Report On

PREDICTING CREDIT RISK FOR LOAN APPLICANTS

Done by

Dona Jince
DT20234893456

1. INTRODUCTION

Credit risk assessment is a fundamental process for financial institutions, serving as a cornerstone for informed lending decisions and financial stability. Accurately predicting the likelihood of loan default is crucial for mitigating potential losses and ensuring responsible lending practices. Traditional methods often rely on manual reviews, limited data points, and subjective judgments, which can lead to inefficiencies and biases. In recent years, the emergence of machine learning has revolutionized various industries, offering powerful tools for data analysis and prediction. This project leverages the potential of machine learning to develop a robust and accurate credit risk prediction model, addressing the limitations of traditional approaches and enhancing the overall efficiency and effectiveness of credit risk assessment.

The German Credit Risk dataset, a publicly available dataset containing information about loan applicants and their creditworthiness, serves as the foundation for this project. This dataset provides a rich source of information, including demographic factors, financial history, and credit behavior, which are crucial for building a comprehensive credit risk model. By applying various machine learning techniques, this project aims to extract meaningful insights from this data and develop a model that can accurately predict the likelihood of loan default.

The project involves a systematic approach encompassing data exploration, preprocessing, model development, and evaluation. Data exploration helps in understanding the underlying patterns and relationships within the dataset, providing valuable insights for feature engineering and model selection. Data preprocessing steps address issues such as missing values, outliers, and data imbalance, ensuring the quality and reliability of the data used for model training. Model development focuses on selecting an appropriate machine learning algorithm, such as a Random Forest classifier, which is known for its robustness and ability to handle complex datasets. Finally, model evaluation involves rigorously assessing the performance of the developed model using metrics such as accuracy, precision, recall, and F1-score. This evaluation ensures that the model meets the desired performance criteria and is suitable for practical application.

The development of an accurate credit risk prediction model has significant implications for financial institutions and borrowers alike. For financial institutions, this model can automate the credit risk assessment process, reducing the time and effort required for manual reviews. This automation leads to faster loan approval decisions, improved operational efficiency, and reduced costs. Moreover, the model's ability to identify high-risk applicants can help mitigate potential financial losses due to loan defaults, enhancing the overall financial stability of the institution. For borrowers, this model can provide a more objective and transparent credit risk assessment, ensuring fair and unbiased lending practices. By streamlining the lending process, the model can also facilitate access to credit for those who might otherwise be excluded due to limitations in traditional credit scoring methods.

In conclusion, this project presents a comprehensive approach to credit risk prediction using machine learning techniques. By leveraging the German Credit Risk dataset and employing rigorous data analysis and model development methodologies, the project aims to develop a robust and accurate credit risk prediction model.

2. METHODOLOGY

1. Data Collection and Exploration

The German Credit Risk dataset was loaded and explored. Descriptive statistics and visualizations like histograms and box plots were used to understand the data's distribution and identify potential outliers. Missing data patterns were visualized using heatmaps. Categorical features like 'Sex', 'Housing', 'Saving accounts', etc., were analyzed against the 'Risk' variable to understand their relationship.

```
df = pd.read_csv('/content/german_credit_data.csv')
print(df)
```

```
     Unnamed: 0  Age     Sex  Job Housing Saving accounts Checking account  \
0             0   67    male    2     own             NaN           little
1             1   22  female    2     own          little         moderate
2             2   49    male    1     own          little              NaN
3             3   45    male    2    free          little           little
4             4   53    male    2    free          little           little
..          ...  ...     ...  ...     ...             ...              ...
995         995   31  female    1     own          little              NaN
996         996   40    male    3     own          little           little
997         997   38    male    2     own          little              NaN
998         998   23    male    2    free          little           little
999         999   27    male    2     own        moderate         moderate

     Credit amount  Duration              Purpose  Risk
0             1169         6             radio/TV  good
1             5951        48             radio/TV   bad
2             2096        12            education  good
3             7882        42  furniture/equipment  good
4             4870        24                  car   bad
..             ...       ...                  ...   ...
995           1736        12  furniture/equipment  good
996           3857        30                  car  good
997            804        12             radio/TV  good
998           1845        45             radio/TV   bad
999           4576        45                  car  good

[1000 rows x 11 columns]
```

Figure 2.1 Dataset

Figure 2.1 shows the dataset. It has 1000 entries and 11 columns. Target variable is Risk.

```
df['Risk'].value_counts()
```

Risk	count
good	700
bad	300

dtype: int64

Figure 2.2 Target Variable

Figure 2.2 shows the target variable Risk which is categorical.It has two values, good and bad. There are 700 entries for predicting good and 300 for predicting bad. So the dataset is unbalanced.

```
[166] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Unnamed: 0        1000 non-null   int64
 1   Age               1000 non-null   int64
 2   Sex               1000 non-null   object
 3   Job               1000 non-null   int64
 4   Housing           1000 non-null   object
 5   Saving accounts   817 non-null    object
 6   Checking account  606 non-null    object
 7   Credit amount     1000 non-null   int64
 8   Duration          1000 non-null   int64
 9   Purpose           1000 non-null   object
 10  Risk              1000 non-null   object
dtypes: int64(5), object(6)
memory usage: 86.1+ KB
```

Figure 2.3 Dataset Information

Figure 2.3 shows the additional information about dataset such as the type of columns,non-null values count.

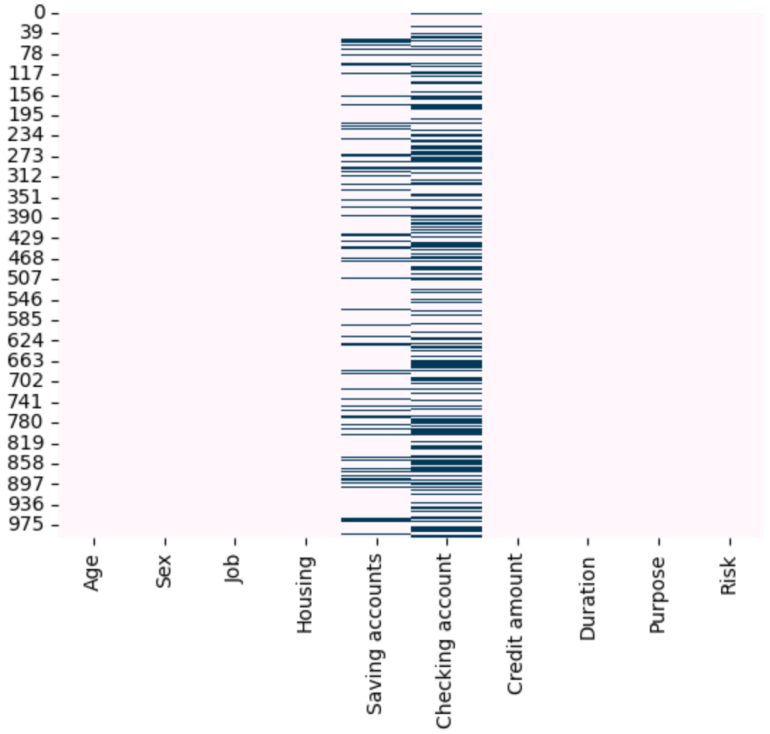

Figure 2.4 Missing Values Visualization

Figure 2.4 shows the missing values pattern in the dataset. Columns Saving accounts and Checking account has missing values.

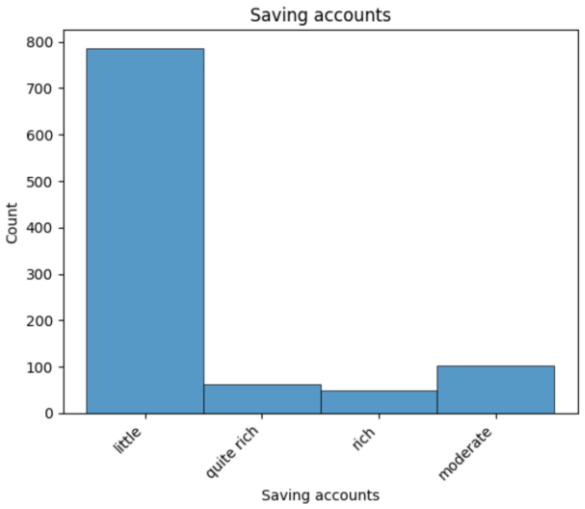

Figure 2.5 Histogram of Saving accounts

Figure 2.5 shows that the majority of customers have "little" savings, indicating a skewed distribution in the Saving accounts feature.

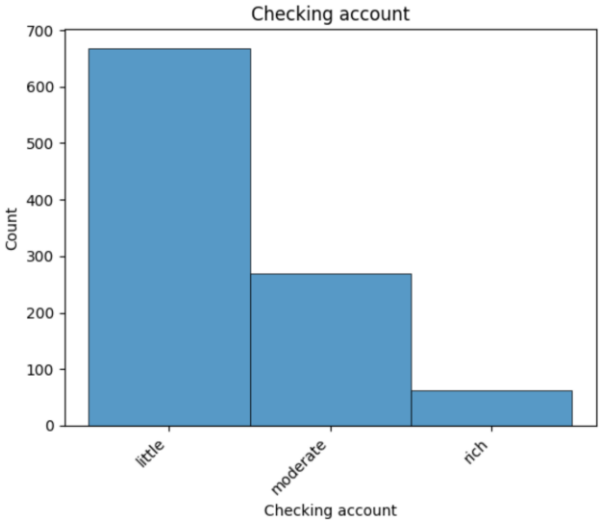

Figure 2.6 Histogram of Checking accounts

Figure 2.6 shows that the majority of customers have lower checking account balances.

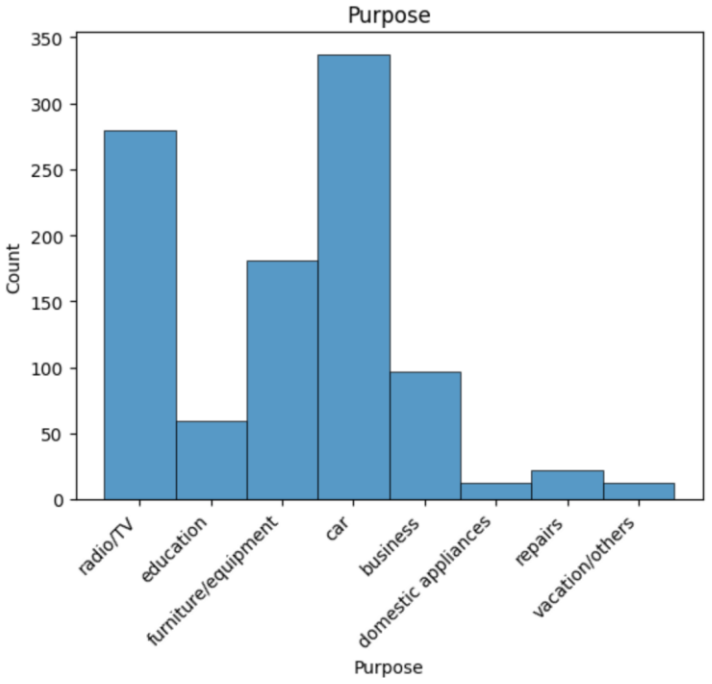

Figure 2.7 Histogram of Purpose

Figure 2.7 shows that the majority of customers request loan for buying cars.

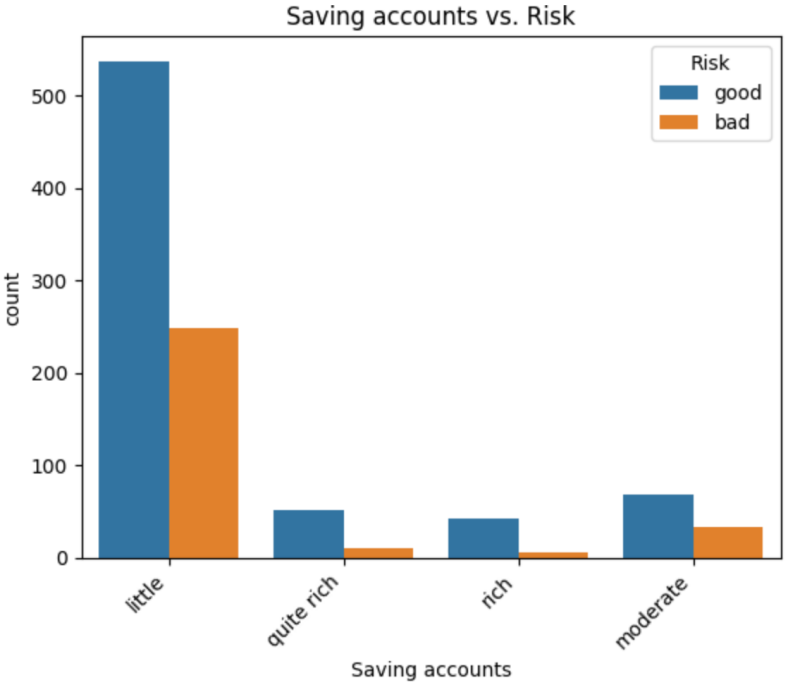

Figure 2.8 count plot showing Saving accounts vs. Risk

Figure 2.8 shows that when the savings are super rich, the chances of the risk being good is very high.

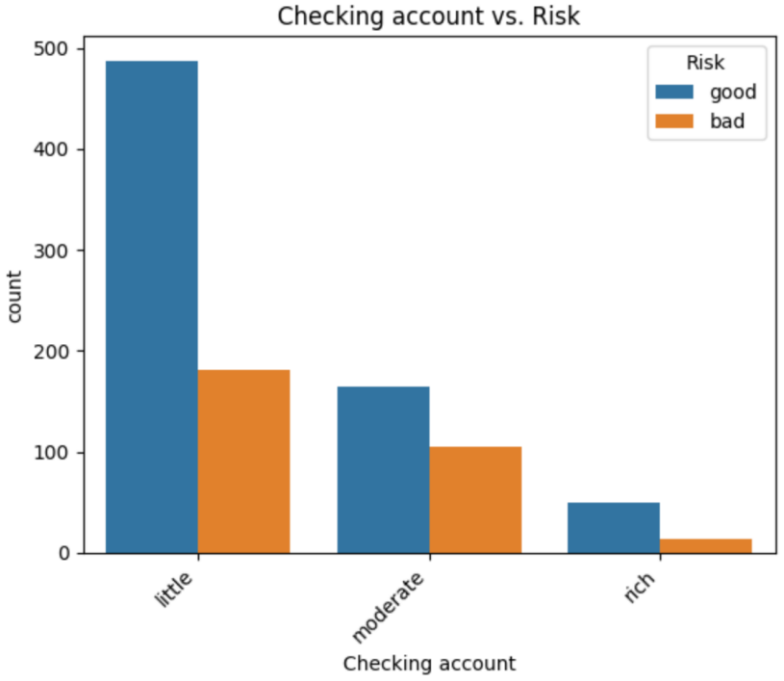

Figure 2.9 count plot showing Checking accounts vs. Risk

Figure 2.9 shows a higher proportion of 'bad risk' among applicants with 'little' or lower checking account balances.

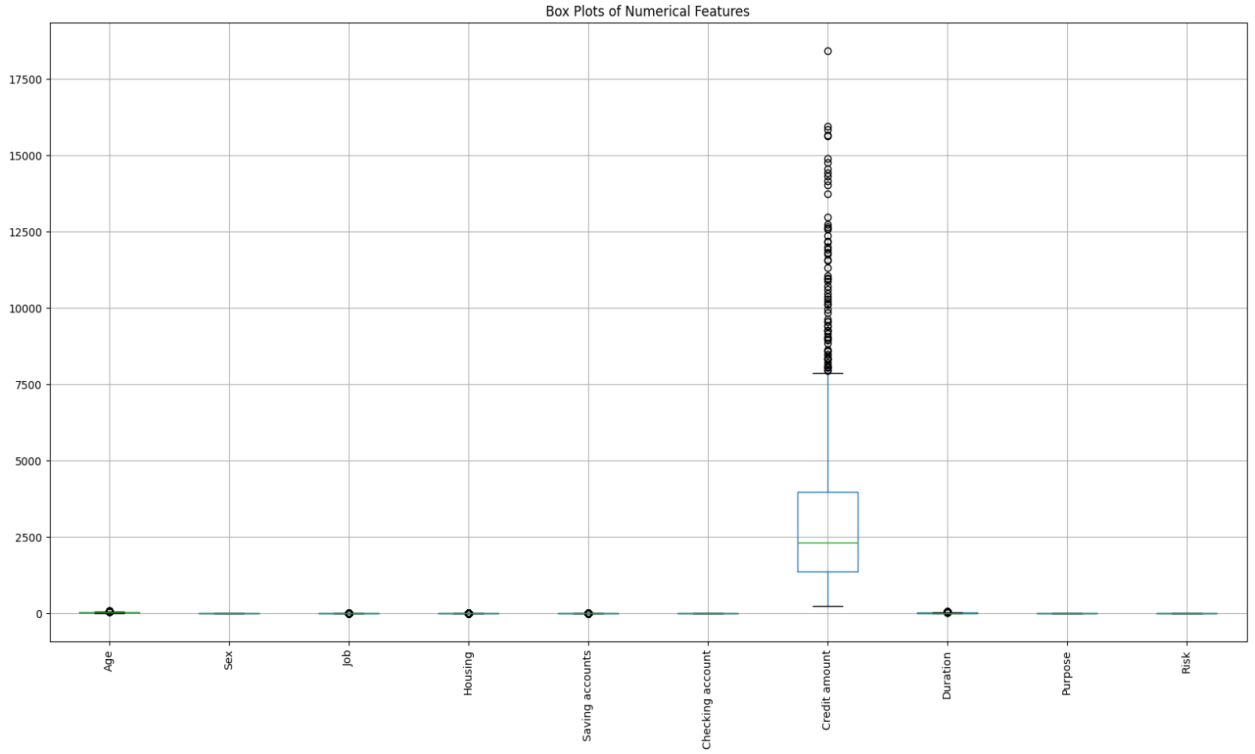

Figure 2.10 Outlier Detection

Figure 2.10 shows that feature credit amount has outliers.

2. Data Preprocessing

Unwanted columns are dropped. Missing categorical values were imputed using the mode. Categorical features were converted into numerical using Label Encoding. Outliers in 'Credit amount' were identified and removed. To balance the dataset and address the class imbalance in the 'Risk' variable, the SMOTE (Synthetic Minority Over-sampling Technique) was applied. This process helps avoid bias in model training.

```
[167] df=df.drop(columns=['Unnamed: 0'])
```

Figure 2.11 Code snippet for dropping unwanted column

Figure 2.11 shows the Code snippet for removing unwanted column ‘Unnamed: 0’.

	missing_values	percent_missing %	data type
Age	0	0.0	int64
Sex	0	0.0	object
Job	0	0.0	int64
Housing	0	0.0	object
Saving accounts	183	18.3	object
Checking account	394	39.4	object
Credit amount	0	0.0	int64
Duration	0	0.0	int64
Purpose	0	0.0	object
Risk	0	0.0	object

Figure 2.12 Count of missing values

Figure 2.12 shows the count of missing values.

```
for column in ['Sex', 'Housing', 'Saving accounts', 'Checking account', 'Purpose', 'Risk']:
    df[column] = df[column].fillna(df[column].mode()[0])
```

Figure 2.13 Code snippet for treating missing values

Figure 2.13 shows the code snippet for treating missing values. It is done by replacing it with the mode of each feature.

```
lencoders = {}
for col in df.select_dtypes(include=['object']).columns:
    lencoders[col] = LabelEncoder()
    df[col] = lencoders[col].fit_transform(df[col])

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 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   int64
 2   Job               1000 non-null   int64
 3   Housing           1000 non-null   int64
 4   Saving accounts   1000 non-null   int64
 5   Checking account  1000 non-null   int64
 6   Credit amount     1000 non-null   int64
 7   Duration          1000 non-null   int64
 8   Purpose           1000 non-null   int64
 9   Risk              1000 non-null   int64
dtypes: int64(10)
memory usage: 78.3 KB
```

Figure 2.14 Code snippet for converting categorical to numerical

Figure 2.14 shows the code snippet for converting categorical to numerical features using labelencoder.

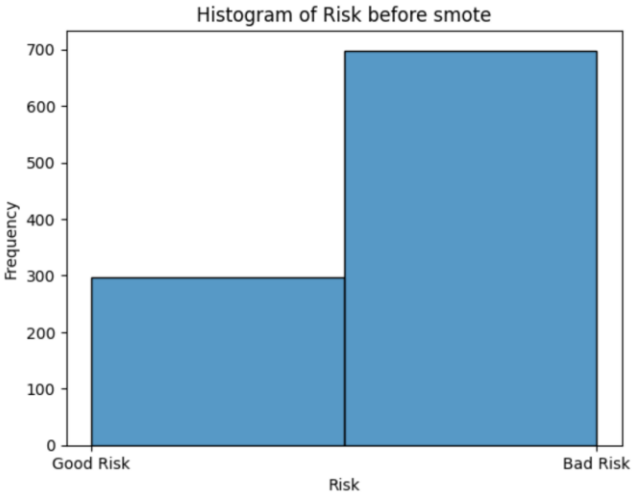

Figure 2.15 Before SMOTE

Figure 2.15 shows the distribution of the feature Risk before SMOTE.

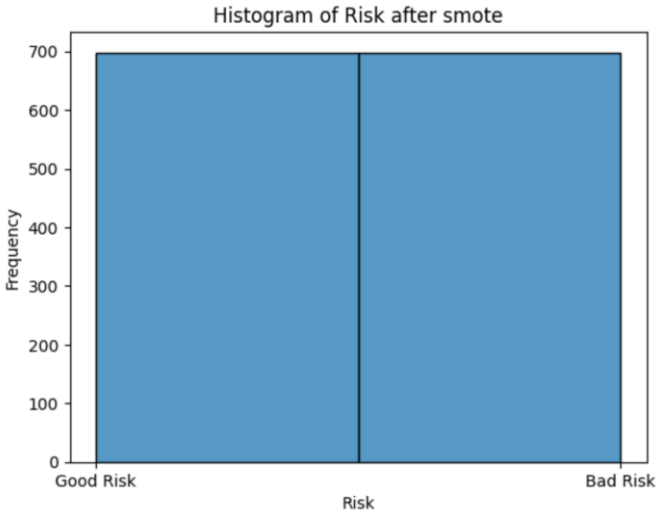

Figure 2.16 After SMOTE

Figure 2.15 shows the distribution of the feature Risk after SMOTE.

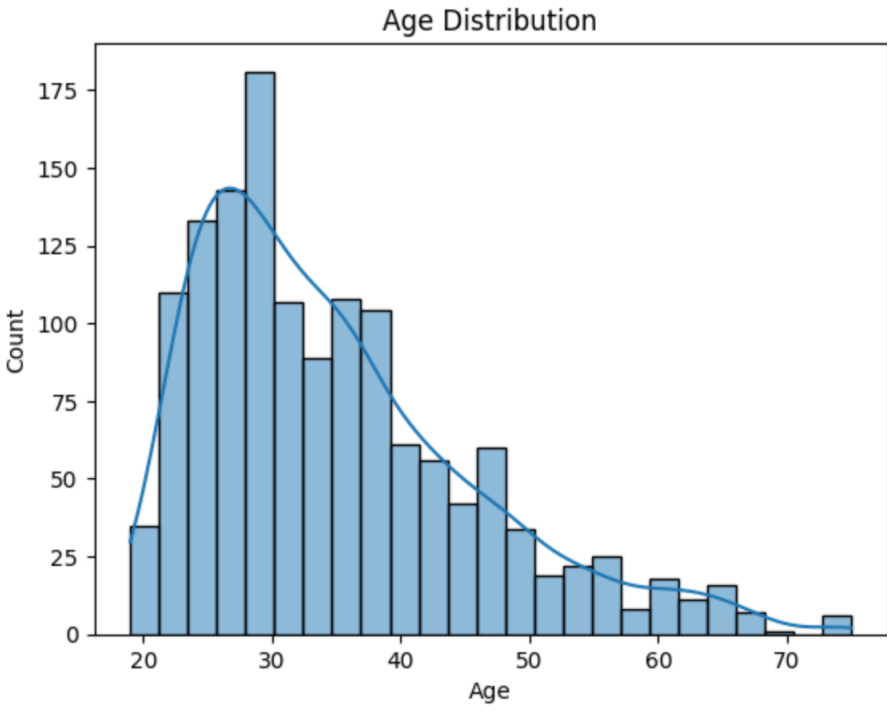

Figure 2.17 Age Distribution

Figure 2.17 shows most individuals are younger, with fewer older individuals and there is a gradual decline in frequency as age increases beyond 30.

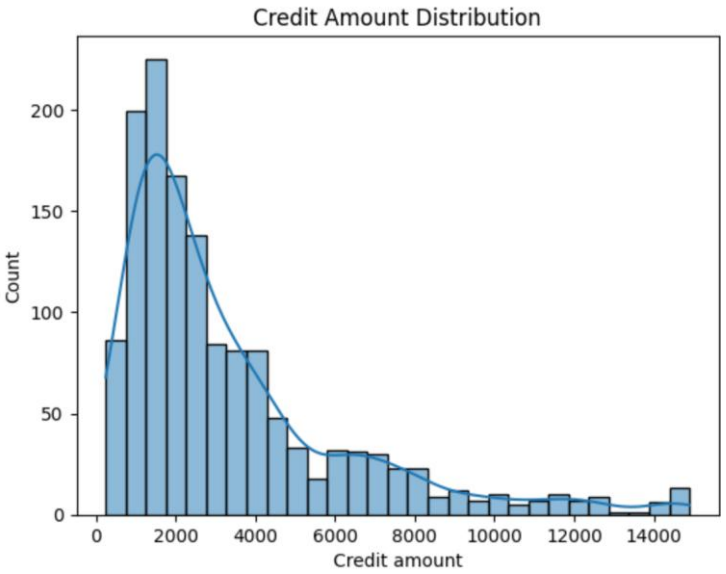

Figure 2.18 Credit Amount Distribution

Figure 2.18 shows that the most frequent credit range is around 1,500–2,000 units.

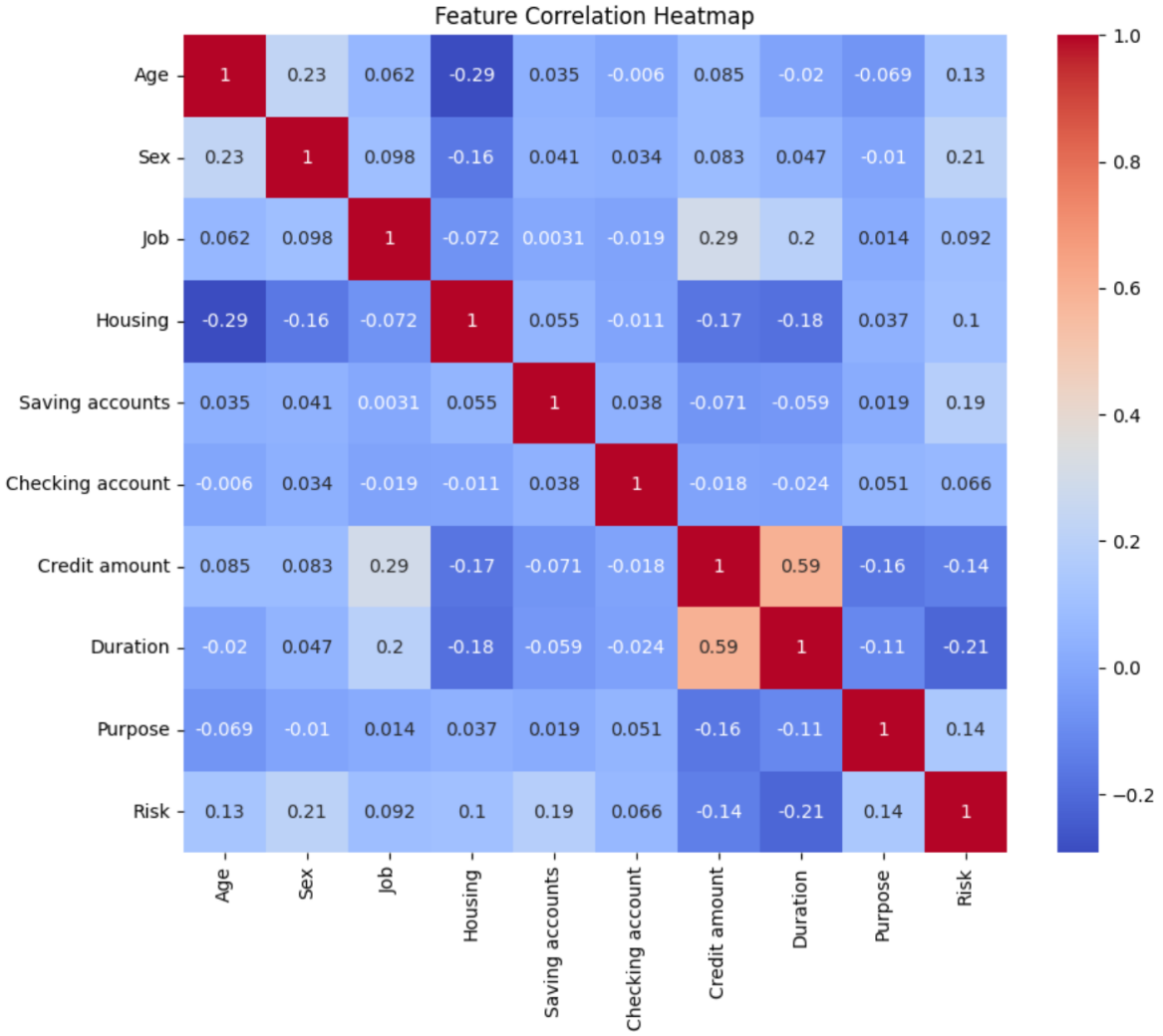

Figure 2.19 Correlation Heatmap

Figure 2.19 shows the correlation heatmap that implies no feature is highly correlated with the target (Risk), suggesting that a combination of multiple features might be important for model performance.

```
X = balanced_df.drop(columns=['Risk'])
y = balanced_df['Risk']
#Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Figure 2.20 Code snippet for splitting the dataset

Figure 2.20 shows the code snippet for splitting the dataset

```
# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Figure 2.21 Code snippet for standardizing the values

Figure 2.21 shows the code snippet for standardizing the dataset

3. Model Selection,Training and Hyperparameter Tuning

A Random Forest classifier was chosen due to its ability to handle both numerical and categorical data and its robustness to overfitting. The model was trained using the preprocessed and balanced data. GridSearchCV was employed to find the best hyperparameter values for the Random Forest model, optimizing for accuracy. This helps fine-tune the model for better performance.

```
[ ] # Define the parameter grid for GridSearchCV
    param_grid = {
        'n_estimators': [100, 200, 300],
        'max_depth': [None, 5, 10],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }

    # Create and train the Random Forest classifier with GridSearchCV
    rf = RandomForestClassifier(random_state=42)
    grid_search = GridSearchCV(rf, param_grid, cv=5, scoring='accuracy')
    grid_search.fit(X_train, y_train)

    # Get the best hyperparameter values
    print("Best hyperparameters:", grid_search.best_params_)

    best_rf = grid_search.best_estimator_
    y_pred = best_rf.predict(X_test)

→ Best hyperparameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
```

Figure 2.22 Code snippet for Model training and Hyperparameter tuning

Figure 2.22 shows the code snippet for Model training and Hyperparameter tuning.Best hyperparameters are 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200

4. Model Evaluation

The model's performance was evaluated using key metrics like accuracy, precision, recall, and F1-score. A confusion matrix was visualized to understand the model's performance in classifying different risk categories.

5. Model Integration with UI

An interactive UI is created and integrated with the model for inputing values and to display the result by the trained model.

2. RESULTS AND DISCUSSION

```
[27] accuracy = accuracy_score(y_test, y_pred)
     accuracy = accuracy_score(y_test, y_pred)
     precision = precision_score(y_test, y_pred)
     recall = recall_score(y_test, y_pred)
     f1 = f1_score(y_test, y_pred)
     conf_matrix = confusion_matrix(y_test, y_pred)
     # Print the metrics
     print(f"Accuracy: {accuracy:.2f}")
     print(f"Precision: {precision:.2f}")
     print(f"Recall: {recall:.2f}")
     print(f"F1-score: {f1:.2f}")

     Accuracy: 0.74
     Precision: 0.73
     Recall: 0.73
     F1-score: 0.73
```

Figure 2.23 Evaluation metrics result

Figure 2.23 shows the different evaluation metrics result.

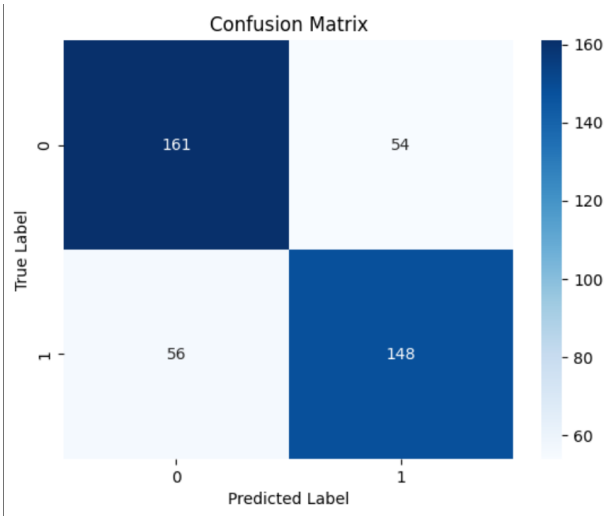

Figure 2.24 Confusion matrix

Figure 2.24 shows the Confusion matrix.

The confusion matrix (Figure 2.24) shows:

- True Negatives (Good risk correctly predicted as good): 161
- False Positives (Good risk predicted as bad): 54
- False Negatives (Bad risk predicted as good): 56
- True Positives (Bad risk correctly predicted as bad): 148

The associated performance metrics are:

- Accuracy: 0.74
- Precision: 0.73
- Recall: 0.73
- F1-score: 0.73

These results indicate a reasonably balanced model with equal treatment of both good and bad risk predictions. Specifically:

- Precision (0.73) reflects that 73% of applicants predicted to be high-risk (bad) were actually high-risk.
- Recall (0.73) shows that 73% of actual high-risk applicants were correctly identified by the model.
- An F1-score of 0.73 indicates a good trade-off between precision and recall.

The slightly lower number of false negatives (56) compared to false positives (54) suggests that the model leans very slightly toward over-predicting good risks, which in a real-world credit setting might pose a moderate risk to lenders if not addressed.

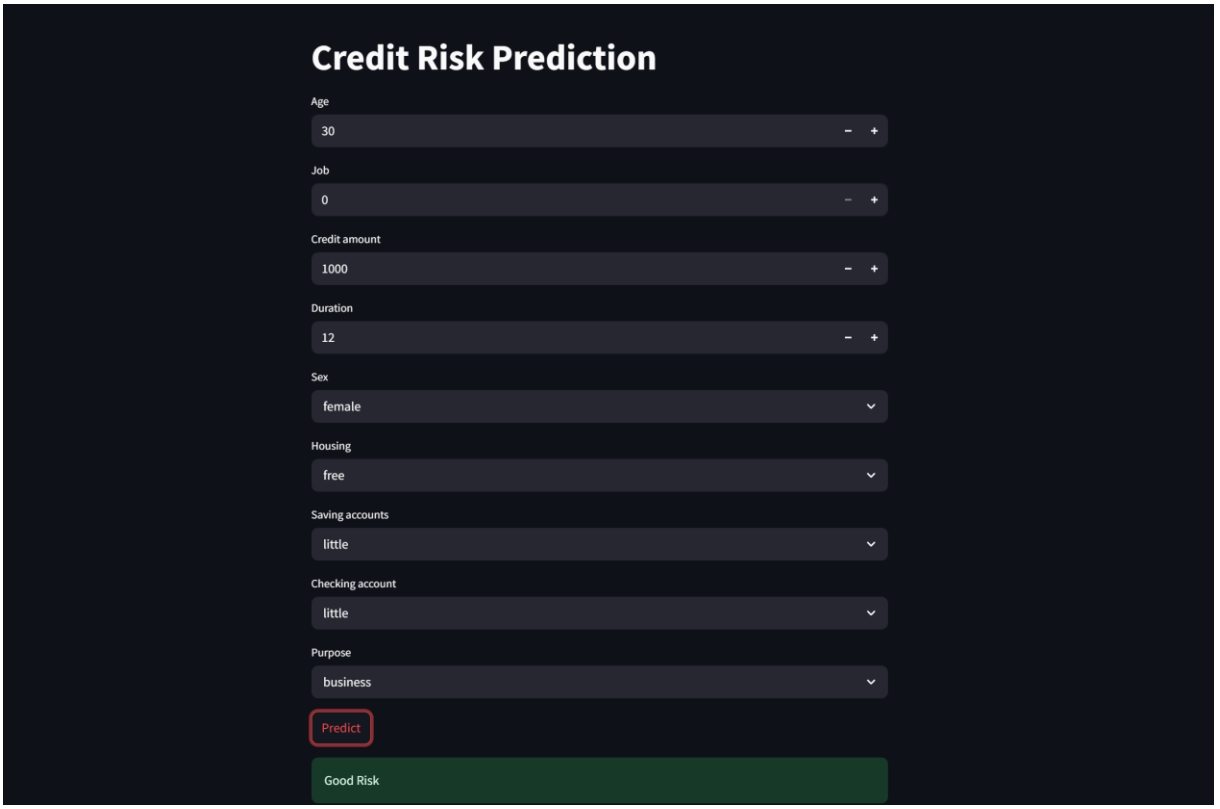

Figure 2.25 UI Design

Figure 2.25 shows the UI Design to which we input values and get the corresponding results.

4. CONCLUSION

This project demonstrated the successful application of machine learning techniques for credit risk prediction using the German Credit dataset. A Random Forest classifier was selected as the core model due to its superior performance in classification tasks, robustness against overfitting, and its ability to model complex interactions between features. Unlike linear models, Random Forest handles both categorical and numerical variables well and automatically accounts for feature importance, making it a strong choice for financial datasets with mixed data types.

To ensure the effectiveness of the model, comprehensive data preprocessing steps were carried out. These included handling missing values, encoding categorical variables, and applying feature scaling. These steps were crucial for preparing the data and ensuring consistency in the learning process. Additionally, hyperparameter tuning using Grid Search was performed to fine-tune the model, thereby improving generalization and accuracy on unseen data.

One of the key challenges with credit risk datasets is class imbalance, where the number of ‘good risk’ applicants significantly outweighs ‘bad risk’ applicants. To address this, the SMOTE algorithm was applied to balance the dataset. This strategy helped the model learn from minority class patterns, resulting in improved recall and F1-scores, especially for the 'bad risk' category.

The final model achieved an accuracy, precision, recall, and F1-score of approximately 0.73, indicating reliable and balanced performance in classifying both risk categories. Insights drawn from visualizations, such as the correlation between savings and risk level, also aligned with practical expectations—showing that applicants with lower savings are generally more likely to be high-risk.

The implemented machine learning approach shows promising potential for automating credit risk assessment in financial institutions. By using a data-driven model, banks and lenders can reduce manual effort, minimize subjective decision-making, and make faster, more consistent lending decisions. This not only reduces financial losses due to defaults but also supports financial inclusion by making credit assessments more accessible and efficient.

In conclusion, the selected Random Forest-based approach, combined with rigorous data handling and optimization, provided a reliable and interpretable solution for credit risk prediction. This work highlights the power of machine learning in transforming traditional credit assessment processes and demonstrates how thoughtful model selection and preprocessing can significantly improve real-world decision-making in the financial sector.