# German Credit Data Project Report


## Summary

An extensive exploration of the German Credit Dataset was conducted, applying multiple machine learning techniques to evaluate credit risk based on defined parameters. The study incorporated Logistic Regression, Decision Trees, K-Nearest Neighbors (KNN), and a hybrid ensemble approach combining Random Forest, Gradient Boosting, and AdaBoost. After hyperparameter tuning, it was found that the Stacking Classifier achieved the highest performance, reaching an accuracy of 76%, though some other models delivered comparable results.


## Introduction

Understanding and evaluating risk accurately is a cornerstone in the world of finance and credit, due to its profound economic consequences. Having a reliable model at disposal aids in informed decision-making, thereby mitigating potential financial mishaps. Classification models have been a key instrument in predicting if a customer represents a high or low risk, thereby aiding in the determination of credit eligibility.

For this research, the German Credit Data was utilized. This dataset fits a classification problem perfectly, where the goal is to categorize a customer's risk scenario into either 'Good' or 'Bad' based on a suite of predictive factors.

## Dataset

The German Credit Dataset is composed of 1,000 individual entries, each representing a unique customer. The dataset encapsulates twenty predictive elements that span both numerical and categorical attributes like 'Age', 'Gender', 'Work', 'Housing', 'Saving accounts', 'Checking account', 'Credit amount', 'Duration', and 'Purpose'. The 'Risk' outcome variable is a binary class marked as 'Good' or 'Bad'.

| | Unnamed: 0 | Age | Sex | Job | Housing | Saving accounts | Checking account | Credit_amount | Duration | Purpose | Risk |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 67 | male | 2 | own | NaN | 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 | NaN | 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 |

To effectively train and evaluate the models, the dataset was split into a training set and a test set, with 80% allocated for training and 20% for testing.. This distribution ensured that we could train our models on a sizable portion of the data while keeping a representative segment for gauging model performance on unseen data.

In the 'good' risk customer group, individuals exhibited attributes associated with a lower risk profile. These customers are often characterized by factors such as age, gender, occupation, housing status, savings and checking accounts status, loan amount, duration, and purpose. For instance, a 67-year-old self-employed male homeowner with a substantial savings balance and a well-managed checking account represents such a case. This individual requests a moderate loan amount over a relatively short term for the purpose of purchasing a radio/TV.

In contrast, 'bad' risk customers demonstrated patterns that reflected higher credit risk. A case in point could be a 22-year-old self-employed female homeowner with minimal savings and a high checking account balance. This person applies for a notably larger loan that she plans to repay over an extended period. Interestingly, the loan purpose mirrors the previous example; that is, to finance a radio/TV.

## Methodology

Preprocessing tasks were necessary before training could begin. These included handling missing values—initially visualized to detect patterns—and replacing them with the most frequently observed entries in each column. In later stages, advanced techniques such as imputation using `HistGradientBoostingRegressor` were explored, and correlations between features (e.g., account balances) and other predictors were examined for more informed imputations.

Categorical variables were encoded as needed using one-hot or ordinal methods. For instance, the binary “Sex” feature was transformed into dummy variables, while ordinal relationships in account types informed ranked encoding. Standardization was applied to continuous features, especially to prevent scale-related bias in distance-based models like KNN. Correlation analyses were conducted between features and the target variable to further guide preprocessing.

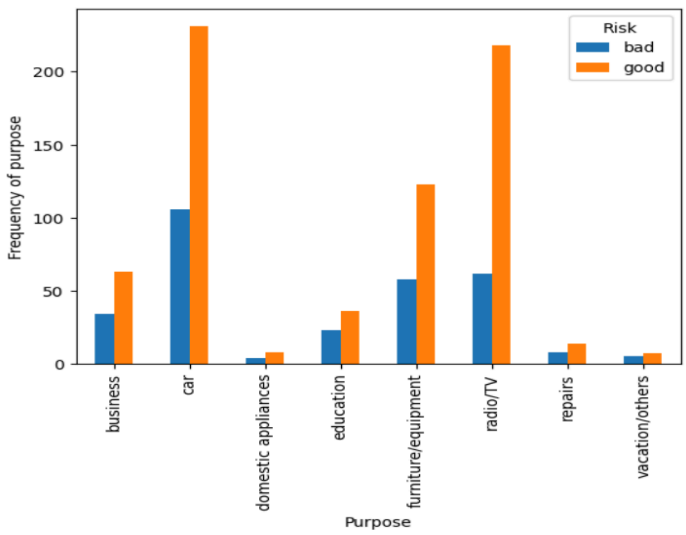
| | Unnamed: 0 | Age | Job | Saving accounts | Checking account | Credit_amount | Duration | Sex_female | Sex_male | Housing_free | ... | Housing_rent | Purpose_business | Purpose_car | Purpose_domestic appliances | Purpose_education | Purpose_furniture/equipment | Purpose_radio/TV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 67 | 2 | 1 | 1 | 1169 | 6 | 0 | 1 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 22 | 2 | 1 | 2 | 5951 | 48 | 1 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 2 | 49 | 1 | 1 | 1 | 2096 | 12 | 0 | 1 | 0 | ... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 3 | 3 | 45 | 2 | 1 | 1 | 7882 | 42 | 0 | 1 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 4 | 4 | 53 | 2 | 1 | 1 | 4870 | 24 | 0 | 1 | 1 | ... | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

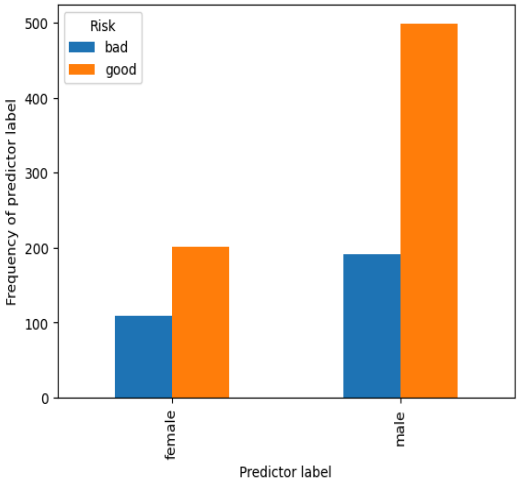
5 rows × 21 columns

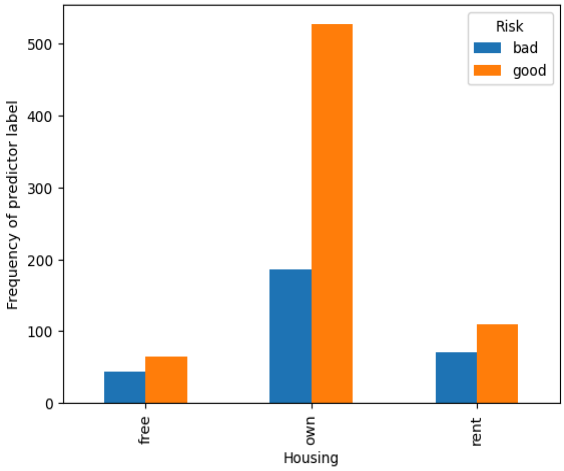
# Experiments

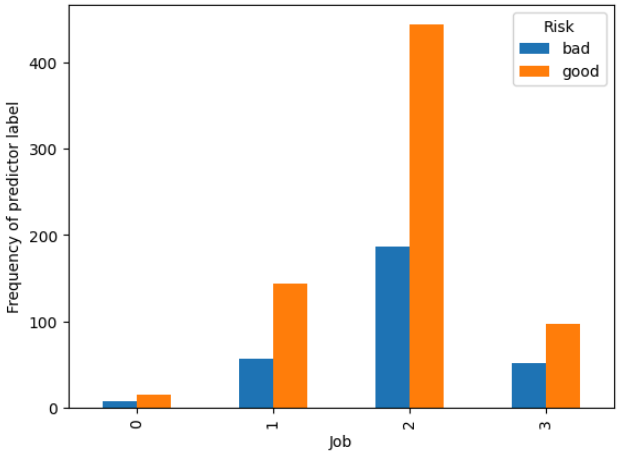
## Visualizations

***Some Predictor vs Target plots**

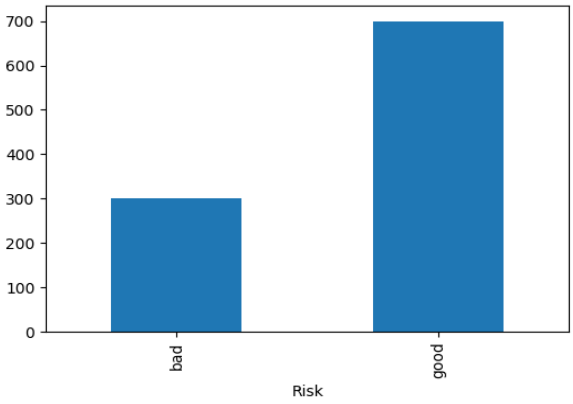





***Plot of out target data:**




## Logistic Regression:

***(Tuned hyperparameters with different "Solvers")***

```
With tuned-parameter lbfgs
Confusion matrix is as
 [[ 12  47]
 [ 11 130]]
Report gives accuracy results as
              precision    recall  f1-score   support

           0       0.52      0.20      0.29        59
           1       0.73      0.92      0.82       141

    accuracy                           0.71       200
   macro avg       0.63      0.56      0.56       200
weighted avg       0.67      0.71      0.66       200

5-fold Cross Validation implemented: [0.71  0.685 0.725 0.71  0.72 ]
Post-Validation Average Accuracy: 0.71
```

```
With tuned-parameter liblinear
Confusion matrix is as
 [[ 12  47]
 [ 12 129]]
Report gives accuracy results as
              precision    recall  f1-score   support

           0       0.50      0.20      0.29        59
           1       0.73      0.91      0.81       141

    accuracy                           0.70       200
   macro avg       0.62      0.56      0.55       200
weighted avg       0.66      0.70      0.66       200

5-fold Cross Validation implemented: [0.71  0.68  0.725 0.71  0.72 ]
Post-Validation Average Accuracy: 0.709
```

```
With tuned-parameter newton-cg
Confusion matrix is as
 [[ 12  47]
 [ 11 130]]
Report gives accuracy results as
              precision    recall  f1-score   support

           0       0.52      0.20      0.29        59
           1       0.73      0.92      0.82       141

    accuracy                           0.71       200
   macro avg       0.63      0.56      0.56       200
weighted avg       0.67      0.71      0.66       200

5-fold Cross Validation implemented: [0.71  0.685 0.725 0.71  0.72 ]
Post-Validation Average Accuracy: 0.71
```

```
With tuned-parameter saga
Confusion matrix is as
 [[ 12  47]
 [ 11 130]]
Report gives accuracy results as
              precision    recall  f1-score   support

           0       0.52      0.20      0.29        59
           1       0.73      0.92      0.82       141

    accuracy                           0.71       200
   macro avg       0.63      0.56      0.56       200
weighted avg       0.67      0.71      0.66       200

5-fold Cross Validation implemented: [0.71  0.685 0.725 0.71  0.72 ]
Post-Validation Average Accuracy: 0.71
```

```
With tuned-parameter sag
Confusion matrix is as
 [[ 12  47]
 [ 11 130]]
Report gives accuracy results as
              precision    recall  f1-score   support

           0       0.52      0.20      0.29        59
           1       0.73      0.92      0.82       141

    accuracy                           0.71       200
   macro avg       0.63      0.56      0.56       200
weighted avg       0.67      0.71      0.66       200

5-fold Cross Validation implemented: [0.71  0.685 0.725 0.71  0.72 ]
Post-Validation Average Accuracy: 0.71
```

*** Logistic Regression with data_for_lr that has been preprocessed with regard to the highest p-values to fill the na's in the Saving accounts and Checking account columns.**

```
Optimization terminated successfully.
         Current function value: 0.574759
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                   Risk   No. Observations:                  800
Model:                          Logit   Df Residuals:                      794
Method:                           MLE   Df Model:                            5
Date:                Mon, 05 Jun 2023   Pseudo R-squ.:                0.06073
Time:                        17:04:12   Log-Likelihood:                -459.81
converged:                       True   LL-Null:                       -489.54
Covariance Type:            nonrobust   LLR p-value:                 1.573e-11
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const              0.1055      0.354      0.298      0.766      -0.588       0.799
Housing            0.1995      0.151      1.320      0.187      -0.097       0.496
Duration          -0.0300      0.008     -3.555      0.000      -0.047      -0.013
Credit_amount  -4.27e-05   3.37e-05     -1.267      0.205      -0.000    2.33e-05
Saving_accounts    0.3378      0.111      3.044      0.002       0.120       0.555
Checking_account   0.5408      0.168      3.223      0.001       0.212       0.870
==============================================================================
Accuracy: 0.755
<ipython-input-20-b63ae1b35c77>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

**Decision Tree Classifier:**

***(Tuned hyperparameters with different "Depths")***

```
DecisionTreeClassifier(max_depth=3)
Confusion matrix is as
 [[ 11  48]
 [  8 133]]
Report gives accuracy results as
              precision    recall  f1-score   support

           0       0.58      0.19      0.28        59
           1       0.73      0.94      0.83       141

    accuracy                           0.72       200
   macro avg       0.66      0.56      0.55       200
weighted avg       0.69      0.72      0.67       200

5-fold Cross Validation implemented: [0.715 0.635 0.69  0.685 0.71 ]
Post-Validation Average Accuracy: 0.687
```

```
DecisionTreeClassifier(max_depth=4)
Confusion matrix is as
 [[  7  52]
 [  5 136]]
Report gives accuracy results as
              precision    recall  f1-score   support

           0       0.58      0.12      0.20        59
           1       0.72      0.96      0.83       141

    accuracy                           0.71       200
   macro avg       0.65      0.54      0.51       200
weighted avg       0.68      0.71      0.64       200

5-fold Cross Validation implemented: [0.71  0.69  0.695 0.685 0.715]
Post-Validation Average Accuracy: 0.699
```

```
DecisionTreeClassifier(max_depth=6)
Confusion matrix is as
 [[ 15  44]
 [ 11 130]]
Report gives accuracy results as
              precision    recall  f1-score   support

           0       0.58      0.25      0.35        59
           1       0.75      0.92      0.83       141

    accuracy                           0.73       200
   macro avg       0.66      0.59      0.59       200
weighted avg       0.70      0.72      0.69       200

5-fold Cross Validation implemented: [0.69  0.67  0.66  0.685 0.67 ]
Post-Validation Average Accuracy: 0.675
```

```
DecisionTreeClassifier(max_depth=7)
Confusion matrix is as
 [[ 14  45]
 [ 18 123]]
Report gives accuracy results as
              precision    recall  f1-score   support

           0       0.44      0.24      0.31        59
           1       0.73      0.87      0.80       141

    accuracy                           0.69       200
   macro avg       0.58      0.55      0.55       200
weighted avg       0.65      0.69      0.65       200

5-fold Cross Validation implemented: [0.67  0.67  0.665 0.675 0.605]
Post-Validation Average Accuracy: 0.6569999999999999
```

```
DecisionTreeClassifier(max_depth=9)
Confusion matrix is as
 [[ 22  37]
 [ 30 111]]
Report gives accuracy results as
              precision    recall  f1-score   support

           0       0.42      0.37      0.40        59
           1       0.75      0.79      0.77       141

    accuracy                           0.67       200
   macro avg       0.59      0.58      0.58       200
weighted avg       0.65      0.67      0.66       200

5-fold Cross Validation implemented: [0.665 0.665 0.64  0.69  0.64 ]
Post-Validation Average Accuracy: 0.66
```

***Plots of decision trees are provided in code**

**Decision tree with further Preprocessed Data and Feature Selection:**

```
Best Hyperparameters: {'max_depth': 3}
Accuracy: 0.705
Confusion Matrix:
[[  1  58]
 [  1 140]]
```

**K-Nearest Neighbors (KNN) Classifier:**

```
Validation accuracy for k= 3  : % 66.875
Confusion matrix is as
 [[ 22  37]
 [ 30 111]]
Report gives accuracy results as
              precision    recall  f1-score   support

           0       0.42      0.37      0.40        59
           1       0.75      0.79      0.77       141

    accuracy                           0.67       200
   macro avg       0.59      0.58      0.58       200
weighted avg       0.65      0.67      0.66       200

5-fold Cross Validation implemented: [0.68  0.645 0.64  0.73  0.655]
Post-Validation Average Accuracy: 0.6700000000000002
```

```
Validation accuracy for k= 7  : % 69.375
Confusion matrix is as
 [[ 22  37]
 [ 30 111]]
Report gives accuracy results as
              precision    recall  f1-score   support

           0       0.42      0.37      0.40        59
           1       0.75      0.79      0.77       141

    accuracy                           0.67       200
   macro avg       0.59      0.58      0.58       200
weighted avg       0.65      0.67      0.66       200

5-fold Cross Validation implemented: [0.695 0.65  0.665 0.705 0.675]
Post-Validation Average Accuracy: 0.6779999999999999
```

```
Validation accuracy for k= 9  : % 70.625
Confusion matrix is as
 [[ 22  37]
 [ 30 111]]
Report gives accuracy results as
              precision    recall  f1-score   support

           0       0.42      0.37      0.40        59
           1       0.75      0.79      0.77       141

    accuracy                           0.67       200
   macro avg       0.59      0.58      0.58       200
weighted avg       0.65      0.67      0.66       200

5-fold Cross Validation implemented: [0.67  0.675 0.665 0.725 0.665]
Post-Validation Average Accuracy: 0.68
```

```
Validation accuracy for k= 11  : % 71.875
Confusion matrix is as
 [[ 22  37]
 [ 30 111]]
Report gives accuracy results as
              precision    recall  f1-score   support

           0       0.42      0.37      0.40        59
           1       0.75      0.79      0.77       141

    accuracy                           0.67       200
   macro avg       0.59      0.58      0.58       200
weighted avg       0.65      0.67      0.66       200

5-fold Cross Validation implemented: [0.68  0.685 0.68  0.725 0.68 ]
Post-Validation Average Accuracy: 0.6900000000000002
```

```
Validation accuracy for k= 16  : % 70.0
Confusion matrix is as
 [[ 22  37]
 [ 30 111]]
Report gives accuracy results as
              precision    recall  f1-score   support

           0       0.42      0.37      0.40        59
           1       0.75      0.79      0.77       141

    accuracy                           0.67       200
   macro avg       0.59      0.58      0.58       200
weighted avg       0.65      0.67      0.66       200

5-fold Cross Validation implemented: [0.67  0.69  0.685 0.745 0.66 ]
Post-Validation Average Accuracy: 0.6900000000000001
```

**KNN with further Preprocessed Data and Feature Selection**

```
Best Hyperparameters: {'n_neighbors': 9}
Accuracy: 0.685
Confusion Matrix:
[[  0  59]
 [  4 137]]
```

**Ensemble Methods: Random Forest, Gradient Boosting, AdaBoost, Stacking, and Voting for Predictive Modeling**

```
Best Random Forest Hyperparameters: {'max_depth': 7, 'n_estimators': 50}
Best Gradient Boosting Hyperparameters: {'learning_rate': 0.5, 'n_estimators': 100}
Best AdaBoost Hyperparameters: {'learning_rate': 0.5, 'n_estimators': 100}
Random Forest Accuracy: 0.74
Gradient Boosting Accuracy: 0.735
AdaBoost Accuracy: 0.75
Stacking Accuracy: 0.76
Voting Accuracy: 0.76
Random Forest Confusion Matrix:
[[ 12  47]
 [  5 136]]
Gradient Boosting Confusion Matrix:
[[ 23  36]
 [ 17 124]]
AdaBoost Confusion Matrix:
[[ 18  41]
 [  9 132]]
Stacking Confusion Matrix:
[[ 17  42]
 [  6 135]]
Voting Confusion Matrix:
[[ 17  42]
 [  6 135]]
```

## Discussion

In this project, classification was made using three different algorithms. These are K-Nearest Neighbors (KNN), Decision Trees (DT), Logistic Regression (LR). Considering the maximum accuracy rate given by the algorithms used, it could be stated that the best algorithm is Logistic Regression.

The confusion matrices we created are as follows:

[TP  FP]  → True Positive , False-Positive

[FN  TN] → False Negative, True Negative

False positive predictions were observed more frequently than false negatives across all models—indicating a tendency toward Type I errors in this classification task.

## Further Methodological Exploration

Beyond the primary classification models, additional exploratory work was conducted to improve prediction accuracy through more advanced preprocessing and ensemble methods. The primary focus was on improving how missing values were handled and understanding the influence of specific features—namely the “Saving accounts” and “Checking account” variables—on model performance.

Initially, missing values in these features were filled using the most frequent category. However, in this phase, a different strategy was tested: imputing missing values using predictions from the `HistGradientBoostingRegressor` model. This approach aimed to account for correlations between these account features and other predictors, thereby producing more context-aware imputations.

To evaluate the effect of this refined imputation, models were retrained and their performances compared. In the K-Nearest Neighbors model, accuracy slightly decreased, whereas in the Decision Tree Classifier, an improvement in performance was observed. These results suggest that the impact of enhanced imputation strategies can vary depending on the learning algorithm used.

Additionally, several ensemble learning methods were applied to further test model robustness and accuracy, including Random Forest, Gradient Boosting, AdaBoost, Voting, and Stacking. Each model’s accuracy was computed, and confusion matrices were generated to assess classification performance and error tendencies.

These advanced experiments provided insight into how different preprocessing choices and ensemble techniques influence overall model outcomes, offering further depth to the original analysis.

## Conclusion

This self-initiated research project provided an end-to-end exploration of credit risk assessment through the lens of machine learning. Using the German Credit Dataset—a well-known benchmark in the field—various classification algorithms were implemented, evaluated, and compared to determine the most effective approach for predicting whether an applicant would be a good or bad credit risk.

The project began with extensive data preprocessing, including imputation of missing values, encoding of categorical features, and standardization of continuous variables. These steps were essential to preparing the data for models such as Logistic Regression, Decision Trees, and K-Nearest Neighbors, as well as for more complex ensemble approaches like Random Forest and Gradient Boosting.

Each model underwent a thorough hyperparameter tuning process using cross-validation, and their final performances were evaluated on an unseen test set. Key performance metrics, including accuracy scores and confusion matrices, were generated to analyze the strengths and weaknesses of each method. Among the tested models, Logistic Regression emerged as a particularly reliable baseline, while ensemble methods demonstrated competitive performance.

In addition to the core experiments, the project also explored more advanced techniques—such as correlation-based imputation and ensemble model combinations—to assess their impact on predictive accuracy. These extensions provided further insights into the role of preprocessing and feature interactions in model performance.

Throughout the study, the application of machine learning in credit risk analysis was not only validated but also enriched through hands-on experimentation. While most findings aligned with expectations, a few surprising observations were made—such as the relatively low number of applicants with “Business” as the loan purpose and the nearly balanced approval/rejection rates in that subgroup.

Overall, the project enhanced understanding of the full machine learning pipeline, from data preparation to model evaluation. It demonstrated how thoughtful model selection and data handling techniques can significantly influence outcomes in real-world financial prediction problems.

**Appendix**

The whole code implementation could be reached from the link below:
https://github.com/gunduzarda/germanCreditData/blob/main/german_credit_data_project.ipynb