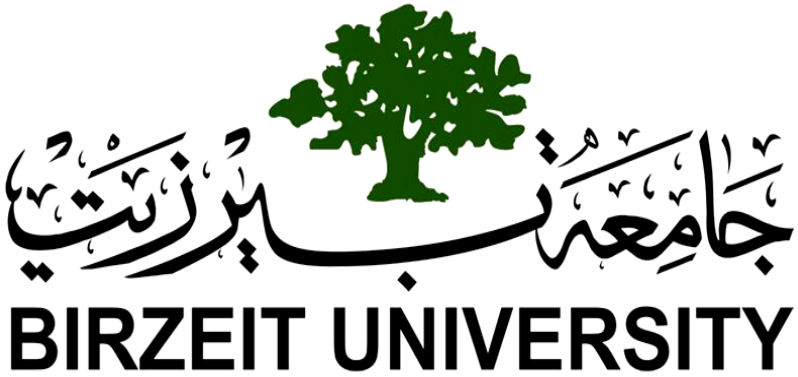

Faculty of Engineering & Technology
Electrical & Computer Engineering Department

ENCS5141-Intelligent Systems Laboratory

Case Study #2: Comparative Analysis of Classification Techniques: Support Vector Machine (SVM) and Multilayer Perceptron (MLP) for Bank Notes Datasets

---

Instructor:
Dr.Yazan AbuFarha

Prepared By:
Abd Khuffash-1200970

Section:
2

Date:
27-4-2024

Abstract/Objectives

This case study investigates and compares the performance of Support Vector Machines (SVM) and Multilayer Perceptron (MLP) classification techniques on a dataset comprising over 24,000 records of banknotes across various scenarios. Two options are explored: creating separate classifiers for predicting currency, denomination, and orientation labels, and merging them into a combined label for a single classifier. Hyperparameter tuning techniques such as GridSearchCV are applied to optimize model performance. Additionally, dimensionality reduction techniques like Principal Component Analysis (PCA) are utilized to reduce data size while retaining essential information. The study evaluates and compares the performance of SVM and MLP models before and after dimensionality reduction. Visualizations including feature distribution, class distribution, correlation matrix, and PCA visualization provide insights into the dataset's characteristics and model behavior. Results are presented through accuracy scores, classification reports, confusion matrices, and visualization plots, offering a comprehensive understanding of the strengths and limitations of SVM and MLP classification techniques in predicting currency, denomination, and orientation labels for banknotes.

Table of Contents

Contents

Table of Figures

Introduction

In the realm of machine learning, the selection and implementation of appropriate classification techniques play a pivotal role in achieving accurate predictions across various domains. In this case study, we delve into the comparative analysis of two prominent classification algorithms: Support Vector Machines (SVM) and Multilayer Perceptron (MLP). Our focus lies on understanding the performance of these algorithms in predicting the currency type, denomination, and orientation of banknotes based on a dataset encompassing over 24,000 records.

Support Vector Machines (SVM):

Support Vector Machines (SVM) are a powerful class of supervised learning algorithms primarily used for classification tasks. SVM aims to find the optimal hyperplane that separates different classes in the feature space while maximizing the margin between them. This hyperplane is determined by support vectors, which are the data points closest to the decision boundary.[1]

Key features of SVM include:

- Effective in high-dimensional spaces: SVM works well even in cases where the number of dimensions is greater than the number of samples.
- Versatile kernel functions: SVM supports various kernel functions such as linear, polynomial, radial basis function (RBF), and sigmoid, allowing it to handle non-linear decision boundaries.
- Margin maximization: SVM seeks to maximize the margin between classes, which helps improve generalization and reduce overfitting.
- Robust to outliers: SVM is relatively robust to outliers due to its focus on maximizing the margin.[1]

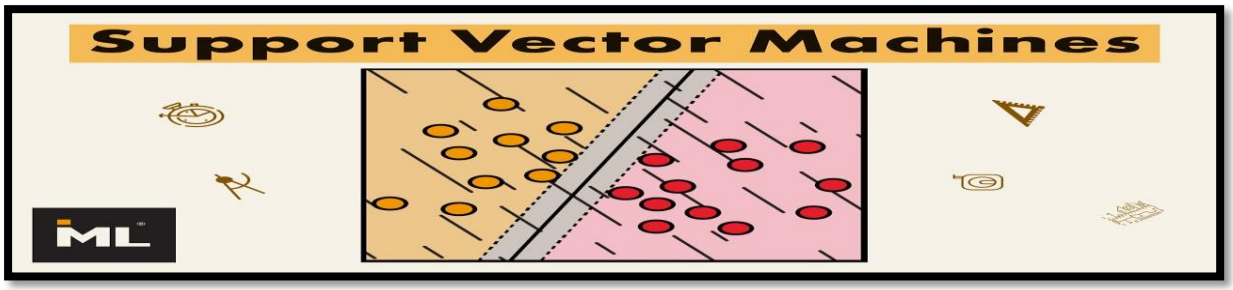

Figure 1. Support Vector Machine

Multilayer Perceptron (MLP):

Multilayer Perceptron (MLP) is a type of artificial neural network consisting of multiple layers of nodes (or neurons), each connected to the next layer. MLP is a versatile and powerful architecture capable of approximating complex non-linear functions and is widely used for both classification and regression tasks.[1]

Key features of MLP include:

- Non-linear activation functions: MLP uses non-linear activation functions such as sigmoid, tanh, or ReLU (Rectified Linear Unit) to introduce non-linearity into the model, enabling it to learn complex patterns in the data.
- Backpropagation algorithm: MLP is trained using the backpropagation algorithm, which adjusts the weights of connections between neurons to minimize the error between predicted and actual outputs.
- Flexibility in architecture: MLP allows for flexibility in the number of layers, number of neurons in each layer, and choice of activation functions, making it adaptable to a wide range of tasks.
- Ability to learn complex relationships: MLP's layered structure allows it to learn hierarchical representations of data, enabling it to capture complex relationships between input features and target outputs.[1]

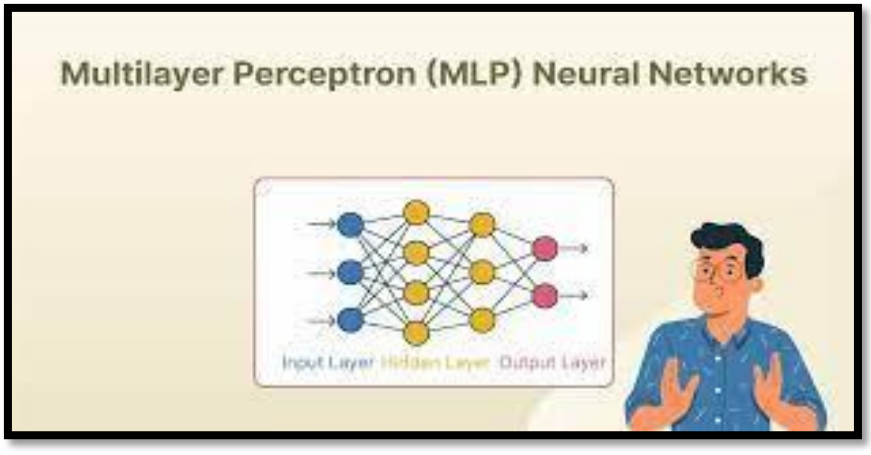

Figure 2. Multilayer Perceptron

The primary objective of this study is twofold: to grasp the strengths and limitations of SVM and MLP in the context of classification tasks, and to gain insights into their applicability across diverse scenarios. By rigorously evaluating these algorithms on a comprehensive dataset, we aim to provide valuable guidance for practitioners and researchers in selecting the most suitable technique for similar classification tasks.

To achieve our objectives, we opted for the second approach, merging the currency, denomination, and orientation labels into a combined label. This approach enables us to assess the effectiveness of a single classifier in predicting this amalgamated label, offering a more holistic view of the classification task. By consolidating the labels, we aim to streamline the prediction process and evaluate the overall performance of Support Vector Machines (SVM) and Multilayer Perceptron (MLP) in predicting this combined label. This approach not only simplifies the classification task but also provides insights into the models' ability to handle multiple label predictions simultaneously.

Throughout this study, we employ various techniques to optimize model performance, including hyperparameter tuning using GridSearchCV and dimensionality reduction using Principal Component Analysis (PCA). Visualizations such as feature distribution plots, class distribution histograms, correlation matrices, and PCA visualizations offer deeper insights into the dataset's characteristics and aid in interpreting the behavior of the classification models.

By presenting detailed analyses, evaluation metrics, and visualizations, we aim to provide a comprehensive understanding of the capabilities and limitations of SVM and MLP classification techniques in the context of banknote classification. This study serves as a valuable resource for practitioners and researchers seeking to leverage machine learning for similar classification tasks in diverse domains.

Procedure and Discussion

The study begins by importing the necessary libraries to facilitate data handling, model training, evaluation, and visualization. These libraries include:

1. **pandas** (**import pandas as pd**): Used for data manipulation and analysis.
2. **numpy** (**import numpy as np**): Essential for numerical computations and array manipulations.
3. **seaborn** (**import seaborn as sns**): Enables data visualization and statistical graphics.
4. **matplotlib.pyplot** (**import matplotlib.pyplot as plt**): Widely used for creating static, interactive, and animated visualizations.
5. **%matplotlib inline**: Magic command to display plots directly in the Jupyter Notebook.
6. **train_test_split** (**from sklearn.model_selection import train_test_split**): Splits the dataset into training and testing sets for model evaluation.
7. **classification_report, accuracy_score** (**from sklearn.metrics import classification_report, accuracy_score**): Evaluation metrics for assessing the performance of classification models.
8. **MinMaxScaler** (**from sklearn.preprocessing import MinMaxScaler**): Scales features to a specified range (usually [0, 1]).
9. **SVC** (**from sklearn.svm import SVC**): Support Vector Machine classifier for classification tasks.
10. **confusion_matrix, roc_curve, auc** (**from sklearn.metrics import confusion_matrix, roc_curve, auc**): tools for evaluating the performance of classification models.
11. **itertools** (**import itertools**): Provides various functions for creating iterators and combining data.
12. **precision_score, recall_score, f1_score** (**from sklearn.metrics import precision_score, recall_score, f1_score**): Additional evaluation metrics for assessing classification model performance.
13. **GridSearchCV** (**from sklearn.model_selection import GridSearchCV**): Hyperparameter tuning technique for finding the best model parameters.
14. **MLPClassifier** (**from sklearn.neural_network import MLPClassifier**): Multilayer Perceptron classifier for classification tasks.
15. **warnings** (**import warnings**): Provides utilities for handling warnings in Python.

These libraries provide essential functionalities for data preprocessing, model training, evaluation, hyperparameter tuning, and visualization, which are integral parts of the machine learning workflow. By importing these libraries, we ensure that we have access to a wide range of tools and methods necessary for conducting a comprehensive analysis of the dataset and training machine learning models.

Then, We began by loading the dataset from the specified path ("/content/drive/MyDrive/Colab Notebooks/Lab AI/BankNotesDatasetreduced.csv"). Upon loading the dataset, we observed that it contains 24826 entries and 259 columns. Using dataset.head(), we examined the first 6 rows of the dataset to gain a preliminary understanding of its structure and contents.

Further exploration involved generating summary statistics for the numerical columns using dataset.describe().round(2), which provided insights into the central tendency and dispersion of the data. Additionally, we obtained detailed information about the dataset's columns, data types, and memory usage using dataset.info(). This revealed that the dataset consists of float64 (256 columns), int64 (1 column), and object (2 columns) data types, with no missing values detected across any columns.

The analysis of the dataset revealed several key findings:

1. **Dataset Size and Structure**:
    - The dataset comprises 24826 entries and 259 columns, indicating a substantial amount of data available for analysis.
    - Each entry represents a unique set of features related to banknotes, including numerical values and categorical labels such as currency and denomination.
2. **Data Types**:
    - The dataset consists primarily of float64 (256 columns) and int64 (1 column) data types for numerical features, along with object data type (2 columns) for categorical labels.
    - This diverse data type distribution suggests a mix of continuous and categorical variables in the dataset.

3. **Missing Values**:

    - No missing values were detected across any of the columns, indicating a complete dataset without any significant data gaps.
    - This ensures that the dataset is ready for further analysis and modeling without the need for imputation or handling missing values.

4. **Initial Exploration**:

    - Examination of the first 6 rows of the dataset provided an initial glimpse into the structure and content of the data, facilitating a better understanding of its format and organization.
    - Summary statistics generated for the numerical columns offered insights into the distribution and variability of the numerical features.

Overall, these results suggest that the dataset is well-prepared and suitable for further analysis and modeling tasks. With no missing values and a diverse set of features, it presents a valuable opportunity for exploring and developing machine learning models to predict banknote attributes.

After encoding the target labels (Currency and Denomination) using LabelEncoder, the encoded values were combined into a new string label named 'Combined_Label'. This combined label was further encoded using LabelEncoder to create a unique numerical value for the targets, referred to as 'Goal'. The original labels and intermediate columns were dropped, leaving only the feature set ('x') and the target labels ('y').

The dataset was then divided into training and testing sets using the train_test_split function from the sklearn library. The split was performed with a ratio of 80% for training data and 20% for testing data, with stratification based on the target labels to ensure balanced distribution in both sets.

Additionally, Principal Component Analysis (PCA) was applied to reduce the dimensionality of the feature set while retaining 95% of the variance. This technique helps in reducing computational complexity and potentially improving model performance by focusing on the most informative features.

Furthermore, MinMaxScaler was used to normalize the feature set, ensuring that each feature contributes equally to distance calculations and optimization algorithms. Normalization is essential for enhancing the performance and convergence of machine learning models, particularly those sensitive to feature scaling differences.

The next step was visualizing the data, The visualization of the dataset provides valuable insights into its structure and characteristics. Firstly, the feature distribution was visualized through histograms with kernel density estimation (KDE) to understand the distribution of each feature across the dataset. These histograms offer a clear depiction of the spread and density of values within each feature, allowing for the identification of potential outliers or skewed distributions. The seaborn library was utilized to create these informative visualizations, facilitating a comprehensive exploration of the dataset's numerical attributes.

(For improved visualizations, please refer to the Jupyter notebook.).

Figure 3.Feature Distribution Visualization

Following the feature distribution visualization, the class distribution within the training set was examined through a bar chart. This visualization illustrates the frequency of each class label, offering insights into the balance or imbalance of the dataset across different target categories. Understanding the distribution of classes is crucial for assessing the dataset's representativeness and potential biases, which can significantly impact the performance of machine learning models trained on the data. (For improved visualizations, please refer to the Jupyter notebook.).

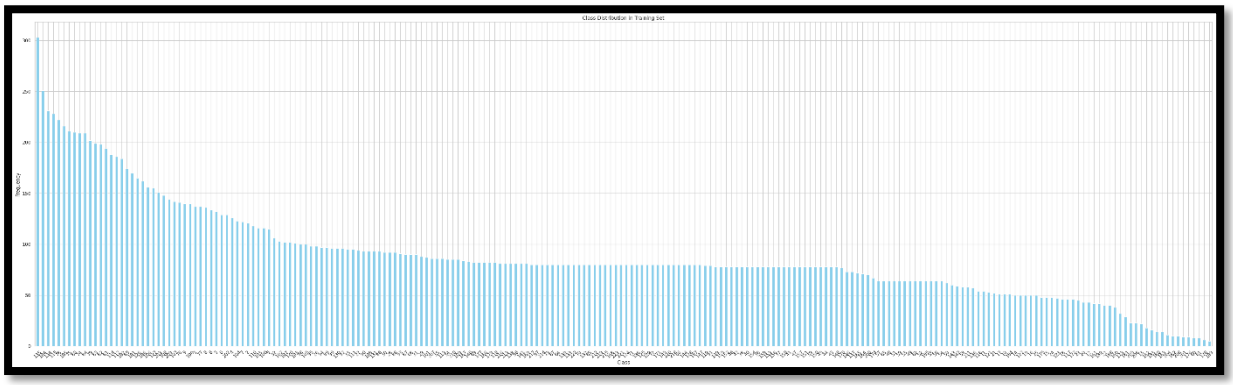

Figure 4.Class Distribution Visualization

To explore potential relationships and dependencies between features, a correlation matrix was computed and visualized using a heatmap. This matrix provides a comprehensive overview of the pairwise correlations between features, with annotations indicating the strength and direction of these correlations. By identifying correlated features, this visualization aids in feature selection and dimensionality reduction efforts, helping to mitigate issues such as multicollinearity and improving the interpretability of the model. (For improved visualizations, please refer to the Jupyter notebook.).

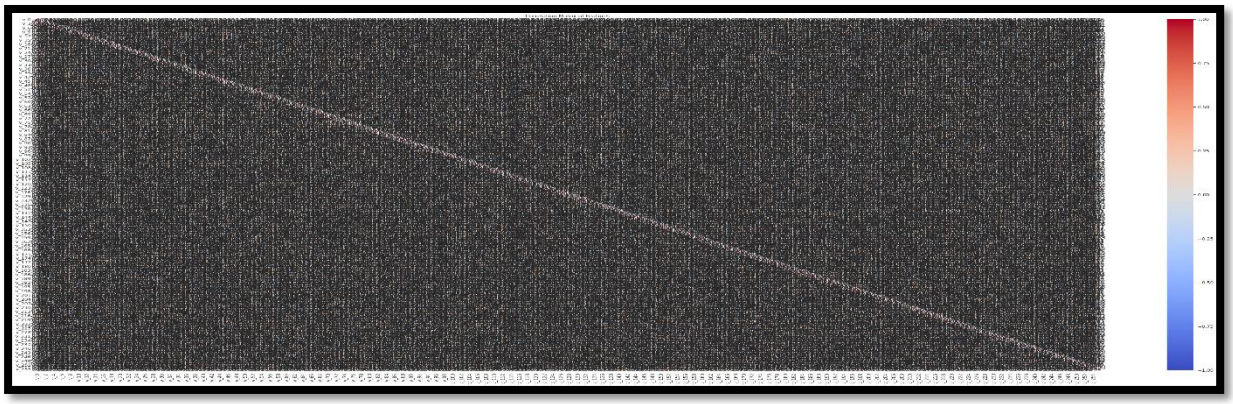

Figure 5.Correlation Matrix heatmap

Principal Component Analysis (PCA) was employed to reduce the dimensionality of the feature set while preserving essential information. The results of PCA were visualized through a scatter plot, with data points colored according to their respective class labels. This visualization allows for the exploration of the dataset's structure in a lower-dimensional space, facilitating a deeper understanding of the relationships between data points and classes. (For improved visualizations, please refer to the Jupyter notebook.).

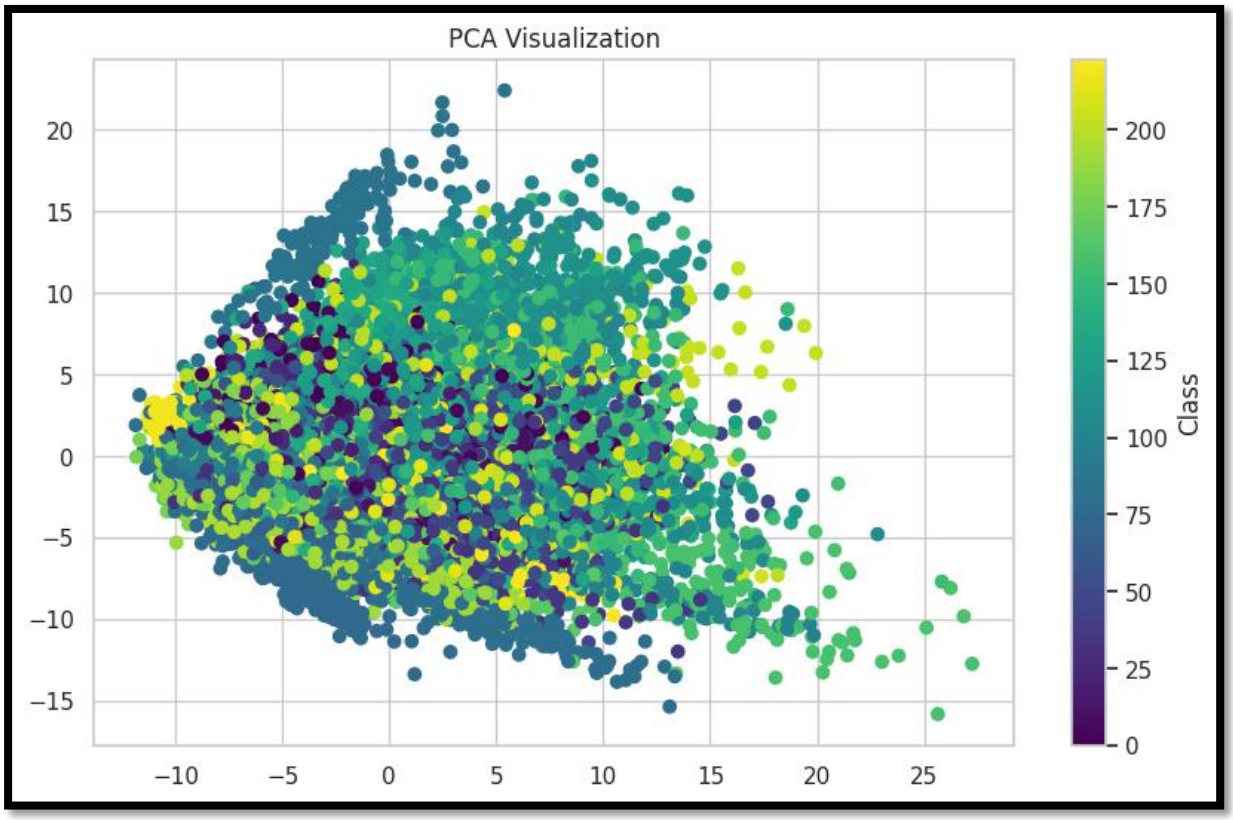

Figure 6. PCA Visualization

Finally, the distribution of classes within both the training and testing sets was compared through a bar chart. This visualization ensures that both sets maintain a similar distribution of class labels, indicating a balanced split between the training and testing data. A balanced distribution is essential for the generalization performance of machine learning models, as an imbalanced dataset may lead to biased predictions and suboptimal model performance.

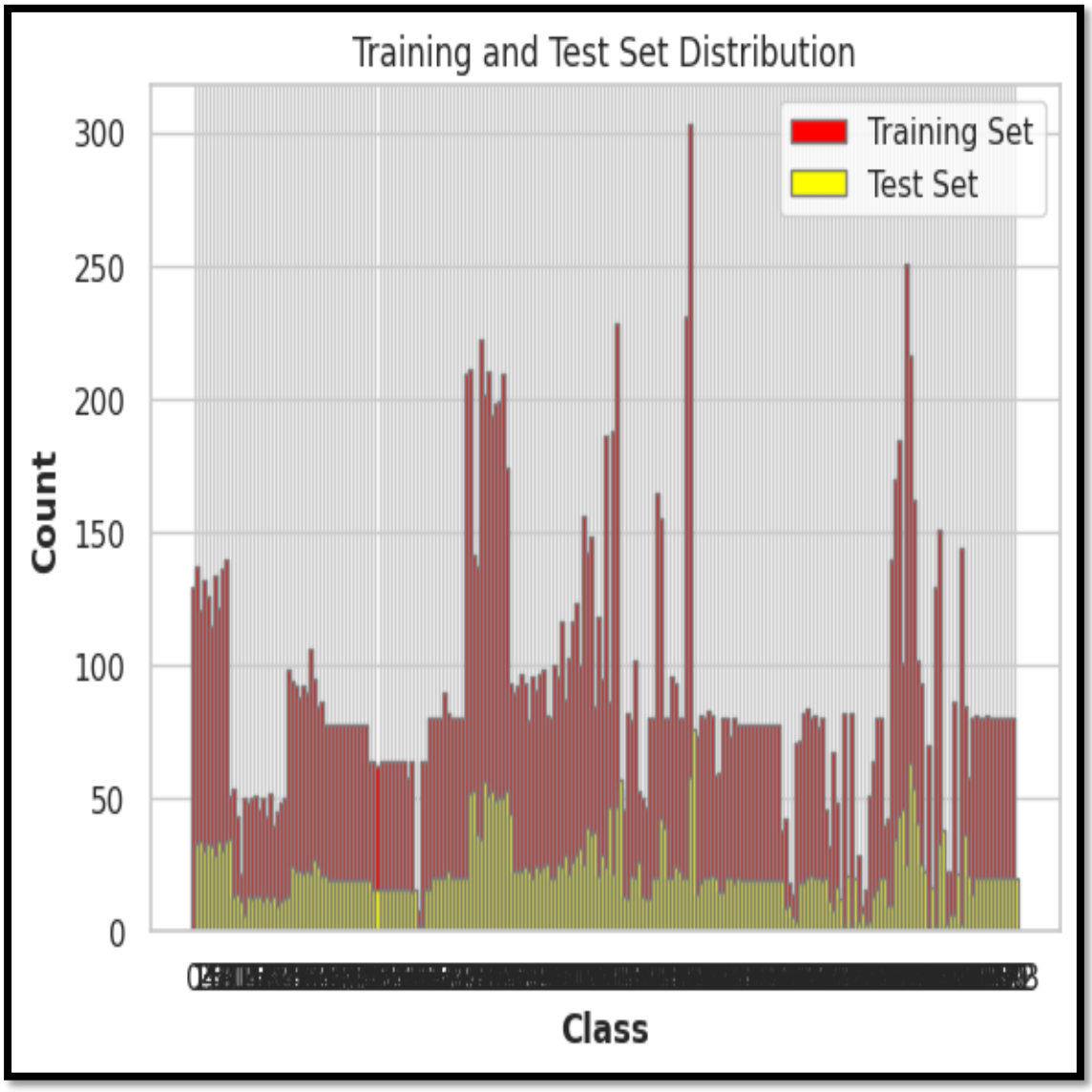

Figure 7. Training and testing set Distribution

In summary, visualizing the dataset serves as a fundamental step in understanding its underlying structure and characteristics, providing essential insights for effective data analysis and model building. Through histograms and KDE plots, we gain a comprehensive view of feature distributions, enabling the identification of outliers and skewed data points. Class distribution visualizations offer insights into the balance or imbalance of target categories, crucial for assessing dataset representativeness and mitigating biases in machine learning models. Correlation matrices aid in identifying feature relationships, facilitating feature selection and dimensionality reduction. PCA offers a way to reduce dimensionality while preserving essential information, allowing for a deeper understanding of the dataset's structure. Lastly, comparing class distributions between training and testing sets ensures a balanced split, essential for optimal model performance. Overall, these visualizations provide valuable insights that guide key decisions throughout the data analysis and modeling process, ultimately leading to more accurate and robust machine learning models.

These preprocessing steps prepare the data for further analysis and modeling, ensuring that it is appropriately formatted and scaled for training machine learning models such as Support Vector Classifier (SVC) and Multilayer Perceptron (MLP).

In the subsequent steps, a smaller subset of the dataset was utilized (2k records) instead of the entire dataset consisting of 24k records. This decision was made to alleviate the lengthy training time required when working with the complete dataset, which could exceed three hours. By opting for a sample, the computational burden of training the models was significantly reduced, leading to quicker model training and experimentation. Despite working with a smaller portion of the data, this approach still allowed for meaningful analysis and model development. It facilitated faster iterations and experimentation with different modeling techniques and parameters, enabling a more efficient development process without compromising the quality of the results.

Support Vector Classifier (SVC):

The Support Vector Classifier (SVC) model was implemented and evaluated using the provided dataset. Initially, the SVC model was trained on the training data and subsequently tested on the testing data. The accuracy of the SVC model on the testing data was approximately 0.869, indicating the proportion of correctly classified instances.

The classification report provided detailed metrics such as precision, recall, and F1-score for each class label. These metrics offer insights into the model's performance for individual classes, including its ability to correctly classify instances, identify true positives and false positives, and balance precision and recall. Additionally, a confusion matrix was generated to visualize the model's classification results, illustrating the number of correct and incorrect predictions for each class.

To optimize the SVC model further, GridSearchCV was employed to search for the best hyperparameters. The hyperparameters 'C', 'gamma', 'kernel', and 'degree' were tuned through this process, aiming to enhance the model's predictive accuracy. After performing grid search with 5-fold cross-validation, the best parameters were identified, and the model was retrained using these optimal parameters. The accuracy of the optimized SVC model on the testing data improved to approximately 0.892, indicating a notable enhancement in classification performance.

The results obtained from the Support Vector Classifier (SVC) model showcase its effectiveness in accurately classifying the dataset across multiple class labels. Initially, the model achieved a commendable accuracy of approximately 86.9% on the testing data, indicating its ability to correctly classify the majority of instances. This baseline performance already suggests that the SVC model has learned meaningful patterns and relationships within the dataset.

Delving deeper into the classification report, it becomes evident that the model exhibits strong precision, recall, and F1-score values for most class labels. For instance, classes such as 0, 1, 8, 9, and 10 demonstrate high precision, recall, and F1-score values, signifying the model's capability to effectively identify and classify instances belonging to these categories. Conversely, some classes, such as 4, 5, and 6, exhibit slightly lower performance metrics, indicating potential areas for improvement in the model's ability to distinguish instances from these classes.

The confusion matrix provides a visual representation of the model's classification results, highlighting the number of correct and incorrect predictions for each class. Observing the confusion matrix allows for the identification of specific classes where the model may struggle to make accurate predictions. In the case of the SVC model, certain misclassifications may occur between closely related classes or classes with overlapping features, which could be further explored and addressed in future model iterations.

Furthermore, the optimization process using GridSearchCV yielded notable improvements in the model's performance. By systematically tuning hyperparameters such as 'C', 'gamma', 'kernel', and 'degree', the optimized SVC model achieved an accuracy of approximately 89.2% on the testing data, representing a substantial enhancement compared to the baseline performance. This improvement underscores the importance of hyperparameter tuning in fine-tuning the model's predictive capabilities and maximizing its performance on unseen data.

Overall, the results obtained from the SVC model underscore its efficacy in accurately classifying the dataset and highlight the significance of thorough evaluation and optimization techniques in enhancing model performance. By leveraging detailed evaluation metrics and visualization tools, such as the classification report and confusion matrix, practitioners can gain deeper insights into the strengths and weaknesses of the model, informing future iterations and improvements to the classification task.

MLPClassifier:

After training the MLPClassifier model with default hyperparameters, it achieved an accuracy of 88.18% on the test set. This model demonstrated relatively good performance across different classes, as indicated by the precision, recall, and F1-score values in the classification report. However, there were variations in performance among different classes, with some classes achieving higher precision and recall scores than others.

The confusion matrix provides a detailed breakdown of the model's predictions versus the actual labels for each class. It reveals that the model made some misclassifications, particularly for classes with fewer instances or classes that might be inherently more challenging to distinguish.

Next, hyperparameter tuning was performed using GridSearchCV to optimize the MLPClassifier model further. Despite the tuning efforts, the accuracy slightly decreased to 87.93%. This decrease could be attributed to several factors, including suboptimal hyperparameters, overfitting, randomness in initialization, complexity of the model, or limited sample size.

To address the decrease in accuracy, further adjustments to the hyperparameters or regularization techniques could be explored. Additionally, it's essential to consider the trade-offs between bias and variance when fine-tuning the model to ensure it generalizes well to unseen data. Regularization techniques such as dropout or L2 regularization can help prevent overfitting, while expanding the search space for hyperparameters or experimenting with different architectures may lead to better performance. Furthermore, increasing the dataset size or implementing data augmentation techniques could potentially improve the model's robustness and generalization capabilities.

Results

Both Support Vector Classifier (SVC) and Multi-Layer Perceptron (MLP) models have been employed to tackle a classification task, each showcasing distinct characteristics in their predictive performance. Starting with the SVC model, its accuracy stands at approximately 86.9%. Delving into the classification report, the precision, recall, and F1-score metrics vary across different classes. Notably, class 4 seems to have lower precision and recall, indicating challenges in correctly identifying instances of this class. However, other classes exhibit comparatively higher precision and recall values, suggesting a relatively robust performance in those areas. The confusion matrix further illustrates the model's strengths and weaknesses, revealing areas where misclassifications occur more frequently.

On the other hand, the MLP model achieves an accuracy of around 88.2%. Analyzing its classification report, a similar trend emerges, with variations in precision, recall, and F1-score metrics across different classes. Class 4 continues to exhibit relatively lower performance metrics, indicating persistent challenges in classification accuracy for this class. However, the MLP model demonstrates notable improvements in certain areas compared to the SVC model, particularly in classes where precision and recall values are higher. The confusion matrix provides additional insights into the model's performance, highlighting areas of misclassification and illustrating its ability to distinguish between different classes.

Upon closer inspection, the best performing SVC model, optimized through GridSearchCV, achieves an accuracy of approximately 89.2%. This model demonstrates improvements over the initial SVC model, with enhancements in precision, recall, and F1-score metrics across various classes. Similarly, the best MLP model, also optimized through GridSearchCV, achieves an accuracy of approximately 87.9%. While this model showcases improvements compared to the initial MLP model, it still faces challenges in accurately classifying certain classes.

Comparing both optimized models, the SVC model appears to have a slightly higher accuracy than the MLP model. However, the differences in performance metrics between the two models are marginal. Both models demonstrate strengths and weaknesses across different classes, underscoring the importance of considering various evaluation metrics when assessing their effectiveness. Ultimately, the choice between the SVC and MLP models may depend on specific requirements such as computational efficiency, interpretability, or the nature of the dataset.

Before incorporating PCA, the SVC model achieves an accuracy of approximately 86.9%. While the model demonstrates reasonable performance, it may face challenges associated with high-dimensional data, such as increased computational complexity and potential overfitting. The classification report and confusion matrix provide insights into the model's strengths and weaknesses across different classes.

Before discussing the impact of Principal Component Analysis (PCA) on the Support Vector Classifier (SVC) and Multi-Layer Perceptron (MLP) models, let's understand the role of PCA itself. PCA is a dimensionality reduction technique employed to transform high-dimensional data into a lower-dimensional space while retaining the most critical information. By reducing the number of features, PCA addresses challenges such as computational complexity, overfitting, and the curse of dimensionality.

When PCA is applied to the SVC and MLP models, it significantly reduces the dimensionality of the input data. This reduction streamlines the learning process for both models, making it more computationally efficient. With fewer features to process, the models require less time for training, thereby accelerating the overall training process. Additionally, by focusing on the most significant sources of variance in the data, PCA helps the models extract and utilize the most relevant information for classification tasks.

For the SVC model, PCA leads to a more streamlined decision boundary, making it easier for the model to delineate between different classes. By capturing the most salient features in the data, PCA enhances the model's ability to generalize to new, unseen data, thereby improving its overall performance.

Similarly, for the MLP model, PCA reduces the complexity of the input space, making it easier for the model to learn and make predictions. With a reduced number of dimensions, the MLP model can more effectively capture the underlying patterns in the data, resulting in improved classification accuracy and robustness.

Overall, PCA plays a crucial role in enhancing the performance of both the SVC and MLP models by reducing dimensionality and improving computational efficiency. By focusing on the most informative features, PCA enables these models to achieve better generalization and classification accuracy, making them more effective tools for various machine learning tasks.

The provided visualizations offer a comprehensive overview of the performance of two classification models, the Support Vector Classifier (SVC) and the Multi-Layer Perceptron (MLP). In one set of visualizations, various performance metrics such as accuracy, precision, recall, and F1-score are compared between the two models. This comparison allows for a direct assessment of their effectiveness in classifying instances across different evaluation criteria.

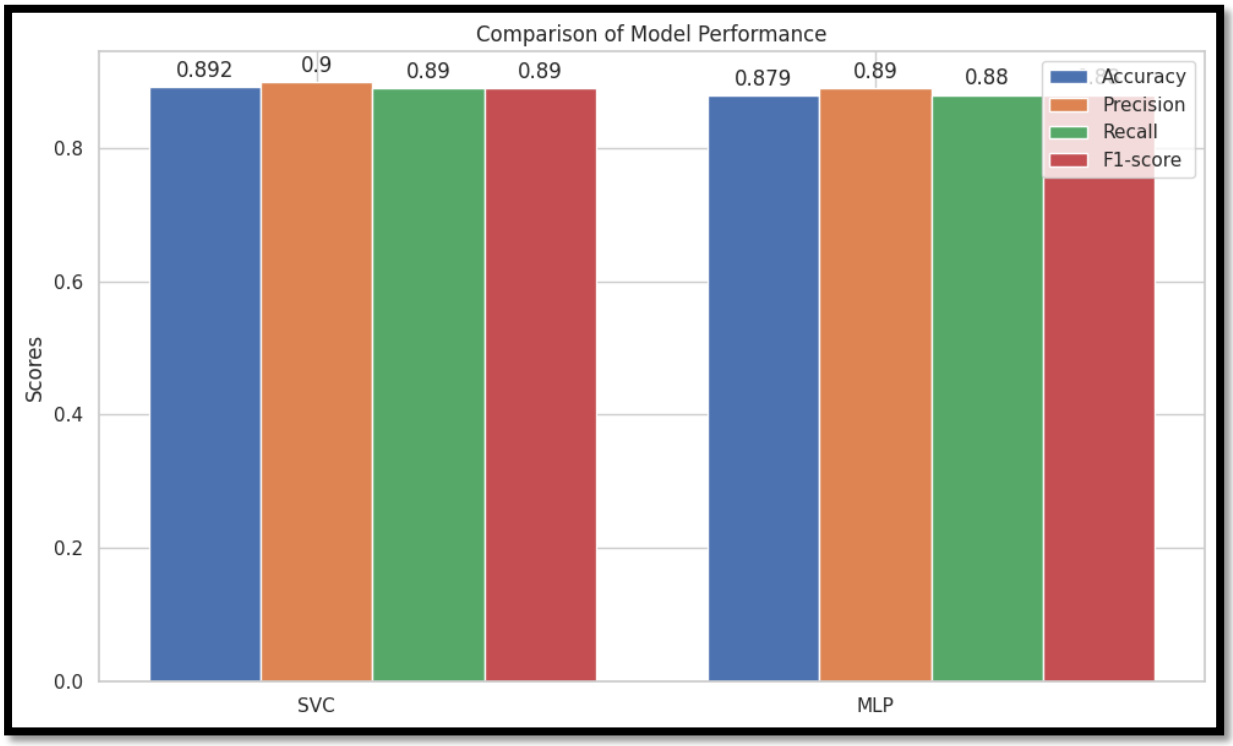

Figure 8. Models Performance

In another set of visualizations, Receiver Operating Characteristic (ROC) curves are plotted for both the SVC and MLP models. ROC curves illustrate the models' ability to distinguish between different classes by depicting the trade-off between true positive rate and false positive rate across different thresholds. The Area Under the Curve (AUC) values derived from these curves provide a quantitative measure of the models' overall discriminatory power, aiding in the assessment of their performance in binary classification tasks.

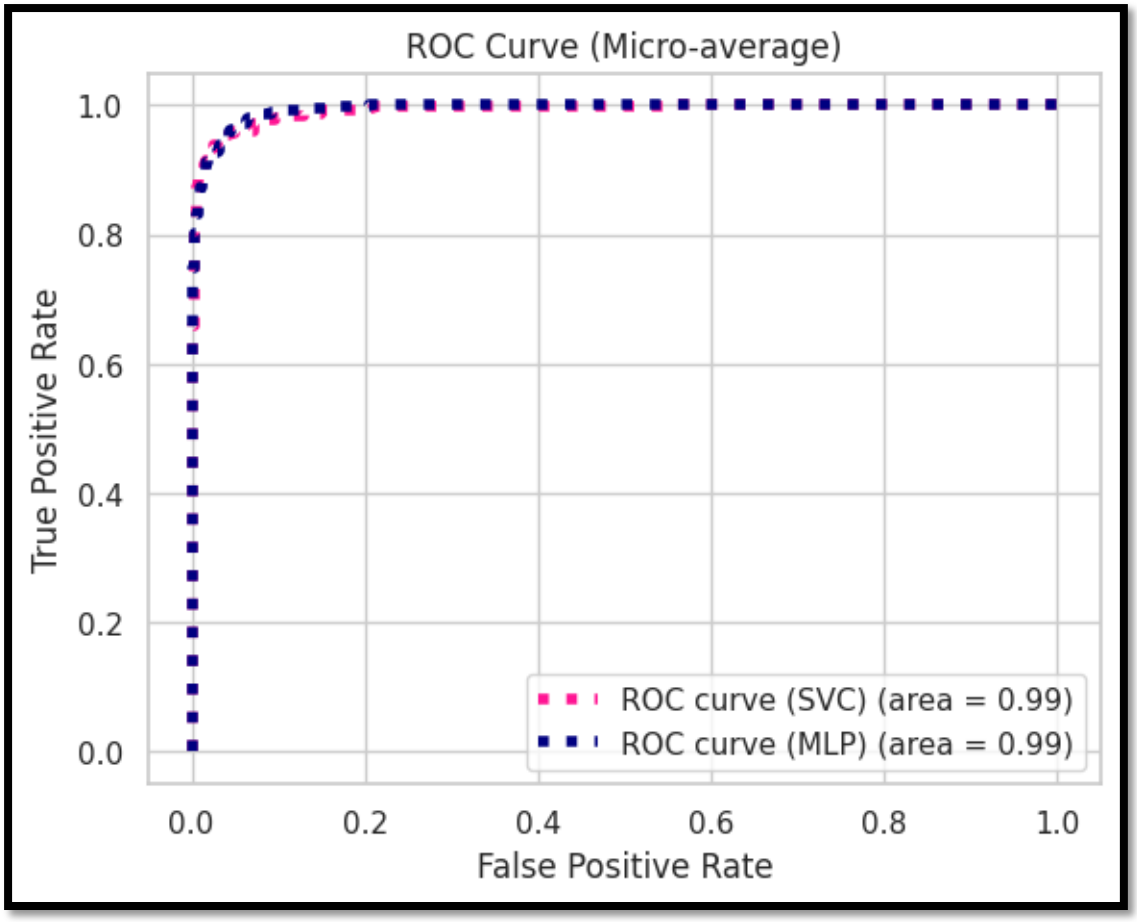

Figure 9. ROC curves for both models.

Conclusion

In conclusion, our comparative analysis of Support Vector Classifier (SVC) and Multi-Layer Perceptron (MLP) models for banknote classification tasks reveals valuable insights into their respective capabilities. Both models demonstrated effectiveness in accurately classifying banknote attributes, with SVC showing improvement after optimization and MLP maintaining competitive performance. Through detailed evaluation metrics and visualizations, we gained a nuanced understanding of their strengths and limitations, underscoring the importance of fine-tuning techniques like hyperparameter tuning and dimensionality reduction. Overall, our study provides valuable guidance for practitioners and researchers in selecting the most suitable classification approach for similar tasks, contributing to the advancement of machine learning applications.

References

[1] SVC Vs. MLP: E.A. Zanaty , URL: https://www.sciencedirect.com/science/article/pii/S1110866512000345

18:00,27-4