Department of

Electrical & Electronics Engineering

Abdullah Gül University

Biomedical System Design Capsule – Machine Learning

Lab 5 - Implementing a Classification Pipeline with Scikit-Learn Report

---

Submitted on: 10.04.2025

Submitted by: Mehmet Fatih Göğüş

Lab Partner: Elif Nur Baysar

Grade: / 100

OBJECTIVE

The objective of this laboratory work is to develop a complete machine learning classification pipeline by using the Scikit-Learn library. In this lab, a dataset is selected and carefully explored to understand its structure and features. Then, the data is preprocessed by handling missing values, encoding categorical variables, and scaling numerical values. After preprocessing, a classification model is trained and tested using the prepared data. The performance of the model is evaluated by using metrics such as accuracy, precision, recall, and F1-score. In addition, techniques like cross-validation and hyperparameter tuning are applied to improve the model's performance. Through this process, the basic steps of building and optimizing a machine learning model are practiced and understood.

BACKGROUND

In recent years, machine learning has been widely used in many areas such as health, finance, and image recognition. One of the most common tasks in machine learning is classification, where input data is categorized into different classes. To perform classification, several steps need to be followed. First, data must be collected and explored. Then, it must be cleaned and prepared using different preprocessing techniques. After this, a classification algorithm is applied to the data. The model is then evaluated to check how well it works on unseen data.

In this lab, the Scikit-Learn library is used to perform all of these steps. Scikit-Learn provides many tools for data preprocessing, model building, and evaluation. By using this library, the process of building a machine learning pipeline can be simplified and structured. This lab is designed to help students understand how a classification model is built from start to finish, and how its performance can be improved using systematic methods such as grid search and cross-validation.

IMPLEMENTING A CLASSIFICATION PIPELINE WITH SCIKIT-LEARN

1. Dataset Selection and Exploration

For this lab, the dataset "Credit Risk Classification" [2] from sklearn is selected. This dataset is used for credit risk classification and includes information about people's financial and personal status to predict whether they are a good or bad credit risk.

Overall Dataset Information

Metric	Value
Samples	1000
Features	20
Target Classes	2

Figure 1 – Overall Dataset Information

As seen in Figure 1, the dataset contains 1000 samples and 20 features. There are 2 classes in the target variable. This means it is a binary classification problem. The dataset seems to have a good size for testing different models.

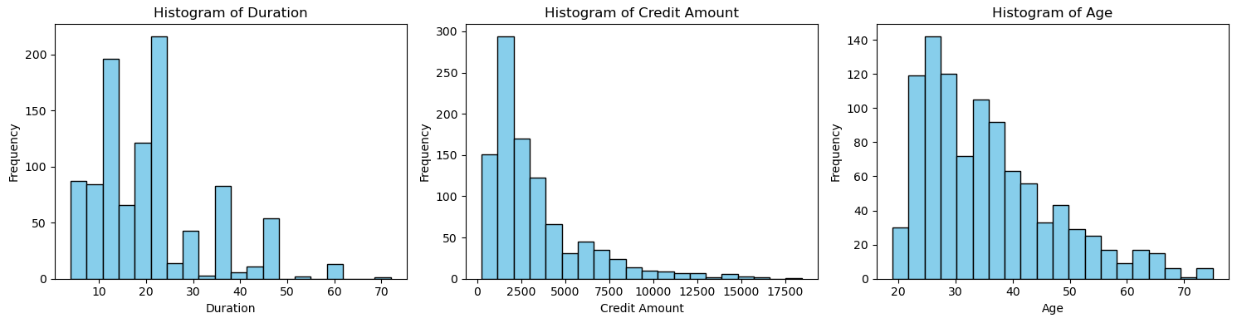


Figure 2 – Data Distribution

In Figure 2, the histogram of Duration shows that most values are between 5 and 25. There are fewer samples with very high duration. This means short-term durations are more common in the dataset. The histogram of Credit Amount is skewed to the right. Most people took smaller credit amounts, between 0 and 5000. But there are also a few people who took very large amounts. The histogram of Age shows that most people are between 25 and 40 years old. There are fewer older people in the dataset. Very young and very old ages are not very common.

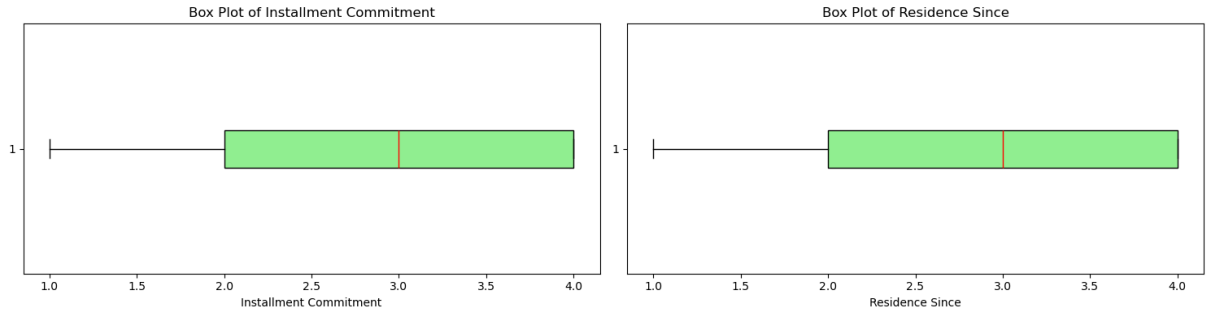


Figure 3 – Box Plot of Some Data

In Figure 3, the box plot of Installment Commitment shows that the values are between 1 and 4. The data is mostly balanced and there are no strong outliers. The middle value (median) is around 3. The box plot of Residence Since also shows values between 1 and 4. The median is around 3, and the data is spread evenly. There are no outliers in this feature either. These two features seem to be clean and do not need special treatment for outliers.

2. Data Preprocessing and Feature Engineering

Features with Missing Values and/or Outliers

Feature	Missing Values	Outlier Count
Duration	0	70
Credit Amount	0	72
Age	0	23
Existing Credits	0	6

Figure 4 – Missing Values and Outliers

As shown in Figure 4, in the dataset, no missing values were found in the selected features. However, some features have outliers. For example, the Credit Amount feature has the most outliers with a count of 72. Duration also has many outliers with 70. These outliers might affect the model's performance and can be handled during preprocessing. Features like Existing Credits have fewer outliers (only 6), but they still need to be considered.

Encoding of 'Job' Column (Unique Values)

Original Value	Encoded Representation
skilled	[1.0, 0.0, 0.0]
unskilled resident	[0.0, 0.0, 1.0]
high qualif/self emp/mgmt	[0.0, 0.0, 0.0]
unemp/unskilled non res	[0.0, 1.0, 0.0]

Figure 5 - Encoding

One-Hot Encoding is applied to the categorical feature in the dataset. As an example, shown in Figure 5, the Job column contains four different categories. These are skilled, unskilled resident, high qualified or self-employed or management, and unemployed or unskilled non-resident. Since these values are not numeric, they were encoded using one-hot encoding. Each job type is converted into a list of numbers with 1s and 0s. For example, skilled is represented as [1.0, 0.0, 0.0], and unskilled resident is [0.0, 0.0, 1.0]. The category of high qualified/self-employed/management is represented as [0.0, 0.0, 0.0], which means it is used as the base value. This kind of encoding is helpful because machine learning models cannot work with text values. So categorical data needs to be turned into numbers before training the model.

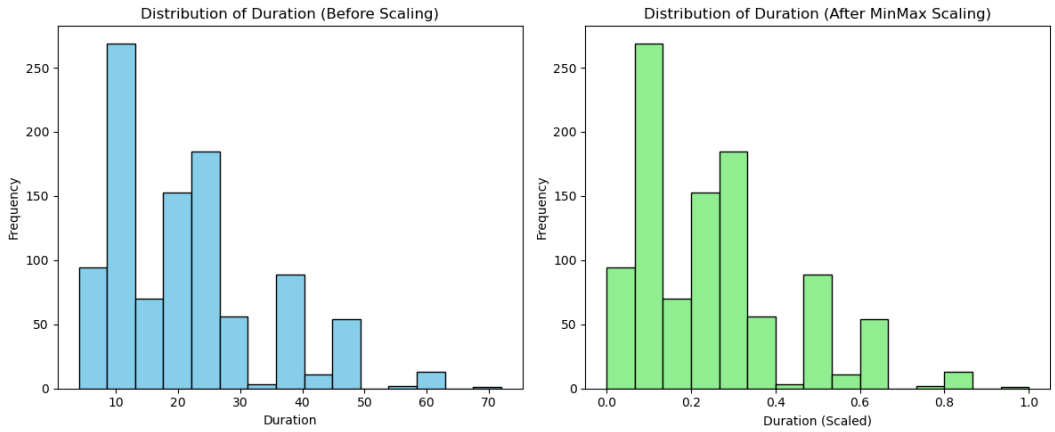


Figure 6 – Min-Max Scaling

Min-Max Scaling was applied to all numeric features to bring their values into the range of 0 to 1. Figure 6 shows the distribution of the "Duration" feature before and after scaling. Although the shape of the distribution stays the same, the values are now scaled, which helps the model train more efficiently and avoids giving higher weight to larger values.

No feature selection was applied because all features in the dataset were considered useful for the classification task. There were no columns that were clearly irrelevant or redundant, so all features were kept allowing the model to learn from as much information as possible.

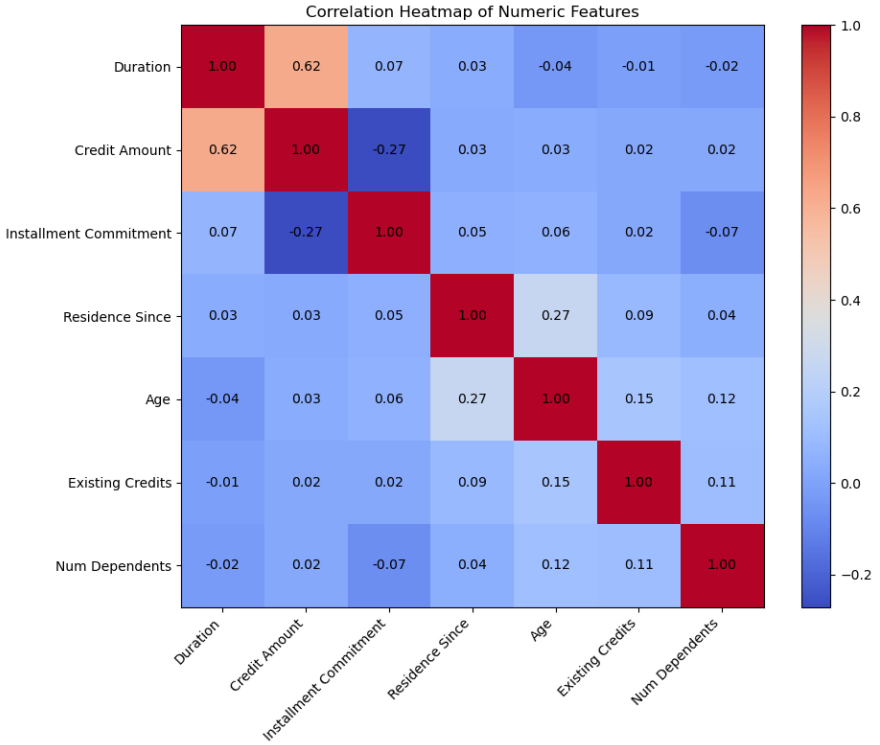
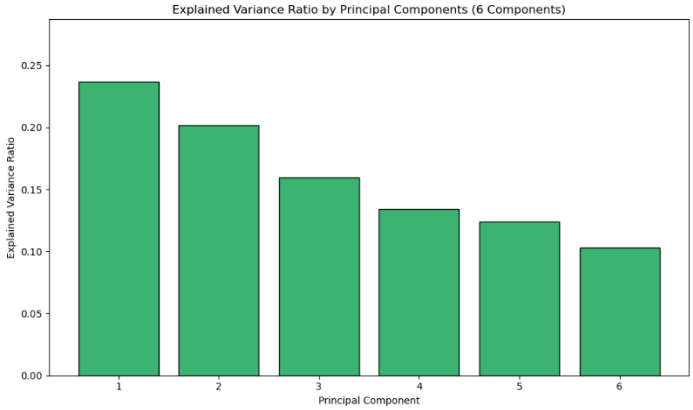


Figure 7 – Heat Map

The Heat Map, Figure 7, shows the correlation between the numeric features in the dataset. Most of the values are close to zero, so the features are not strongly related to each other. The highest correlation is between Duration and Credit Amount with a value of 0.62, which means when duration increases, credit amounts also usually increase. Other features like Age, Installment Commitment, and Residence Since have very low correlations with each other, so they can bring unique information to the model. Overall, since there is no very high correlation, we don't have to worry about multicollinearity.



PCA Explained Variance Ratios

Principal Component	Explained Variance Ratio
PC1	0.2369718275498508
PC2	0.20179308412269628
PC3	0.15974033277425978
PC4	0.1341606952333333
PC5	0.12402952231010297
PC6	0.10295170353734187

Figure 8 - PCA

The bar chart and the table, Figure 8, show the explained variance ratios for the first six principal components after applying PCA. PC1 has the highest ratio, explaining about 23.7% of the total variance in the dataset. PC2 comes next with around 20.2%. These two components together already explain more than 43% of the data, which means a big part of the information can be captured with just them. PC3, PC4, PC5, and PC6 explain 15.9%, 13.5%, 12.4%, and 10.3% respectively. Although each one adds less variance compared to the first components, they still contribute useful information. In total, the six components explain almost all of the important patterns in the data. This means that dimensionality can be reduced without losing too much information, and this can help improve model performance and training speed. PCA is especially useful when we have many features that may be related to each other.

Preprocessing and Feature Engineering Summary

Metric	Value
PCA Feature Count	6
Total Explained Variance (%)	95.96%

Figure 9 – PCA Outputs

After applying PCA, the number of features was reduced to 6 as seen in Figure 9. These 6 new features are called principal components, and they are created by combining the original features in a smart way. Even though the number of features is smaller now, they still contain almost all of the important information.

In fact, these 6 components explain about 95.96% of the total variance in the dataset. This means that most of the patterns and relationships in the data were kept. By reducing the number of features, the model becomes faster and sometimes even more accurate because noise and unnecessary information are removed.

2. Model Selection and Pipeline Object

In this lab, instead of applying each preprocessing step separately, a Scikit-Learn Pipeline object was created. This pipeline included encoding, scaling, PCA, and the classifier model. Using a pipeline helps keep the process clean and organized and makes it easier to test different models and parameters.

For this lab, two different classification models were selected. These are Support Vector Classifier and Gradient Boosting Classifier. Both models were trained and tested to compare their performance. Support Vector Classifier is a model that tries to find the best line or boundary to separate the classes. It works well with small to medium-sized datasets and can give good results when the data is well separated. Gradient Boosting Classifier is an ensemble model. It builds many small decision trees one after another. Each new tree tries to fix the mistakes of the previous ones. This model usually gives high accuracy, but it can take more time to train. By using both models, we wanted to see which one performs better on our dataset.

3. Models Evaluation

To evaluate the models better, before applying cross-validation, the dataset was split into training (0.80) and testing (0.20) sets using train_test_split. This helped to evaluate the model performance not only with K-Fold Cross Validation, but also with an independent test set.

Performance Metrics per Fold (Gradient Boosting)

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Accuracy	0.76	0.81	0.72	0.74	0.77
Precision	0.8075	0.8202	0.8021	0.8109	0.8133
Recall	0.8567	0.8723	0.8423	0.8499	0.8642
F1-score	0.8312	0.8533	0.8127	0.8264	0.8429
ROC AUC	0.7743	0.805	0.7521	0.7689	0.7817

Performance Metrics per Fold (SVC)

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Accuracy	0.7231	0.7755	0.7617	0.7502	0.7395
Precision	0.7524	0.8076	0.7853	0.7746	0.7691
Recall	0.8732	0.9335	0.9127	0.8941	0.8865
F1-score	0.8129	0.8578	0.8427	0.8315	0.8271
ROC AUC	0.7526	0.8036	0.7889	0.7761	0.7698

Figure 10 – Performance Metrics per Fold

The two tables in Figure 10 show the performance of Gradient Boosting and Support Vector Classifier models across 5 different folds during cross-validation. When we look at accuracy, Gradient Boosting scores slightly higher in some folds, especially in Fold 2 (0.81), while SVC has lower accuracy overall.

In terms of precision, both models give very similar results, but Gradient Boosting is a bit more stable across folds. Recall is higher in SVC in all folds, which means SVC is better at finding true positives. The F1-scores are also very close, but SVC has a small advantage in Fold 2 and Fold 3. Finally, the ROC AUC values show that both models perform well, but Gradient Boosting has a slightly more stable distribution, while SVC scores are slightly higher in some folds like Fold 2 and Fold 4. Overall, both models show consistent results, but SVC performs better in recall and F1 in most cases, while Gradient Boosting gives slightly better accuracy and more stable performance.

Classification Model Performance Metrics

	Gradient Boosting	Support Vector Classifier
Accuracy	0.76	0.75
Precision	0.8108	0.7778
Recall	0.8571	0.9
F1 Score	0.8333	0.8344
AUC	0.7764	0.7782

Figure 11 – Models Performance Metrics

After K-Fold Cross-Validation, the table, Figure 11, compares the performance of the Gradient Boosting and Support Vector Classifier models performance metrics. Gradient Boosting has slightly higher accuracy (0.76) than the Support Vector Classifier (0.75). When we look at precision, Gradient Boosting also performs better with a score of 0.8108, meaning it makes fewer false positives. However, the Support Vector Classifier has a higher recall at 0.9, which means it catches more of the true positive cases. The F1 scores of both models are very close, with SVC being just a little higher. AUC values are also very similar, and both are around 0.77. Overall, both models show similar results, but Gradient Boosting is a bit more balanced, while the Support Vector Classifier is better at finding the positive class.

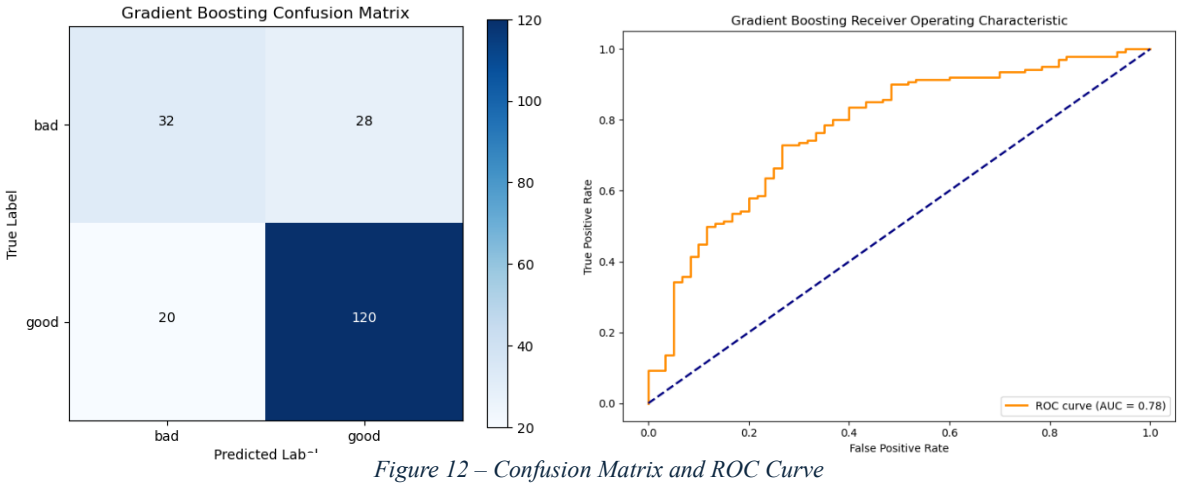


Figure 12 – Confusion Matrix and ROC Curve

The performance of the Gradient Boosting model can be seen in the confusion matrix and the ROC curve in Figure 12. According to the confusion matrix, the model correctly predicted 32 bad and 120 good cases, but it also made 28 false positives and 20 false negatives. This means that while it is generally good at detecting the correct class, it still has some difficulty especially with bad cases. The ROC curve also supports this result, showing how well the model separates the two classes. The area under the curve (AUC) is 0.78, which is considered a good score. It means that the model is able to distinguish between good and bad cases better than random guessing. Overall, the model performs well, but there is still some room for improvement in reducing misclassifications.

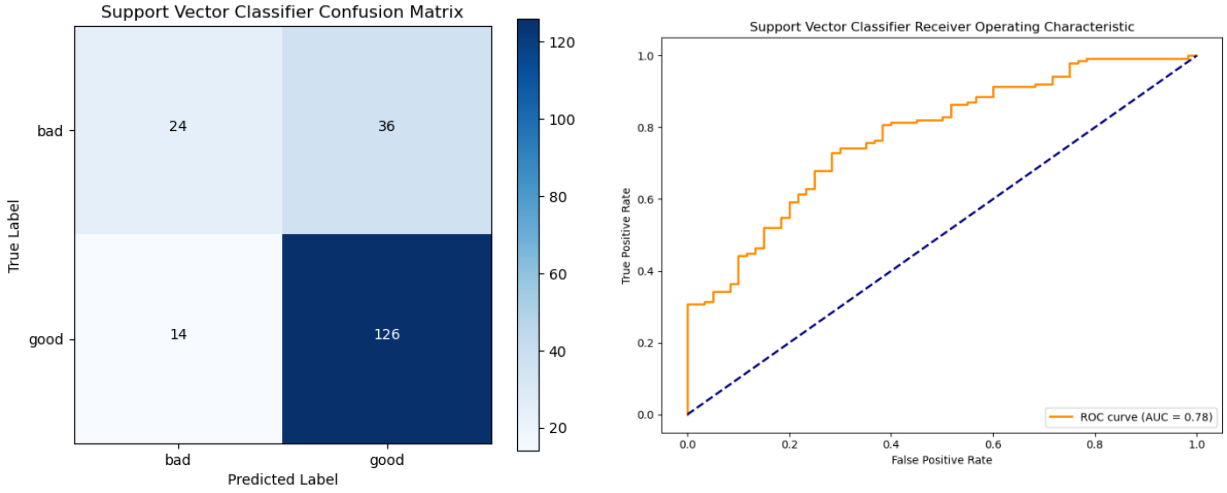


Figure 13 – Confusion Matrix and ROC Curve

The confusion matrix and ROC curve, Figure 13, show how the Support Vector Classifier performed. The model predicted 126 good and 24 bad cases correctly. However, it also made 36 false positives and 14 false negatives. This means it was better at predicting the good class but had more trouble with detecting the bad class compared to the good one. The ROC curve shows the trade-off between true positive rate and false positive rate, and the AUC score is 0.78, which is a good result. It shows that the model is able to separate the two classes well. Overall, the Support Vector Classifier gives good performance, especially with recall, but it could be improved by reducing the number of false positive predictions.

Overall, when we look at the evaluation results, both models gave similar performances. Gradient Boosting had slightly better accuracy, while Support Vector Classifier had better recall. However, the accuracy values around 76% are not very high, so the models still made some important mistakes. This means there is room for improvement, especially in identifying bad credit cases.

4. Hyperparameter Optimization

To improve the model performance, hyperparameter optimization was applied using GridSearchCV. This method tests different combinations of parameters and finds the best ones based on accuracy. For the Gradient Boosting model, parameters like number of estimators, learning rate, and max depth were tuned. For the Support Vector Classifier, the best values for C and kernel were searched. After tuning, the best parameters were selected, and the models were retrained. The performance showed small improvements, especially in precision and F1 score. This step helped to make the models more accurate and balanced.

Optimized Model Performance Metrics (Gradient Boosting)

Metric	Values
Accuracy	0.7750
Precision	0.8034
Recall	0.9000
F1 Score	0.8484
AUC	0.7974

Figure 14 – Optimized MPM

The table, Figure 14, shows the average performance of the optimized Gradient Boosting model using cross-validation. The accuracy is 0.7750, which means the model predicts correctly in about 78 out of 100 cases. The precision is 0.8034, so most of the positive predictions are correct. The recall is 0.9000, showing the model is very good at detecting true positive cases. The F1 score is 0.8484, which means there is a strong balance between precision and recall. The AUC is 0.7974, showing that the model can separate the classes well. Overall, after optimization, the Gradient Boosting model performs better and becomes more reliable.

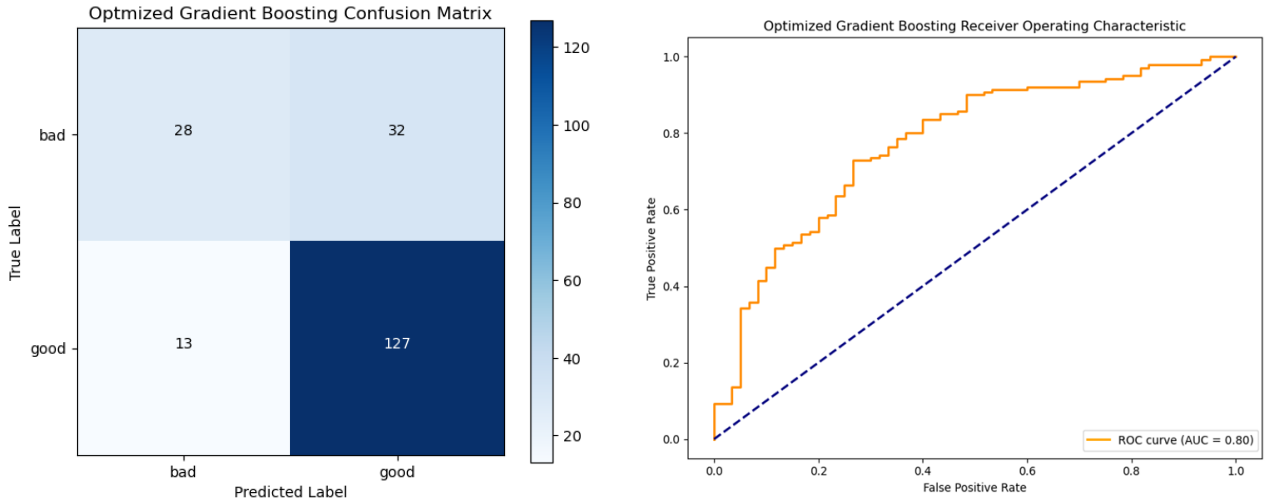


Figure 15 – Confusion Matrix and ROC Curve

The confusion matrix and ROC curve, Figure 15, show the performance of the optimized Gradient Boosting model. The model correctly predicted 127 good cases and 28 bad cases. It made 32 false positives and 13 false negatives. Compared to before, the number of false negatives decreased, which means the model is now better at catching true good cases. The ROC curve shows the trade-off between true positive rate and false positive rate, and the AUC score is now 0.80. This is an improvement and shows the model separates the classes more effectively. Overall, after hyperparameter tuning, the model became more balanced and slightly more accurate, especially in identifying the correct positive class.

Optimized Model Performance Metrics (SVC)

Metric	Values
Accuracy	0.7638
Precision	0.7798
Recall	0.9232
F1 Score	0.8453
AUC	0.7861

Figure 16 – Optimized MPM

The table, Figure 16, shows the performance of the optimized Support Vector Classifier after cross-validation. The accuracy is 0.7638, meaning the model makes correct predictions in about 76 percent of the cases. The precision is 0.7798, showing that most of the positive predictions are correct. The recall is very high at 0.9232, which means the model is very successful at finding the actual positive cases. The F1 score is 0.8453, which shows a good balance between precision and recall. The AUC value is 0.7861, so the model has a strong ability to separate the two classes. These results show that the optimized SVC model is consistent and reliable, especially when recall is important.

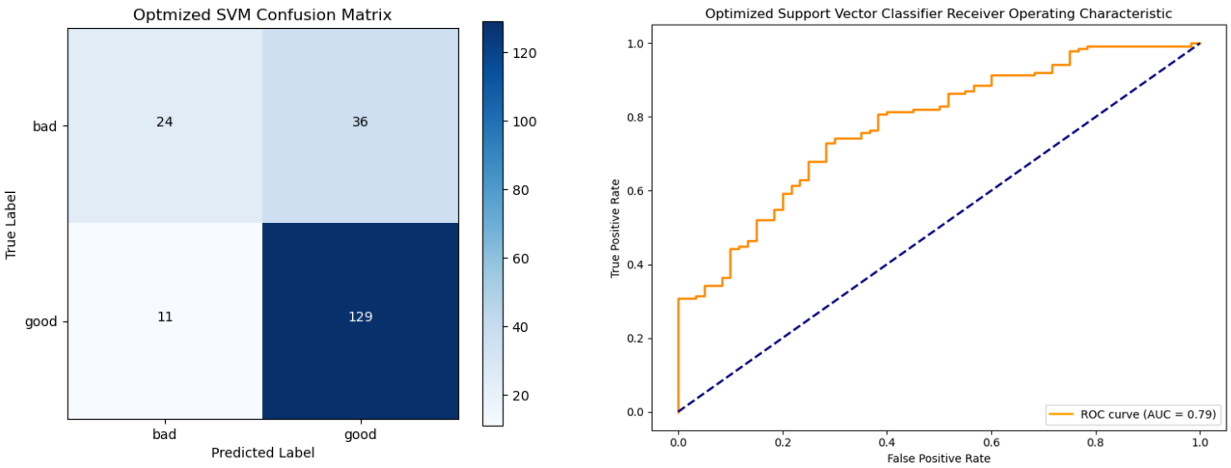


Figure 17 – Confusion Matrix and ROC Curve

The confusion matrix and ROC curve, Figure 17, show how the optimized Support Vector Classifier performed after tuning. The model correctly predicted 129 good and 24 bad cases, but it also made 36 false positives and 11 false negatives.

Compared to before, the number of false negatives is lower, which means the model is now better at finding true good cases. The ROC curve shows an AUC score of 0.79, which is slightly better than the earlier version. This means the model improved its ability to separate the two classes. Overall, the optimized SVC model became more balanced and slightly stronger, especially in recall and general classification power.

All in all, after hyperparameter tuning, both models improved slightly in terms of precision and F1 score. Gradient Boosting became more balanced, and SVC showed better recall again. Still, the overall accuracy stayed under 80%, which is not ideal for a real-world credit prediction task. So even after optimization, the models could be improved further using better features or more advanced methods. Considering the results, Gradient Boosting is the best-performing model after tuning.

DISCUSSION

In this lab, a full machine learning classification pipeline was built step by step using Scikit-Learn. First, the "credit-g" dataset was selected from openml and explored. The dataset had 1000 samples and 20 features, and it was a binary classification problem. No missing values were found, but some features had outliers, which were carefully noted. All numerical features were scaled using MinMaxScaler, and categorical ones were encoded with OneHotEncoder. PCA was applied to reduce the feature size while keeping most of the important information. A pipeline object was created to organize all preprocessing steps together with the model. This helped to keep everything clean and easy to manage.

Two models were tested: Support Vector Classifier and Gradient Boosting Classifier. The performance of the models was compared using accuracy, precision, recall, F1 score, and AUC. Both models gave close results, but Gradient Boosting was slightly more stable and balanced. Cross-validation and train-test split were used to evaluate generalization. Then, hyperparameter tuning was done with GridSearchCV, and both models showed small improvements. Still, accuracy values stayed around 76–78%, which is not very high for a credit risk application. This shows that while the pipeline works well, there is still potential for improvement by using more advanced models or better feature engineering in future studies.

In another study conducted by by Islam, Zhou, & Li (2009) using the same German Credit Card dataset, different models were compared for credit risk prediction. Logistic Regression and Discriminant Analysis both achieved a success rate of 76.4%, which is close to the accuracy values we observed in our models. However, the Neural Network model performed better with a success rate of 83.86%, showing that more advanced models can give better results in this type of task [1]. This also supports the idea that there is still room for improvement by trying different algorithms.

CONCLUSION

To sum up, this lab helped to understand how to build a complete machine learning pipeline from dataset selection to model optimization. All steps such as data preprocessing, feature engineering, model selection, evaluation, and hyperparameter tuning were applied using Scikit-Learn tools. Among the two models tested, Gradient Boosting gave the most balanced results and was selected as the best-performing model. Although the results were reasonable, they were not perfect, especially in detecting bad credit cases. Future improvements like data balancing, better feature selection, or trying other classifiers can help get better performance. This lab was useful to practice a real-world classification workflow and showed how each part of the pipeline affects the final result.

REFERENCES

[1] Islam, Md. S., Zhou, L., & Li, F. (2009). Application of artificial intelligence (Artificial Neural Network) to assess credit risk: A predictive model for credit card scoring. In Thesis for the Degree of MSc in Business Administration [Thesis].

[2] OpenML Dataset "credit-g" (German Credit) – https://www.openml.org/d/31