BIRZEIT UNIVERSITY

Department of Electrical & Computer Engineering
ENCS5141 - Intelligent Systems Laboratory

Case Study #2—Comparative Analysis of Classification Techniques: Support Vector Machines (SVM) and Multilayer Perceptron (MLP)

Prepared by: Mazen Batrawi
Instructor: Dr. Ismail Khater
Assistant: Eng. Hanan Awawdeh
Date: April 29, 2024

Abstract

The study meticulously compares the performance of Support Vector Machines (SVM) and Multilayer Perceptrons (MLP) on the banknotes dataset, employing robust methodologies such as grid search for hyperparameter optimization and Principal Component Analysis (PCA) for dimensionality reduction. SVM demonstrates its prowess in high-dimensional spaces, exhibiting versatility in handling both linear and non-linear classification tasks. However, the sensitivity of SVM to parameter choices poses a challenge, particularly in scenarios with numerous dimensions. Additionally, the interpretability of SVM models is questioned, especially in complex, high-dimensional settings. Conversely, Multilayer Perceptrons (MLP) exhibit remarkable flexibility in capturing intricate non-linear relationships within the data, making them suitable for tasks where decision boundaries are complex. However, MLP models are susceptible to overfitting, particularly when confronted with large parameter spaces, which can compromise their generalization performance. The study underscores the importance of grid search for optimizing hyperparameters, leading to notable improvements in model metrics like accuracy and precision. Furthermore, the application of PCA not only mitigates the curse of dimensionality but also enhances computational efficiency by reducing the dataset's dimensionality without significant loss of information. The comparative analysis provides valuable insights into the strengths and limitations of SVM and MLP models. By understanding these characteristics, practitioners can make well-informed decisions when selecting the most suitable approach for specific classification tasks. This study contributes significantly to the advancement of machine learning techniques and aids in practical decision-making for real-world classification challenges.

Contents

List of Figures

List of Tables

1 Motivation

Understanding the strengths and weaknesses of different classification methods is crucial for effectively tackling various data-driven problems. By comparing Support Vector Machines with Multilayer Perceptrons, we gain valuable insights into their capabilities and performance characteristics. SVMs excel at constructing intricate decision boundaries using kernel functions, while MLPs leverage neural networks to discern complex patterns within the data. Through a comparative analysis, we aim to discern scenarios where SVMs or MLPs outperform each other, aiding practitioners in selecting the most suitable approach for their specific classification tasks. This comparative exploration not only enhances our understanding of machine learning techniques but also equips us to make informed decisions when confronted with classification challenges in real-world scenarios.

2 Introduction

In this case study, we'll delve into the realm of classification techniques, specifically focusing on the comparative analysis between support vector machines and multilayer perceptron. Using the banknotes dataset as our testing ground, we'll embark on a journey to understand the nuances, strengths, and limitations of each algorithm. By meticulously studying their performance, we'll gain valuable insights into their applicability across various scenarios, equipping yourself with essential knowledge for tackling classification tasks effectively.

2.1 Dataset

The banknotes dataset, which contains over 24,000 carefully documented records, offers a thorough look into the complex realm of cash authenticity. All of the entries in this large collection are snapshots of individual banknotes taken under a variety of accessibility circumstances. With so much data available, researchers may examine the subtleties of money authenticity using 256 different features. These features capture a wide range of attributes and provide information on the finer points of the composition and appearance of each banknote.

Furthermore, the dataset goes above and beyond simple feature representation by providing related labels that enhance the analytical process. These labels give important background information. They identify the type of currency, which includes AUD, USD, CAD, and other currencies, as well as the denominations, which include little amounts like 10 and large amounts like 100. The dataset further clarifies each banknote's orientation by differentiating between front and back views, which are assigned values of 1 and 2, respectively.

Researchers have a unique opportunity to understand the complexities of money authentication through this vast data source. Analysts can make well-informed decisions and improve algorithms for currency authentication and categorization by exploring the relationship between features and labels. This allows them to find patterns, trends, and anomalies.

2.2 Grid Search

Grid search [1] is a method used in machine learning to systematically explore a predefined subset of hyperparameters for a given model. These hyperparameters, such as learning rates or tree depths, significantly influence a model's performance but are set before training. By exhaustively testing different combinations, grid search aims to find the optimal set that maximizes the model's performance.

To evaluate each combination, grid search commonly employs cross-validation, dividing the training data into subsets for training and validation. This helps to ensure robust performance assessment and prevents overfitting. Additionally, a chosen evaluation metric, such as accuracy or F1-score, guides the selection of the best-performing hyperparameter set.

While grid search is straightforward and effective, it can be computationally expensive, particularly with large datasets or complex hyperparameter spaces. Despite this, it remains a valuable tool in fine-tuning machine learning models, providing a systematic approach to optimizing performance by navigating the hyperparameter landscape.

2.3 Principal Component Analysis (PCA)

The curse of dimensionality presents challenges in high-dimensional datasets, causing overfitting, increased computation time, and reduced model accuracy. As dimensions increase, obtaining statistically significant results becomes exponentially harder due to the vast number of feature combinations.

To combat this, feature engineering techniques like feature selection and extraction are used. Principal Component Analysis [2], introduced by Karl Pearson in 1901, is a key method for dimensionality reduction. PCA transforms correlated variables into uncorrelated ones while maximizing variance in a lower-dimensional space. It's widely applied in exploratory data analysis and machine learning, uncovering variable interrelations without prior knowledge of target variables.

PCA's main goal is to reduce dataset dimensionality while retaining important patterns. By identifying a smaller set of variables, it preserves most sample information crucial for regression and classification tasks, enhancing model interpretability and computational efficiency. PCA is a valuable tool for simplifying complex datasets and optimizing machine learning workflows amidst the challenges of the curse of dimensionality.

3 Literature Review

Multilayer Perceptron and Support Vector Machines are widely-used machine learning algorithms for classification and regression tasks. While MLP belongs to the family of artificial neural networks, SVM is a supervised learning model with associated learning algorithms that analyze data for classification and regression analysis.

3.1 Multilayer Perceptron (MLP)

Multilayer Perceptron [3] is a class of feedforward artificial neural network consisting of multiple layers of nodes, each layer fully connected to the next layer. It is capable of learning non-linear models and can be used for both regression and classification tasks.

3.1.1 Training Methodology

MLP employs a backpropagation algorithm for training. It involves forward propagation of inputs through the network to generate predictions, followed by backward propagation of errors to update the network weights and minimize the loss function.

3.1.2 Main Hyperparameters

- **Hidden Layers** [4]: The number of hidden layers and the number of neurons in each layer.
- **Activation Function**: Functions applied to the output of each neuron to introduce non-linearity.
- **Learning Rate**: Rate at which the model adapts during training to minimize the loss function.
- **Batch Size**: Number of samples over which the loss function is averaged during training.
- **Solver**: Optimization algorithm used to update network weights during backpropagation (e.g., SGD, Adam).

3.1.3 Advantages

- **Non-linearity**: MLP can capture complex non-linear relationships in data.
- **Flexibility**: Can be used for various tasks including classification, regression, and pattern recognition.
- **Scalability**: Can handle large datasets and high-dimensional feature spaces.

3.1.4 Limitations

- **Overfitting**: Prone to overfitting, especially with large numbers of parameters.

- **Sensitivity to Initialization**: Performance can depend on the initial weights and biases.
- **Computational Complexity**: Training time can be high, particularly for large networks and datasets.

3.2 Support Vector Machines (SVM)

Support Vector Machines [5] are supervised learning models used for classification and regression analysis. They find the hyperplane that best separates classes in the feature space.

3.2.1 Training Methodology

SVM seeks to find the hyperplane that maximizes the margin between classes in the feature space. It can be linear or non-linear depending on the kernel function used.

3.2.2 Main Hyperparameters

- **Kernel Type** [6]: Specifies the kernel function used for non-linear classification (e.g., linear, polynomial, radial basis function).
- **Regularization Parameter (C)**: Controls the trade-off between maximizing the margin and minimizing classification errors.
- **Kernel Coefficient (Gamma)**: Parameter for non-linear kernels, affecting the influence of training samples on the decision boundary.

3.2.3 Advantages

- **Effective in High-Dimensional Spaces**: Can handle datasets with many features.
- **Versatility**: Suitable for both linear and non-linear classification and regression tasks.
- **Memory Efficiency**: Uses a subset of training points (support vectors) for decision function.

3.2.4 Limitations

- **Sensitivity to Parameters**: Performance can be sensitive to the choice of kernel and regularization parameters.
- **Lack of Interpretability**: Decision boundaries can be difficult to interpret, especially in high-dimensional spaces.

- **Training Time**: Training time can be relatively long for large datasets, especially with non-linear kernels.

3.3 Key Differences

- **Model Type**: MLP is a type of artificial neural network, while SVM is a discriminative model.

- **Training Approach**: MLP uses backpropagation for training, while SVM optimizes a margin-based objective function.

- **Complexity**: MLP tends to have more parameters and is more computationally intensive compared to SVM.

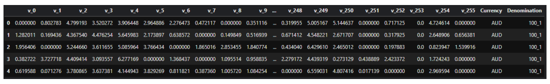
- **Interpretability**: SVM typically provides clearer decision boundaries compared to MLP, which can be advantageous in some applications.

- **Handling of Non-linearity**: While both can handle non-linear data, MLP inherently captures non-linearities through its architecture, whereas SVM requires explicit kernel specifications for non-linear transformations.

4 Procedure and Discussion

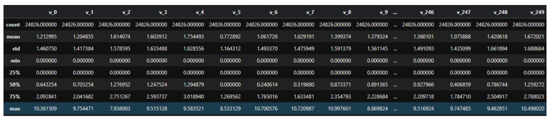
4.1 Loading The BankNotes Dataset

In this step, the banknotes dataset was cloned from the github repository and loaded using the provided code snippet in the Jupyter Notebook file. Figures 4.1a and 4.1b below show the dataset's information.



	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	...	v_248	v_249	v_250	v_251	v_252	v_253	v_254	v_255	Currency	Denomination
0	0.000000	0.802783	4.799193	3.520272	3.906448	2.964886	2.276473	0.472117	0.000000	0.351116	...	0.319955	5.005167	5.144637	0.000000	0.717125	0.0	4.724614	0.000000	AUD	100_1
1	1.282011	0.169436	4.367540	4.476254	5.645983	2.173897	0.638572	0.000000	0.149849	0.516939	...	0.671412	4.548221	2.671707	0.000000	0.317925	0.0	2.648906	0.656381	AUD	100_1
2	1.956406	0.000000	5.244660	3.611655	5.085964	3.766434	0.000000	1.865016	2.853455	1.840774	...	0.434040	6.429610	2.465012	0.000000	0.197883	0.0	0.823947	1.539916	AUD	100_1
3	0.382722	3.727718	4.409414	3.093557	6.277169	0.000000	1.368437	0.000000	1.095514	0.958835	...	2.279172	4.439319	0.273129	0.438889	2.423372	0.0	1.724243	0.000000	AUD	100_1
4	0.619588	0.071276	3.780865	3.637381	4.144943	3.829269	0.811821	0.387360	1.005720	1.084254	...	0.000000	6.559031	4.807416	0.017139	0.000000	0.0	2.969594	0.000000	AUD	100_1

(a) Dataset Header



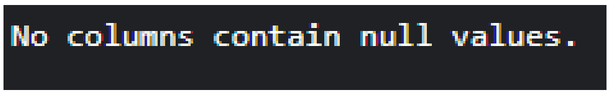
	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	...	v_246	v_247	v_248	v_249
count	24826.000000	24826.000000	24826.000000	24826.000000	24826.000000	24826.000000	24826.000000	24826.000000	24826.000000	24826.000000	...	24826.000000	24826.000000	24826.000000	24826.000000
mean	1.212995	1.204855	1.614074	1.603912	1.754493	0.772892	1.061728	1.029191	1.399374	1.379324	...	1.360101	1.075888	1.420618	1.672021
std	1.460750	1.417384	1.578595	1.633488	1.828556	1.164312	1.493370	1.475949	1.591379	1.561145	...	1.491093	1.435099	1.661894	1.688684
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000
50%	0.643254	0.703254	1.276952	1.247524	1.294879	0.000000	0.240614	0.319880	0.873371	0.891365	...	0.927966	0.406859	0.786744	1.259272
75%	2.092841	2.041682	2.751267	2.593737	3.018940	1.269562	1.765016	1.633481	2.354793	2.228684	...	2.209718	1.784710	2.504917	2.788023
max	10.361509	9.754471	7.938003	9.515128	9.583521	8.533129	10.700576	10.720987	10.997601	8.869824	...	9.516924	9.747485	9.462851	10.498020

(b) Dataset Description

Figure 4.1: Dataset Information

4.2 Checking for null values and unimportant data

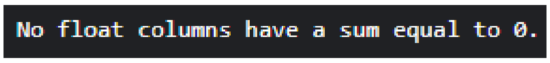
We started exploring the dataset by checking for null values in the dataset as shown in figure 4.2 below. We can see that the dataset has no null values.



```
No columns contain null values.
```

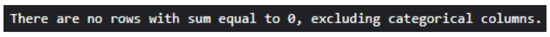
Figure 4.2: Checking for null values

After that, we checked for any columns that has a zero sum, or any rows with zero sum, to remove them. Figures 4.3a and 4.3b below show that there are no rows or columns that has a zero summation.



```
No float columns have a sum equal to 0.
```

(a) Columns with zero sum



```
There are no rows with sum equal to 0, excluding categorical columns.
```

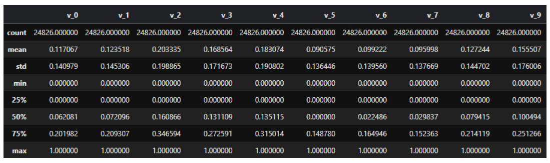
(b) Rows with zero sum

Figure 4.3: Columns and rows with zero sum

After that, the columns we want to predict (**Currency**, **Denomination**, and **Orientation**) were separated each in a single column, and another column was added (**currency_denomination_orientation**), which combines all of the outputs together.

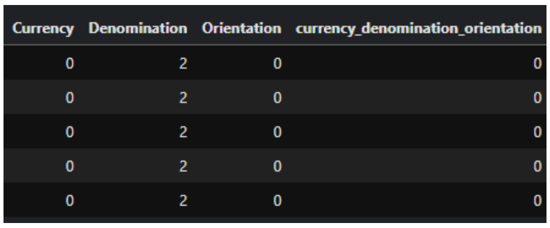
4.3 Normalizing and Encoding the data

The process of normalizing the numerical data and encoding the categorical is crucial for any model. Figures 4.4a and 4.4b below shows the data after applying encoding and normalization. We used **MinMaxScaler** for normalizing the numerical data between 0 and 1, and **LabelEncoder** to encode the categorical data. Using **OneHotEncoding** would create a lot of columns because we have a lot of classes, so **LabelEncoder** was a better option.



	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
count	24826.000000	24826.000000	24826.000000	24826.000000	24826.000000	24826.000000	24826.000000	24826.000000	24826.000000	24826.000000
mean	0.117067	0.123518	0.203335	0.168564	0.183074	0.090575	0.099222	0.095998	0.127244	0.155507
std	0.140979	0.145306	0.198865	0.171673	0.190802	0.136446	0.139560	0.137669	0.144702	0.176006
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.062081	0.072096	0.160866	0.131109	0.135115	0.000000	0.022486	0.029837	0.079415	0.100494
75%	0.201982	0.209307	0.346594	0.272591	0.315014	0.148780	0.164946	0.152363	0.214119	0.251266
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

(a) Normalized data



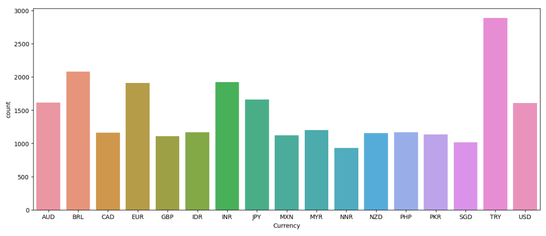
Currency	Denomination	Orientation	currency_denomination_orientation
0	2	0	0
0	2	0	0
0	2	0	0
0	2	0	0
0	2	0	0

(b) Categorized data

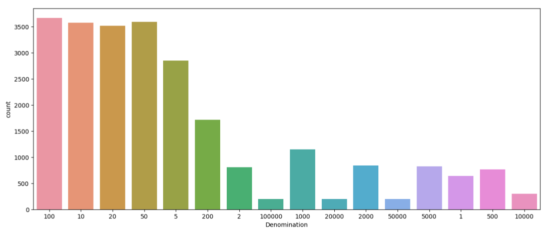
Figure 4.4: Normalizing and Encoding

4.4 Data Visualization

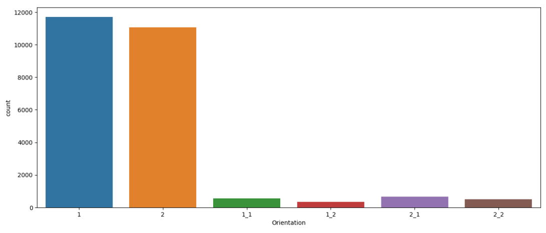
Figure 4.5 below shows the distribution of each of the outputs classes.



(a) Currency classes distributions and count



(b) Denomination classes distributions and count

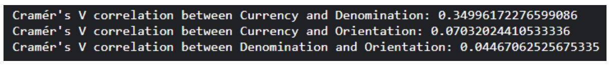


(c) Orientation classes distributions and count

Figure 4.5: Output classes distribution and count

4.5 Outputs Correlation

Keeping the other outputs while predicting one of them might affect the process. So, we checked the correlation between the outputs to see if they give important information about each other. Figure 4.6 shows the correlation value between the outputs. We can see that values are low, so we will not consider any of the outputs while training for the prediction of any output.



```
Cramér's V correlation between Currency and Denomination: 0.34996172276599086
Cramér's V correlation between Currency and Orientation: 0.07032024410533336
Cramér's V correlation between Denomination and Orientation: 0.04467062525675335
```

Figure 4.6: Correlation between the outputs

4.6 Hyperparamter tuning

The grid in 4.1 shows the parameters grid that will be used for hyperparameter tuning for MLP, while the grid in 4.2 shows the parameters grid that will be used for hyperparameter tuning for SVC.

4.6.1 MLP

```
param_grid = {
    'hidden_layer_sizes': [(100,), (100, 50), (50, 50)],
    'activation': ['relu', 'tanh'],
    'solver': ['adam', 'sgd'],
    'alpha': [0.01, 0.1, 1, 10],
}
```

Listing 4.1: Parameter Grid for MLP

4.6.2 SVC

```
param_grid = {
    'C': [0.01, 0.1, 1],
    'gamma': [1, 0.1, 0.01],
    'kernel': ['rbf', 'linear', 'poly']
}
```

Listing 4.2: Parameter Grid for SVC

4.7 Evaluation Metrics Before PCA

4.7.1 MLP Evaluation

In Table 1, we can see the performance metrics of MLP before grid search. Table 2 shows the performance metrics of MLP after grid search.

Table 1: Performance Metrics of MLP Before Grid Search

Metric	MLP Before Grid Search			
	Currency	Denomination	Orientation	All
F1 Score	0.95	0.92	0.94	0.93
Precision	0.95	0.92	0.94	0.93
Recall	0.95	0.92	0.94	0.93
Accuracy	0.95	0.92	0.94	0.93

The MLP model exhibits consistently high performance across multiple evaluation metrics including F1 Score, Precision, Recall, and Accuracy, with scores ranging from 0.92 to 0.95. These results indicate robust performance across various criteria and categories. Specifically, the F1 Score, which balances precision and recall, reflects a strong ability to correctly identify positive instances and capture all positive instances. Precision, Recall, and Accuracy metrics also demonstrate high values across all categories, confirming the model's proficiency in classification and overall accuracy in prediction. Further optimization through grid search could potentially enhance performance by fine-tuning hyperparameters to better suit the dataset and improve generalization.

Table 2: Performance Metrics of MLP After Grid Search

Metric	MLP After Grid Search			
	Currency	Denomination	Orientation	All
F1 Score	0.96	0.94	0.94	0.96
Precision	0.96	0.94	0.94	0.96
Recall	0.96	0.94	0.94	0.96
Accuracy	0.96	0.94	0.94	0.96

Before grid search optimization, the model's performance metrics showed room for improvement across all evaluation categories. Following grid search, there were noticeable enhancements in all metrics, particularly in the "All" category, which combines the columns before it in the data frame, where the F1 Score rose from 0.93 to 0.96, as we can see from tables 1 and 2. This signifies a significant improvement in the model's overall ability to balance precision and recall across the entire dataset. Additionally, consistent improvements were observed in Precision, Recall, and Accuracy metrics across individual evaluation categories, reinforcing the comprehensive refinement achieved through grid search optimization. Overall, the optimization process resulted in a more robust and effective model performance across all aspects of the dataset.

The tables below provide a detailed comparison of the training time and memory usage for MLP models before and after grid search, showcasing the varying resource requirements across different feature categories including Currency, Denomination, Orientation, and All. Table 3 presents the time comparison for MLP before grid search, while Table 4 presents the time comparison for MLP after grid search.

Table 3: Time Comparison for MLP Before Grid Search

Metric	Currency	Denomination	Orientation	All
Training Time	63.6 seconds	54.9 seconds	45.2 seconds	107.5 seconds
Memory Usage	14.9 MB	14.8 MB	30.6 MB	25.4 MB

Table 4: Time Comparison for MLP After Grid Search

Metric	Currency	Denomination	Orientation	All
Training Time	64.1 seconds	65.9 seconds	50.1 seconds	109.8 seconds
Memory Usage	1.5 MB	26.4 MB	6.3 MB	197.9 MB

We can notice from tables 3 and 4 that the column (**All**) combining all of the 3 outputs (**Currency**, **Denomination**, and **Orientation**) uses more time and memory while fitting the data in the model.

4.7.2 SVC Evaluation

In Table 5, we can see the performance metrics of SVC before grid search. Table 6 shows the performance metrics of SVC after grid search.

Table 5: Performance Metrics of SVC Before Grid Search

Metric	SVC Before Grid Search			
	Currency	Denomination	Orientation	All
F1 Score	0.96	0.90	0.91	0.95
Precision	0.96	0.90	0.91	0.95
Recall	0.96	0.90	0.90	0.95
Accuracy	0.96	0.90	0.90	0.95

Table 6: Performance Metrics of SVC After Grid Search

Metric	SVC After Grid Search			
	Currency	Denomination	Orientation	All
F1 Score	0.96	0.94	0.95	0.95
Precision	0.96	0.94	0.95	0.95
Recall	0.96	0.94	0.95	0.95
Accuracy	0.96	0.94	0.95	0.95

We can see from tables 5 and 6 the significant improvement in the metrics of the columns **Denomination** and **Orientation** after the grid search, while maintaining high values for the **Currency** and **All** columns. While the result between the MLP and the SVC are nearly the same, the column **All** shows disimprovement after the grid search, where it showed 0.96 in all of the metrics in MLP, and 0.95 in SVC. Testing more hyperparameters can improve the overall results for the SVC.

Table 7: Time Comparison for SVC Before Grid Search

Metric	Currency	Denomination	Orientation	All
Training Time	5.6 seconds	13.2 seconds	18.9 seconds	7.8 seconds
Memory Usage	411.6 MB	2.2 MB	46.7 MB	220.9 MB

Table 8: Time Comparison for SVC After Grid Search

Metric	Currency	Denomination	Orientation	All
Training Time	6.8 seconds	12.7 seconds	13.7 seconds	7.9 seconds
Memory Usage	0.6 MB	0.4 MB	28.7 MB	6.1 MB

The tables above provide a detailed comparison of the training time and memory usage for SVC models before and after grid search, showcasing the varying resource requirements across different feature categories including Currency, Denomination, Orientation, and All. Table 7 presents the time comparison for SVC before grid search, while Table 8 presents the time comparison for SVC after grid search.

Table 9: Best parameters for MLP before PCA

Parameter	Currency	Denomination	Orientation	All
hidden_layer_sizes	(100, 50)	(100,)	(100, 50)	(100,)
activation	relu	relu	relu	tanh
solver	adam	adam	adam	adam
alpha	0.1	0.01	0.1	0.01

Table 10: Best parameters for SVC before PCA

Parameter	Currency	Denomination	Orientation	All
C	0.01	0.01	0.01	1
gamma	1	1	1	1
kernel	poly	poly	poly	linear

We can see from tables 9 and 10 the best hyperparameters we got from the grid search. Another notable thing is that the SVC is more time consuming than the MLP, because of the heavier computations it makes during the process of learning and predicting.

4.8 Evaluation Metrics After PCA

In this section, we will redo the same process, but with applying dimensionality reduction for the data, using PCA. We begin the process by normalizing and encoding the data, then applying PCA with **n_components**=100 (Lower values were tried but the results got worse). A notable thing is that the run process was faster than the run without PCA, because we reduced the size of the data.

4.8.1 MLP Evaluation

In Table 11, we can see the performance metrics of MLP before grid search. Table 12 shows the performance metrics of MLP after grid search.

Table 11: Performance Metrics of MLP Before Grid Search with PCA

Metric	MLP Before Grid Search with PCA			
	Currency	Denomination	Orientation	All
F1 Score	0.95	0.91	0.94	0.92
Precision	0.95	0.91	0.94	0.92
Recall	0.95	0.91	0.94	0.92
Accuracy	0.95	0.91	0.94	0.92

Table 12: Performance Metrics of MLP After Grid Search with PCA

Metric	MLP After Grid Search with PCA			
	Currency	Denomination	Orientation	All
F1 Score	0.96	0.94	0.95	0.95
Precision	0.96	0.94	0.95	0.95
Recall	0.96	0.94	0.95	0.95
Accuracy	0.96	0.94	0.95	0.95

We can see from table 11 that the results were the same as without PCA for the columns and **Currency** and **Orientation**, while it got less by 0.01 for both of the **Denomination** and **All** columns. And from table 12, we can see that the columns **All** and **Orientation** increased every metric by 0.01, while the others kept their values the same as without PCA.

Table 13: Time Comparison for MLP Before Grid Search with PCA

Metric	Currency	Denomination	Orientation	All
Training Time	41.1 seconds	55 seconds	65.1 seconds	93.3 seconds
Memory Usage	12.2 MB	33.5 MB	2.1 MB	24 MB

Table 14: Time Comparison for MLP After Grid Search with PCA

Metric	Currency	Denomination	Orientation	All
Training Time	66.8 seconds	65.1 seconds	37.3 seconds	86.1 seconds
Memory Usage	105 MB	2.1 MB	49.2 MB	25.1

The tables above provide a detailed comparison of the training time and memory usage for MLP models before and after grid search, showcasing the varying resource requirements across different feature categories including Currency, Denomination, Orientation, and All. Table 13 presents the time comparison for MLP before grid search, while Table 14 presents the time comparison for MLP after grid search.

4.8.2 SVC Evaluation

In Table 15, we can see the performance metrics of SVC before grid search. Table 16 shows the performance metrics of SVC after grid search.

Table 15: Performance Metrics of SVC Before Grid Search with PCA

Metric	SVC Before Grid Search with PCA			
	Currency	Denomination	Orientation	All
F1 Score	0.95	0.87	0.89	0.94
Precision	0.95	0.87	0.89	0.95
Recall	0.95	0.87	0.88	0.94
Accuracy	0.95	0.87	0.88	0.94

Table 16: Performance Metrics of SVC After Grid Search with PCA

Metric	SVC After Grid Search with PCA			
	Currency	Denomination	Orientation	All
F1 Score	0.97	0.95	0.95	0.94
Precision	0.97	0.95	0.95	0.95
Recall	0.97	0.95	0.95	0.94
Accuracy	0.97	0.95	0.95	0.94

We can see from table 15 that almost all of the metrics went lower than the ones without PCA, but when we applied grid search, the values were better thatn the ones without PCA, which shows the efficiency of the grid search along with dimensionality reduction techniques when used correctly.

Table 17: Time Comparison for SVC Before Grid Search with PCA

Metric	Currency	Denomination	Orientation	All
Training Time	3.7 seconds	9.5 seconds	11.7 seconds	4.9 seconds
Memory Usage	9.3 MB	15.2 MB	210.5 MB	0.1 MB

Table 18: Time Comparison for SVC After Grid Search with PCA

Metric	Currency	Denomination	Orientation	All
Training Time	8.7 seconds	12.2 seconds	10.3 seconds	4.7 seconds
Memory Usage	0.4 MB	6.7 MB	300 MB	0.1 MB

The tables above provide a detailed comparison of the training time and memory usage for SVC models before and after grid search, showcasing the varying resource requirements across different feature categories including Currency, Denomination, Orientation, and All. Table 17 presents the time comparison for SVC before grid search, while Table 18 presents the time comparison for SVC after grid search.

Table 19: Best parameters for MLP after PCA

Parameter	Currency	Denomination	Orientation	All
hidden_layer_sizes	(100, 50)	(100, 50)	(100,)	(100,)
activation	relu	relu	tanh	tanh
solver	adam	adam	adam	adam
alpha	0.1	0.1	0.01	0.01

Table 20: Best parameters for SVC after PCA

Parameter	Currency	Denomination	Orientation	All
C	1	0.1	0.1	1
gamma	1	1	1	1
kernel	poly	poly	poly	linear

We can see from tables 19 and 20 the best hyperparameters we got from the grid search with PCA. Another notable thing is that despite that the SVC is more time consuming than the MLP, the dimensionality reduction made the process faster.

5 Conclusion

In conclusion, the comparative study between Support Vector Machines and Multilayer Perceptrons on the banknotes dataset sheds light on the nuanced strengths and limitations of these two popular machine learning approaches in classification tasks.

SVM demonstrates its versatility in high-dimensional spaces, excelling in both linear and non-linear classification tasks. However, its sensitivity to parameter choices, particularly in scenarios with numerous dimensions, poses a challenge. Moreover, the interpretability of SVM models in complex, high-dimensional settings remains questionable.

On the other hand, MLP models exhibit remarkable flexibility in capturing intricate non-linear relationships within the data, making them suitable for tasks with complex decision boundaries. However, they are prone to overfitting, especially when faced with large parameter spaces, potentially compromising their generalization performance.

The study underscores the significance of employing robust methodologies such as grid search for hyperparameter optimization, leading to notable improvements in model metrics such as accuracy and precision. Grid search enables practitioners to fine-tune parameters effectively, enhancing the overall performance of both SVM and MLP models.

Furthermore, the application of Principal Component Analysis proves beneficial in mitigating the curse of dimensionality and enhancing computational efficiency by reducing the dataset's dimensionality without significant loss of information. In particular, PCA demonstrates its efficacy in improving the performance of Support Vector Machines, enabling faster runtime and achieving better results.

Overall, while Support Vector Machines and Multilayer Perceptrons excel in certain aspects of classification tasks, they also exhibit limitations in others. By understanding these characteristics, practitioners can make well-informed decisions when selecting the most suitable approach for specific classification challenges, ultimately contributing to the advancement of machine learning techniques and aiding in practical decision-making for real-world applications.

References

[1] GeeksforGeeks, “Grid searching from scratch using python,” Mar 2024, accessed on 28-04-2024. [Online]. Available: https://www.geeksforgeeks.org/grid-searching-from-scratch-using-python/

[2] Geeksforgeeks, “Principal component analysis(pca),” Dec 2023, accessed on 28-04-2024. [Online]. Available: https://www.geeksforgeeks.org/principal-component-analysis-pca/

[3] GeeksForGeeks, “Multi-layer perceptron learning in tensorflow,” Nov 2021, accessed on 28-04-2024. [Online]. Available: https://www.geeksforgeeks.org/multi-layer-perceptron-learning-in-tensorflow/

[4] Sklearn, accessed on 28-04-2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

[5] GeeksGorgeeks, “Support vector machine (svm) algorithm,” Jun 2023, accessed on 28-04-2024. [Online]. Available: https://www.geeksforgeeks.org/support-vector-machine-algorithm/

[6] S. Learn, “Sklearn.svm.svc,” accessed on 28-04-2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html