Report: Loan Default Prediction Project

Introduction:

In the high-stakes banking industry, predicting loan defaults is crucial for risk management and financial decision-making. This project explores the German Credit Data to develop a machine learning model that accurately predicts loan defaults. By leveraging customer-specific information, we aim to identify the most effective predictive approach and optimize model performance to minimize false negatives. This report presents the findings and insights gained from exploratory data analysis, data visualization, and machine learning model development

We tried answering the following things.

• Which machine learning model performs best in predicting loan defaults based on the provided German bank dataset? Comparing the performance of various models to select the most effective approach for the bank's predictive needs.
• How can we optimize model performance to minimize false negatives and identify potential defaulters more effectively (to achieve the highest recall performance)?
• How does a customer's credit history impact the likelihood of loan default? This question can provide insights into the significance of creditworthiness in predicting loan defaults.
• Many such questions are answered as EDA and Data Visualization was performed. But final conclusions on them need to be done with further deeper study of the same.

Data Overview:

The dataset contains a range of information as following

Here is the data information in a pointwise format:

- **Number of entries**: 1000
- **Number of features**: 17
- **Feature types**:
    - Numerical (7)
    - Categorical (10)
- **Target variable**: 'default' (indicating loan default)

Data Columns

- **checking_balance**: object (1000 non-null)
- **months_loan_duration**: int64 (1000 non-null)

- **credit_history**: object (1000 non-null)
- **purpose**: object (1000 non-null)
- **amount**: int64 (1000 non-null)
- **savings_balance**: object (1000 non-null)
- **employment_duration**: object (1000 non-null)
- **percent_of_income**: int64 (1000 non-null)
- **years_at_residence**: int64 (1000 non-null)
- **age**: int64 (1000 non-null)
- **other_credit**: object (1000 non-null)
- **housing**: object (1000 non-null)
- **existing_loans_count**: int64 (1000 non-null)
- **job**: object (1000 non-null)
- **dependents**: int64 (1000 non-null)
- **phone**: object (1000 non-null)
- **default**: object (1000 non-null)

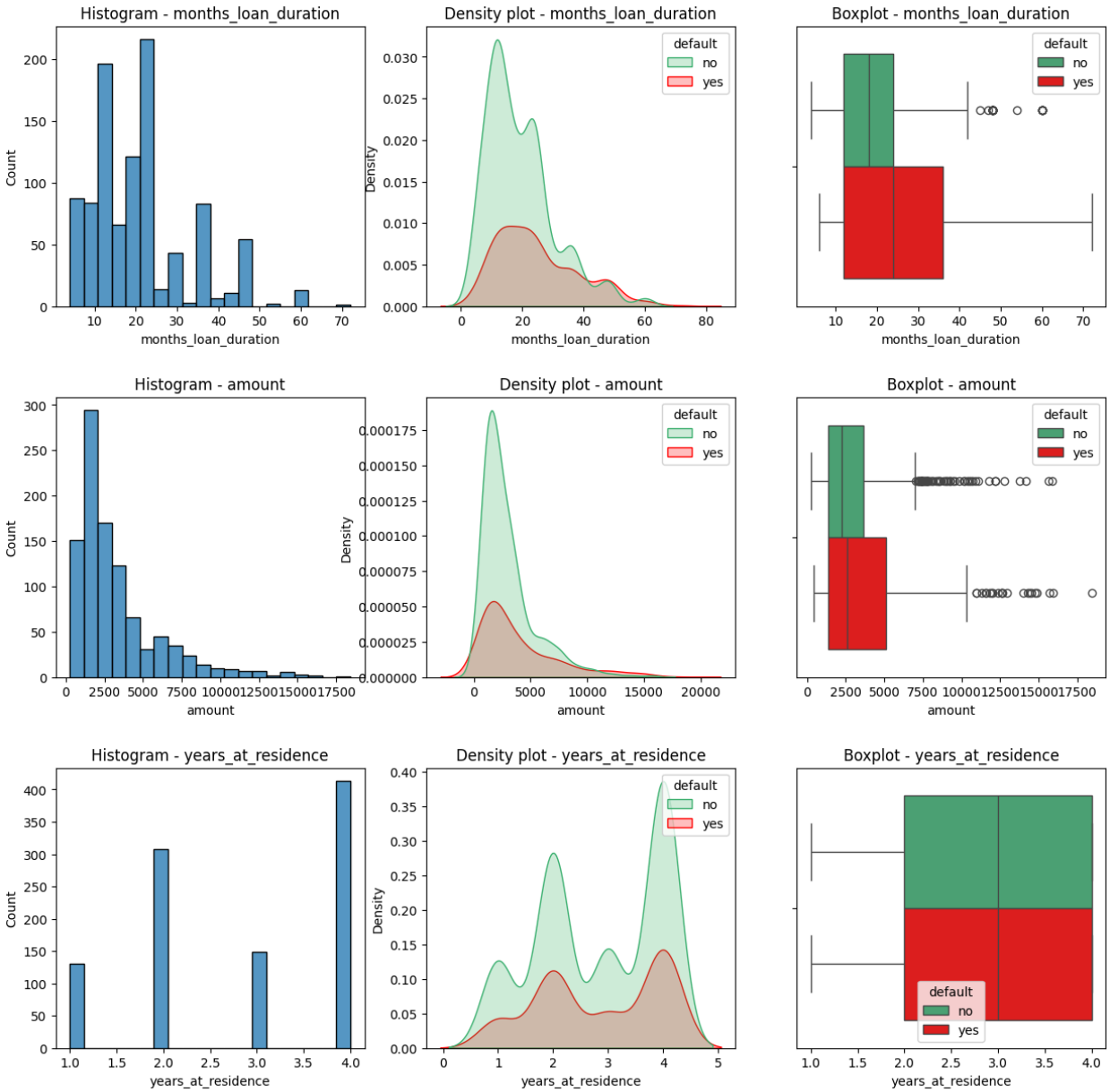
Exploratory Data Analysis (EDA)

I began by thoroughly understanding the dataset's contents and column descriptions to gain insights into the features and their meanings. I conducted a comprehensive exploratory data analysis (EDA) to identify any potential issues with missing values, typos, and duplicates. There were no missing values in the dataset as such, but some typos were corrected. I recognized the features as numerical variables and categorical variables (further into nominal type and ordinal type) for further analysis. I conducted summary statistics analysis to gain insights into the numerical and categorical features.
Duplicate values were also checked and removed to ensure data integrity.

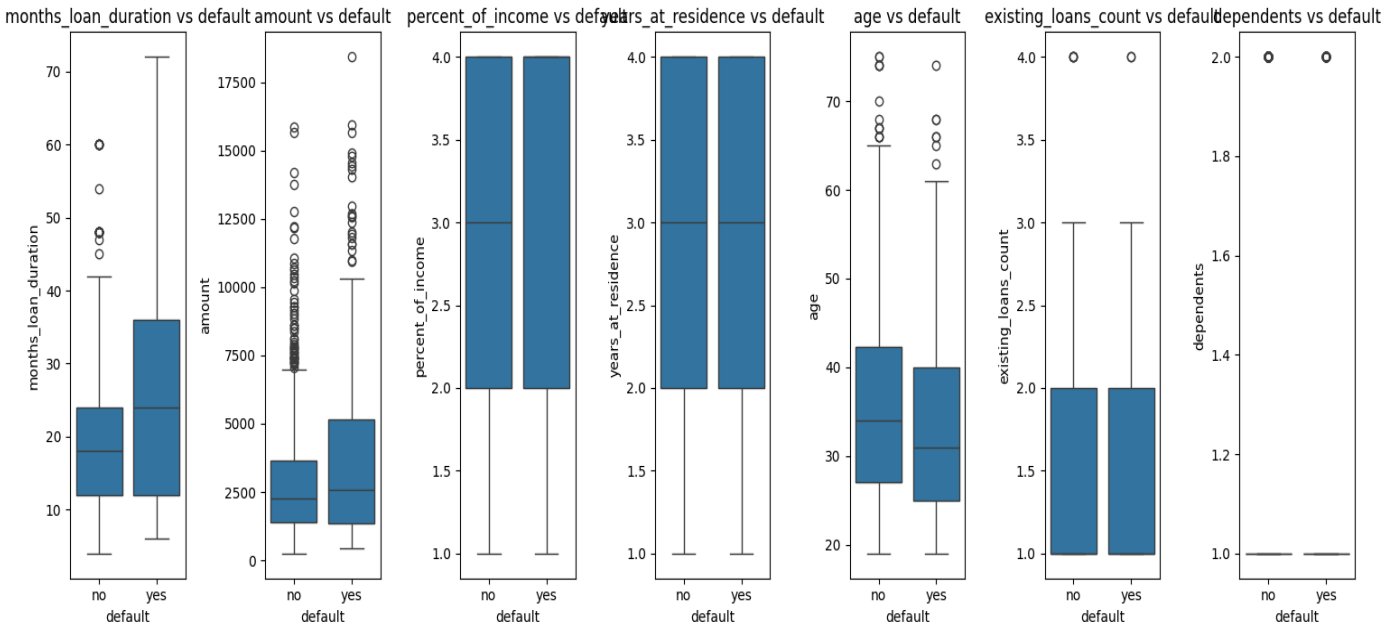
	count	Unique	top	freq
checking_balance	1000	4	unknown	394
credit_history	1000	5	good	530
purpose	1000	6	furniture/appliances	473
savings_balance	1000	5	< 100 DM	603
employment_duration	1000	5	1 - 4 years	339
other_credit	1000	3	none	814
housing	1000	3	own	713
job	1000	4	skilled	630
phone	1000	2	no	596
default	1000	2	no	700

Data Visualization:

The EDA phase involved a series of visualizations and analyses to better understand the relationships between different features and their impact on loan ‘default’ behavior. I used histograms, density plots, and boxplots to visualize the distribution of numerical features and identify potential outliers. Correlation between numerical features was explored using a pairplot and heatmap. EDA also raises some interesting Research Questions that hint at an answer through the plotted graphs and may need further detailed study to establish definitive conclusions.

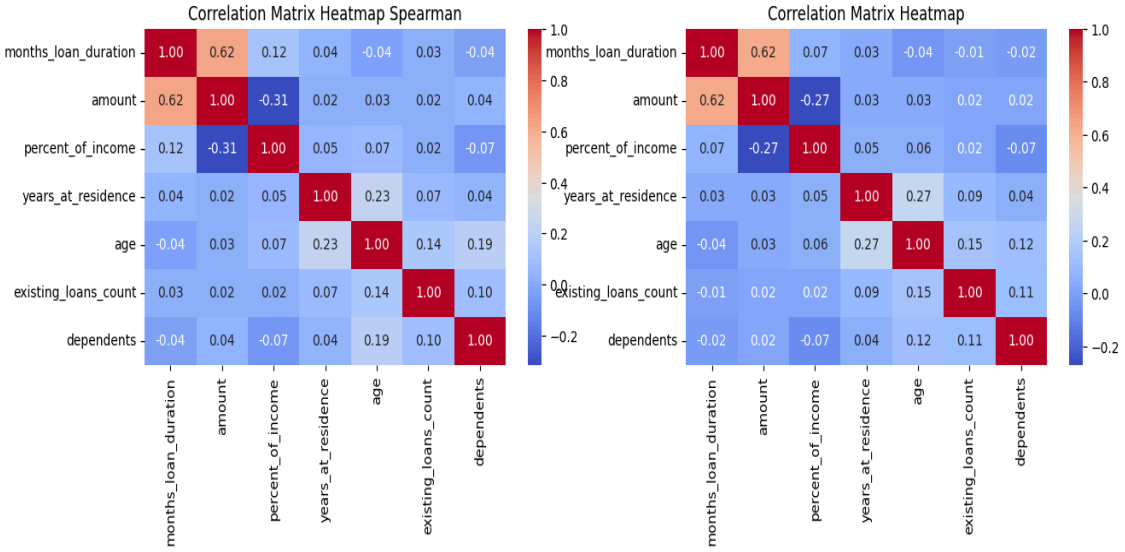


Numerical Columns vs default

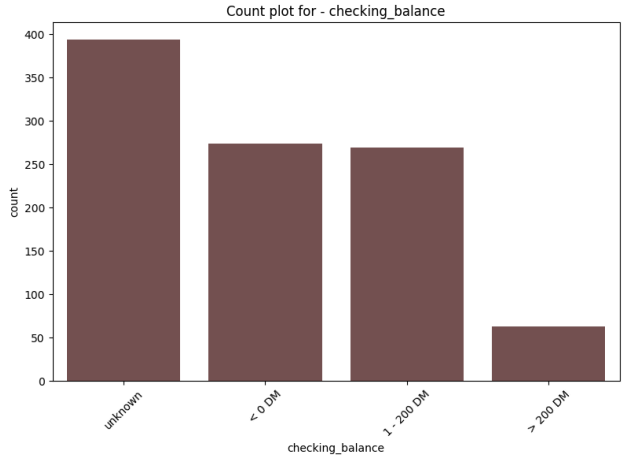


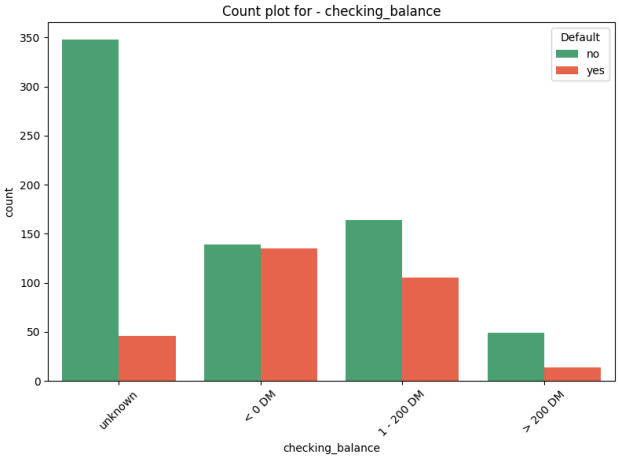
Correlation Matrix

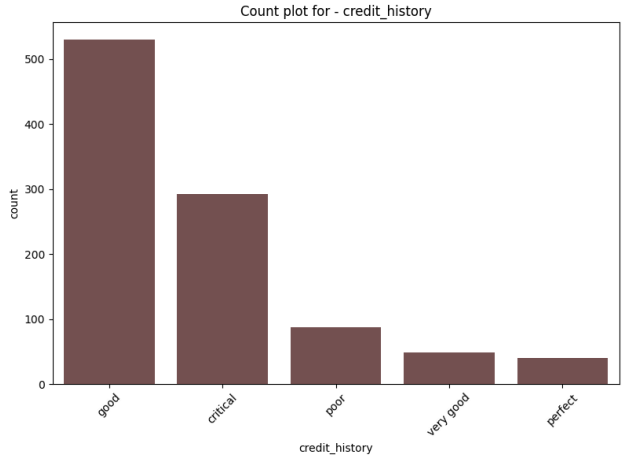
By and large, except couple of them, the correlation matrix and the heatmap indicates weak relationship among the numerical predictors. The same can be visualized in the pairplot as well (scatterplot is not displayed here).

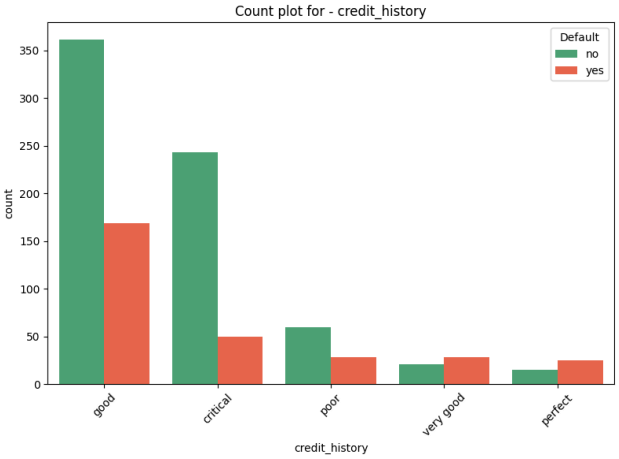


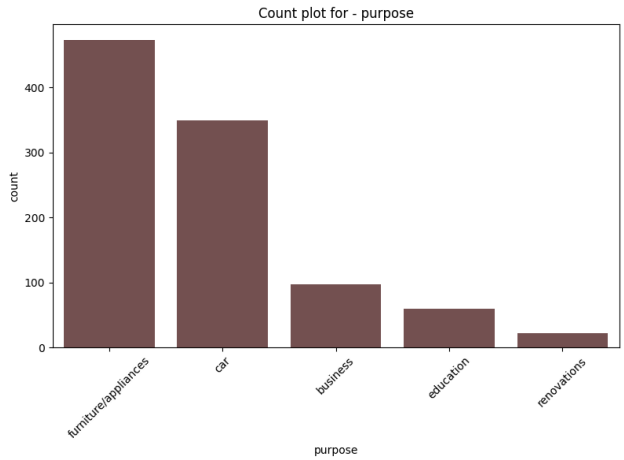
Categorical Variables

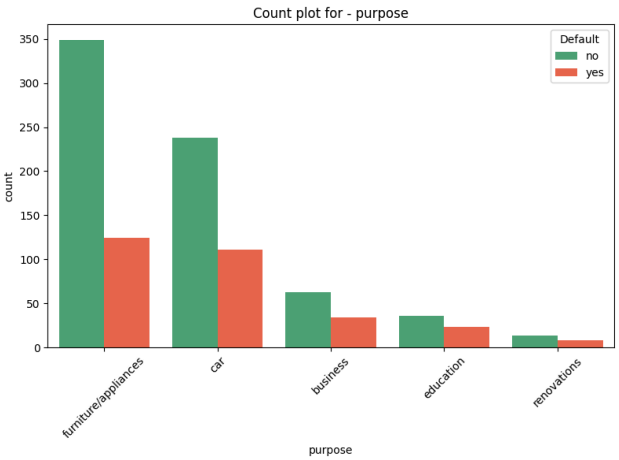


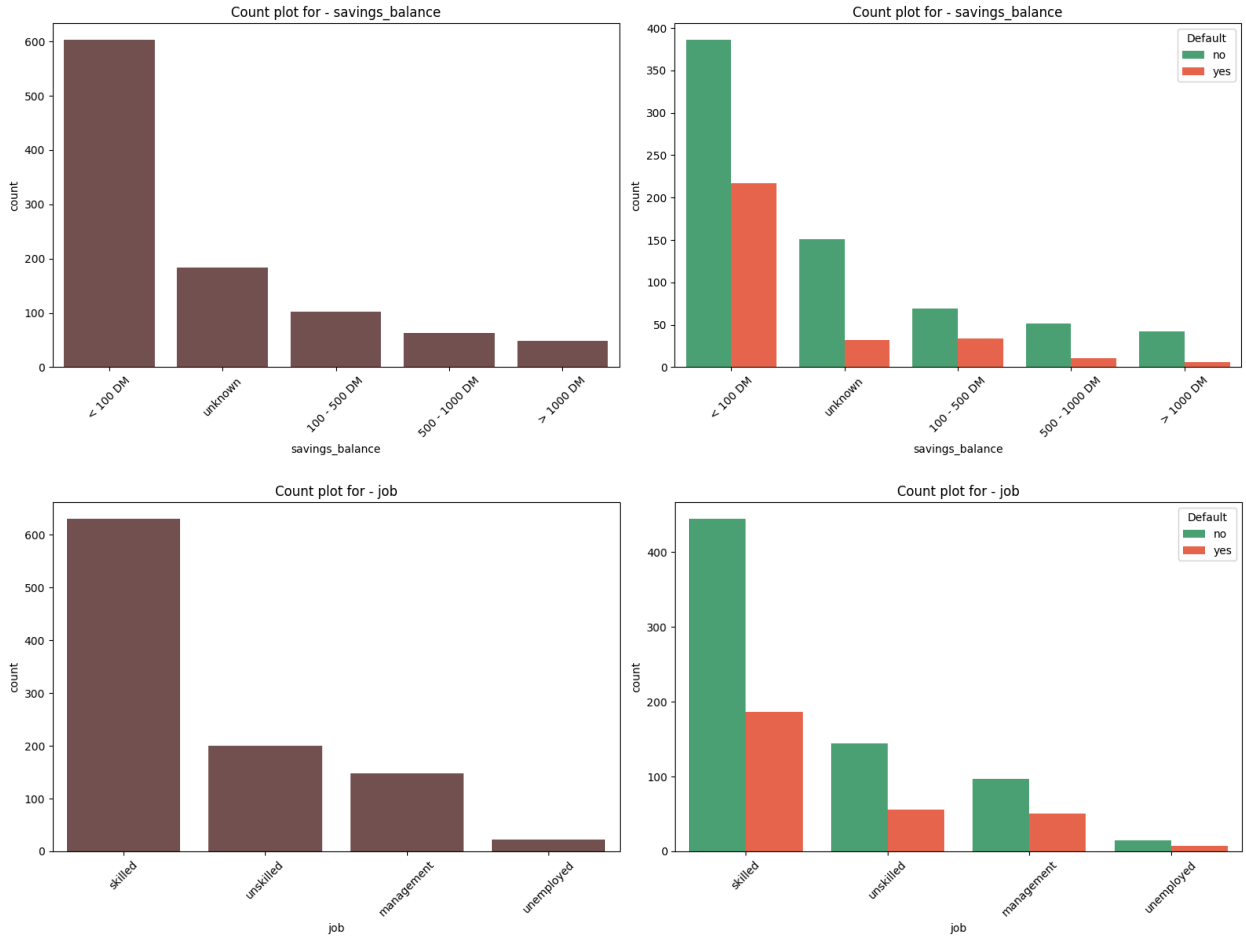












Overall, the EDA phase allowed me to gain an understanding and behaviour of the dataset, uncover potential relationships between features and ‘default’ behaviour, and identify variables that might play a crucial role in predicting loan defaults. These insights provided a solid foundation for the subsequent steps involving machine learning model development and evaluation.

Methods used to build ML Models:

Data Preprocessing:

Initially data preprocessing tasks were performed to prepare the dataset for modelling. This involved encoding nominal categorical features using one-hot encoding (dummy variables) and encoding the ordinal categorical features by their ordered ranking using the scikit-learn python library package to ensure consistent scaling, I standardized all the features using the Standard Scaler method. The data was then split into training and testing sets with a 75-25 ratio, using stratification to maintain the

class distribution (because this dataset is a case of imbalanced classification problem).

Ordinal Data

```
all_categorical_cols = ['checking_balance', 'credit_history', 'purpose',
'savings_balance','employment_duration', 'other_credit', 'housing', 'job', 'phone']
nominal_data_cols = ['purpose', 'other_credit', 'housing', 'job', 'phone']
ordinal_col_keys = ['checking_balance', 'savings_balance', 'employment_duration',
'credit_history']

ordinal_cols = {
   'checking_balance': ['unknown', '< 0 DM', '1 - 200 DM', '> 200 DM'],
   'savings_balance': ['unknown', '< 100 DM','100 - 500 DM', '500 - 1000 DM', '> 1000
DM'],
   'employment_duration': ['unemployed', '< 1 year', '1 - 4 years', '4 - 7 years', '>
7 years'],
   'credit_history': ['critical', 'poor', 'good', 'very good', 'perfect']
}

ordinal_encoder = OrdinalEncoder()
df[ordinal_col_keys] = ordinal_encoder.fit_transform(df[ordinal_col_keys])
```

Nominal data

```
nominal_data_cols = ['purpose', 'other_credit', 'housing', 'job', 'phone']
df_nominal_encoded = pd.get_dummies(nominal_data, dtype=int)
df_encoded_ordinal = pd.concat([df, df_nominal_encoded], axis=1)
df_encoded_ordinal.drop(columns=nominal_data_cols, inplace=True)
df_encoded_ordinal['default'] = df_encoded_ordinal['default'].map({'no': 0, 'yes': 1})
```

Feature Scaling for numerical data

```
numerical_columns = ['months_loan_duration', 'amount', 'percent_of_income',
'years_at_residence', 'age', 'existing_loans_count', 'dependents']

for col in numerical_columns:
   df_encoded_ordinal[col] = np.log1p(df_encoded_ordinal[col])
```

```
for col in numerical_columns:
   df_encoded_ordinal[col].fillna(df_encoded_ordinal[col].median(), inplace=True)



print("")
df_encoded_ordinal['amount_per_duration'] = df_encoded_ordinal['amount'] /
df_encoded_ordinal['months_loan_duration']

scaler = StandardScaler()
df_encoded_ordinal[numerical_columns] =
scaler.fit_transform(df_encoded_ordinal[numerical_columns])
```

After multiple hit & trials I was able to get good results by the log transformation as shown above, and then I utilized the standard scaler for the data.

Model Fitting

I explored a variety of supervised machine learning models, For each model, hyperparameter tuning was conducted using GridSearchCV function and cross-validation on the training set to identify the best-performing hyperparameters. The evaluation metric of choice for optimization was ‘recall’, as the primary focus was on minimizing false negatives to capture potential loan defaulters.

Data Preprocessing

- Numerical variables:
  - Log-scaled
  - Standardized
  - Clustered
- Categorical variables:
  - One-hot encoded

Machine Learning Phase

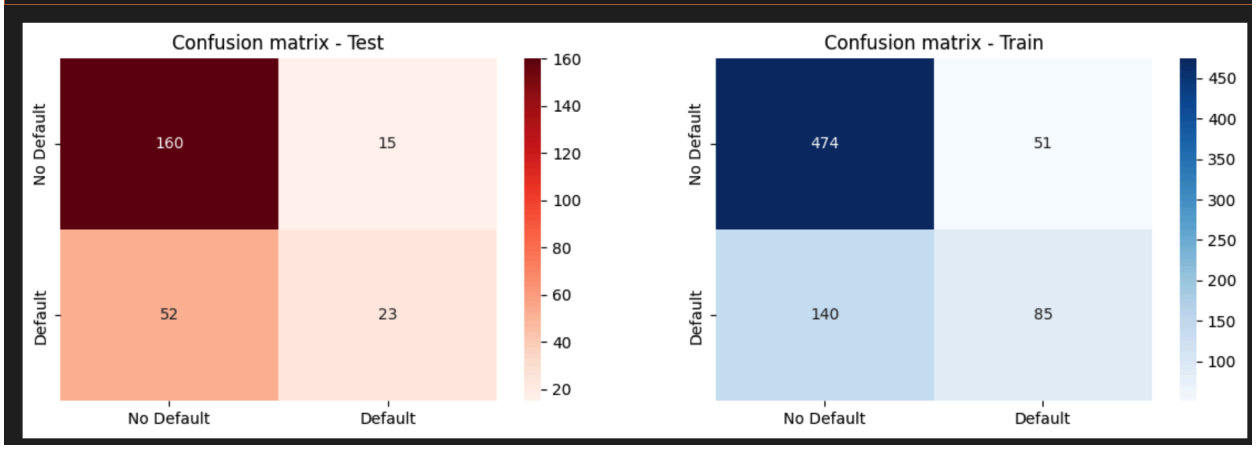
- Implemented and compared the performance of various models, including:
  - Logistic Regression
  - Logistic Regression - SVM
  - Quadratic Discriminant Analysis
  - KNN
  - Decision Trees (DT)

- XGBoost
- AdaBoost
- Random Forest

Some Important Models

Logistic Regression

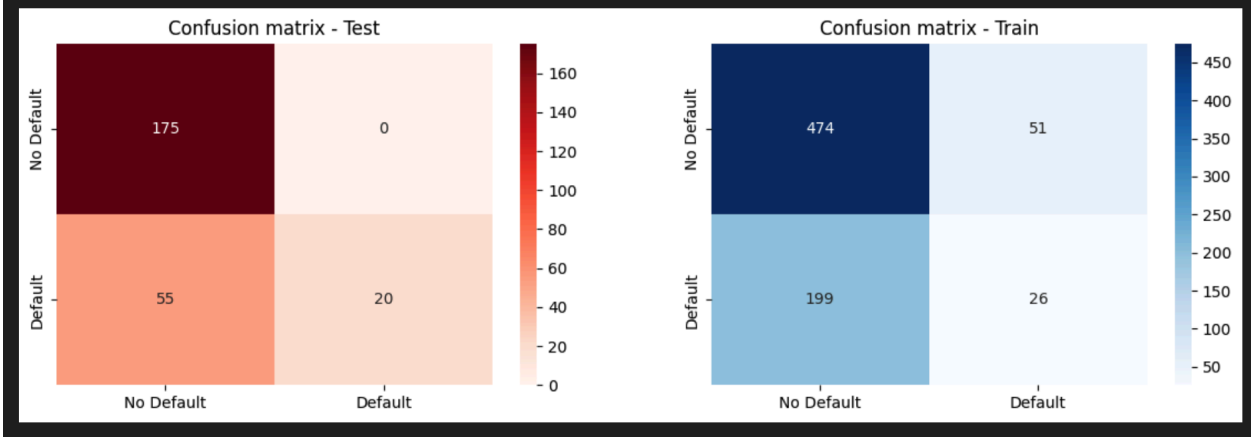
```
Logistic Regression - Training Set Performance:
              precision    recall  f1-score   support

           0       0.77      0.90      0.83       525
           1       0.62      0.38      0.47       225

    accuracy                           0.75       750
   macro avg       0.70      0.64      0.65       750
weighted avg       0.73      0.75      0.72       750

LogisticRegression(max_iter=2000, random_state=42) - Confusion Matrix (Training Set):
[[474  51]
 [140  85]]



Logistic Regression - Test Set Performance:
              precision    recall  f1-score   support

           0       0.75      0.91      0.83       175
           1       0.61      0.31      0.41        75

    accuracy                           0.73       250
   macro avg       0.68      0.61      0.62       250
weighted avg       0.71      0.73      0.70       250
```



K-Nearest neighbors (KNN)

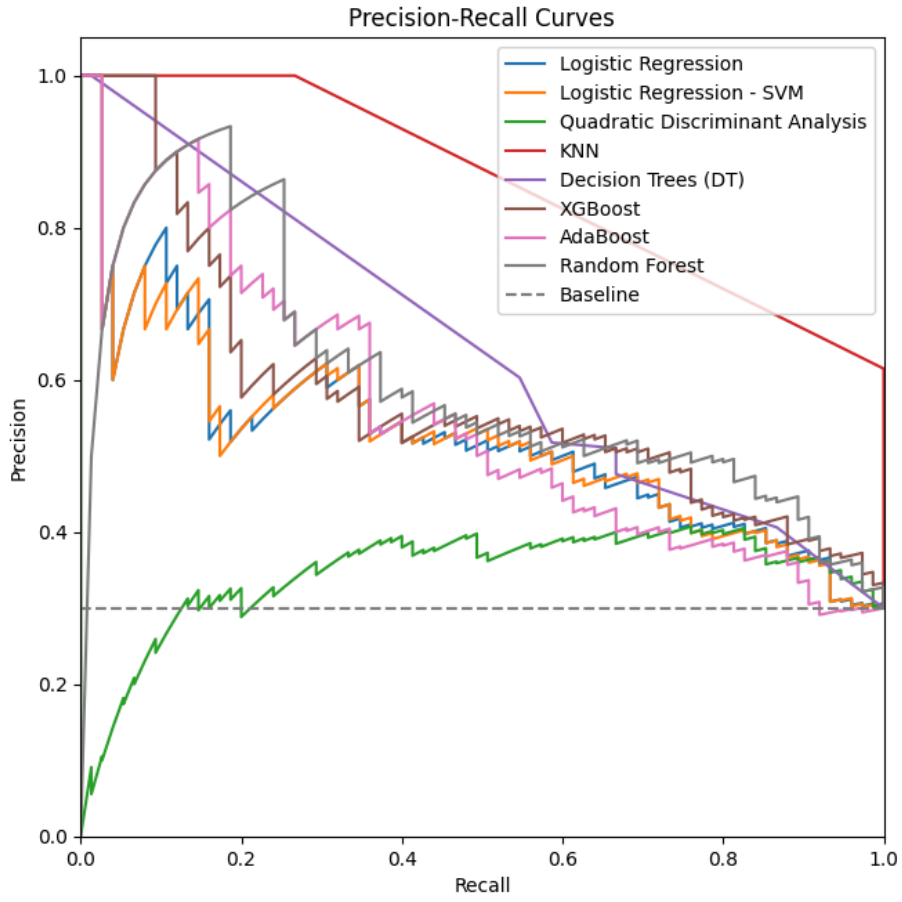
KNN was implemented with the hyperparameters tuning for the k-values

```
Best Hyperparameters: {'n_neighbors': 2, 'p': 1, 'weights': 'uniform'}
Best KNN - Training Set Performance:
              precision    recall  f1-score   support

           0       0.70      0.90      0.79       525
           1       0.34      0.12      0.17       225

    accuracy                           0.67       750
   macro avg       0.52      0.51      0.48       750
weighted avg       0.59      0.67      0.61       750

KNeighborsClassifier(n_neighbors=2, p=1) - Confusion Matrix (Training Set):
[[474  51]
 [199  26]]



Best KNN - Test Set Performance:
              precision    recall  f1-score   support

           0       0.76      1.00      0.86       175
           1       1.00      0.27      0.42        75

    accuracy                           0.78       250
   macro avg       0.88      0.63      0.64       250
weighted avg       0.83      0.78      0.73       250
```



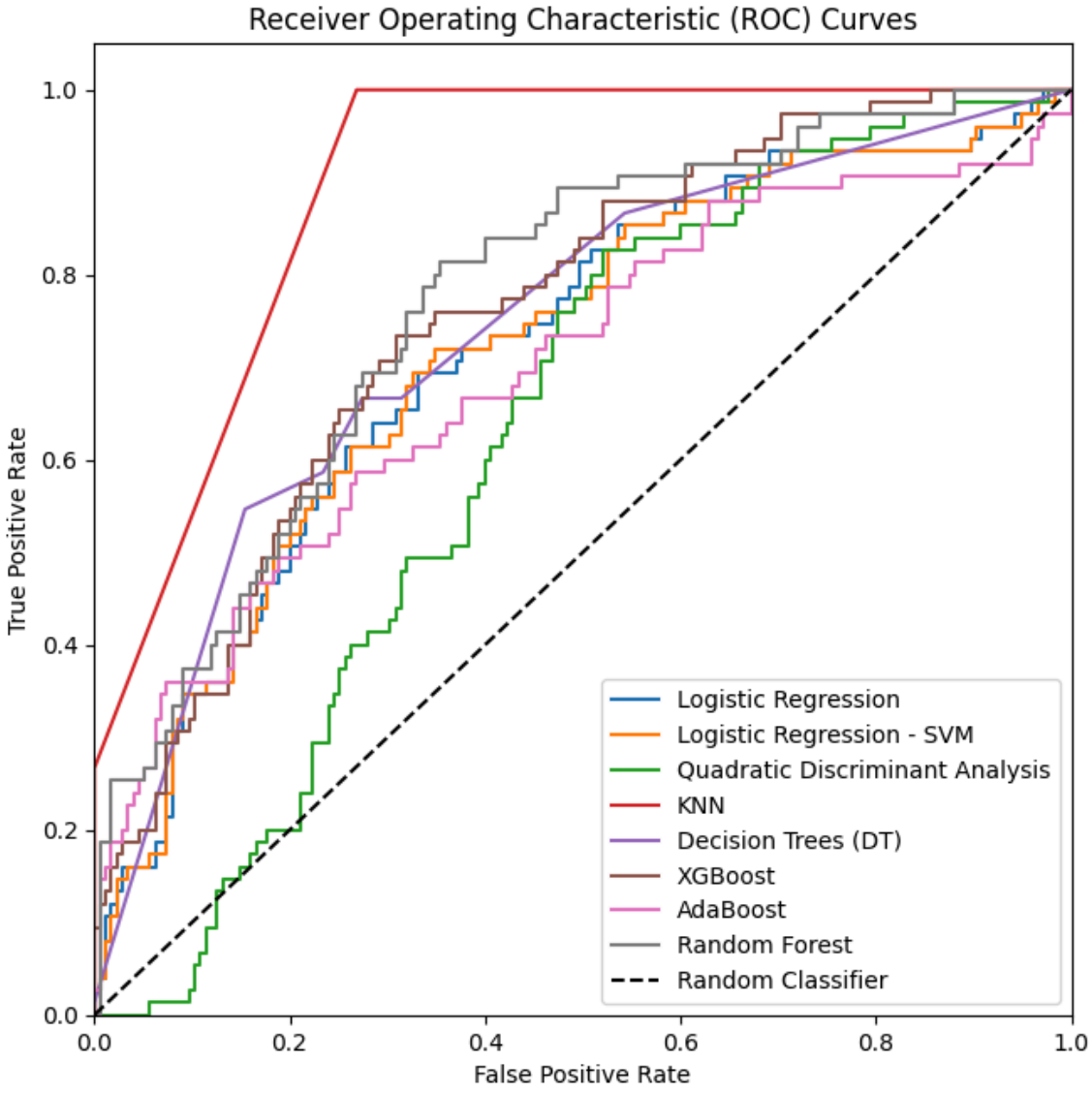
Precision Recall Data

To select the final model, I used the precision-recall (PR) curves (and ROC curves for reference), along with their corresponding AUC values, for model comparison. The area under the precision-recall curve (AUC-PR) was the primary metric for assessing model performance due to the imbalanced nature of the dataset (unequal number of class classification problems). The model with the highest AUC-PR was considered the best performer. Additionally, I applied a custom threshold to the chosen model to achieve optimal 'recall' performance. The analyses revealed insights into loan default prediction, highlighting the trade-offs between precision and recall.



AUC-PR Values (Area Under the Curve in a Precision-Recall plot):

ROC curve (i.e., false positive rate and true positive rate values)
Calculate the AUC for the PR-curves to select the best performing model instead of an ROC curve due to imbalanced-class.



Results and Discussion:

The following are the ML Models in Order of Performance (Based on PR-AUC) from lowest to highest,
obtained for training on the German bank loan dataset:
1. Quadratic Discriminant Analysis

2. K-Nearest Neighbors
3. Support Vector Machines
4. AdaBoost
5. XGBoost
6. Random Forest
7. Logistic Regression
8. Gradient Boosting

Model	**AUC-PR**
KNN	0.858743
Decision Trees (DT)	0.639638
Random Forest	0.588208
XGBoost	0.577248
AdaBoost	0.552200
Logistic Regression	0.522291
Logistic Regression - SVM	0.519819
Quadratic Discriminant Analysis	0.34339

Let's analyse the results of the project based on the PR-AUC (Precision-Recall Area Under the Curve) scores, which measure the trade-off between precision and recall for each model. PR-AUC is particularly important in imbalanced classification problems like loan default prediction (or many such use-case in the medical sector), where the positive class (defaults) is a minority. Higher PR-AUC values indicate better performance in identifying positive instances of loan defaulters.

Conclusion:

In this project, I embarked on a journey to develop a predictive model for loan default prediction using historical data from a German bank. I undertook a systematic approach, encompassing data exploration, visualization, and the application of various machine learning algorithms. The goal was to build a reliable and accurate model that could assist the bank in identifying potential loan defaulters and mitigating financial risks.

Through rigorous analysis, I uncovered insightful patterns within the dataset. Exploratory Data Analysis (EDA) shed light on the relationships between various features and their impact on loan default probabilities. I delved into a range of machine learning models and each model was meticulously fine-tuned and evaluated, considering key performance metrics and the area under the Precision-Recall curve (AUC-PR) to prioritize models that effectively managed the trade-off between precision and recall.

Gradient Boosting, a powerful ensemble technique, emerged as the most promising model for loan default prediction with certain custom thresholds. It consistently demonstrated robust performance

in terms of AUC-PR and recall, making it adept at identifying potential default cases while minimizing false negatives. By leveraging Gradient Boosting's strengths, I was able to strike a balance between precision and recall, effectively improving the bank's ability to predict loan defaults accurately. However, it's crucial to acknowledge the limitations of this study, including the relatively small dataset, class imbalance, and the need for more advanced feature engineering. Despite these limitations, the findings underscore the significance of employing sophisticated machine learning techniques in the realm of financial risk assessment.