SAPIENZA
UNIVERSITÀ DI ROMA

Report

Machine Learning Homework 1

Professor:
Luca Iocchi

Student:
De Sio Ilaria - 2064970

Academic Year 2023/2024

Contents

1 Introduction

This is my report for the first Machine Learning Homework, about classification of 10 classes. I will explain the assigned datasets initially (See Section 1.1), with their specifications and associated goals (See Section 1.2).
In the following sections, I will provide an overview of the preprocessing activities undertaken (See Section 3). This will be followed by a comparative analysis of the results, with an evaluation based on various metrics,algorithms and scaling techniques applied to check their impact on performance (See Sections 6 and 5), and also I decided to provide a Training Time Comparison in the final part 7 .

1.1 Datasets

We have two Datasets, each containing N *samples*, where the main difference is the size of the *input space* d.

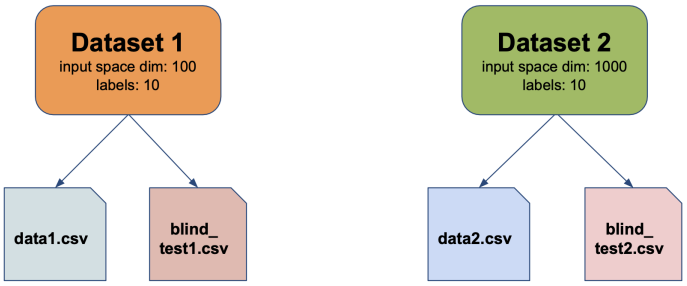


Figure 1: Datasets

The elements Vi belonging to the X column are feature vectors that represent your input data. The Y column contains the associated labels C_k (k=0,...,9).
Each line i in column X is a feature vector structured as:

$$[V_1^i, V_2^i, V_3^i, V_4^i, \ldots, V_j^i, \ldots, V_d^N]$$

where d is the size of the feature vector (100 for Dataset 1 and 1000 for Dataset 2) and i = [1,...,N] where N represents the total number of samples.

1.2 Assigned Task

The assignment involves a classification task, where in the primary objective is to implement and describe a solution. This includes detailing the data preprocessing methods employed and elucidating all other critical aspects of the process. The task further necessitates a comparative analysis of the models utilized, accompanied by an evaluation of their performance in terms of computational time and relevant metrics.

2 Data Acquisition

To acquire the data in the first place I have used the built-in function of panda pd.read csv to read the given .csv files. Since the newly extracted data were interpreted as strings, it was necessary to apply a regex defined this way in order to display them as a list of floats.

	x	y
0	[0.9942099, 0.33943707, 0.0, 0.48194316, 4.104...	6
1	[0.0, 1.4159914, 0.0, 2.8857417, 0.0, 0.0, 0.0...	9
2	[0.0, 1.3204838, 0.0, 2.8144786, 0.832429, 0.0...	9
3	[0.4596575, 1.2021416, 0.0, 0.0, 2.2813082, 0....	4
4	[0.02370751, 0.0, 0.0, 1.0069093, 0.0, 0.0, 0....	1

Figure 2: Structure of samples

3 Data Preprocessing

In the initial phase of the project, the focus is on the critical step of data preprocessing to ensure the reliability and quality of our results. This process involved initially meticulously analyze the distribution of the data to gain a comprehensive understanding of its underlying structure, and in the second part to allows the user to choose between two different methods: Z-score Standardization or Min Max Normalization, to reduce the features to a common scale, ensuring that the data is optimally formatted .

3.1 Data Exploration

For each dataset, is possible to visualize the distribution of classes to assess the balance within our variables of interest. The following bar chart illustrates the frequency of each class, providing a clear representation of the dataset's composition. The bar chart reveals a uniform distribution across the

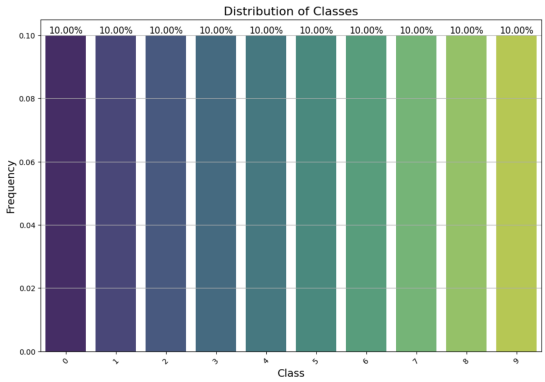
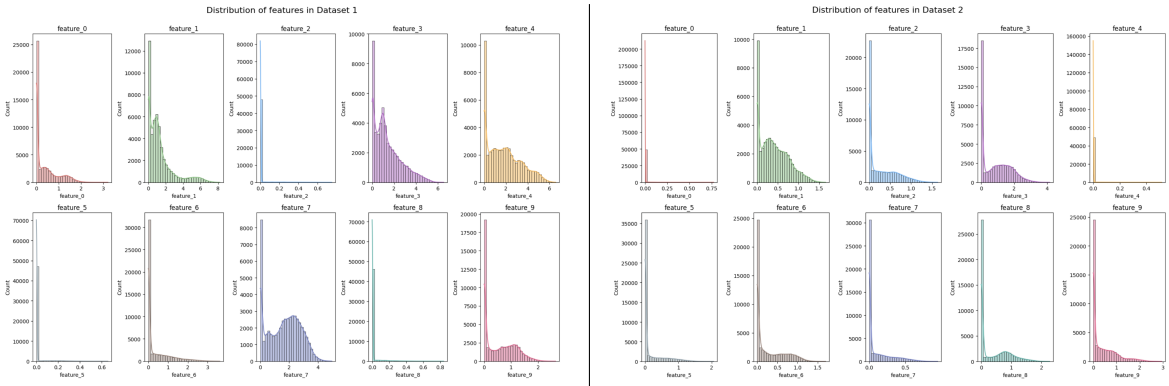


Figure 3: Distribution of Classes (same for each dataset)

classes, with each class accounting for exactly 10% of the dataset. This uniformity suggests that the dataset is well-balanced, which is advantageous for modeling as it implies that each class is equally represented, reducing the potential bias for a particular class in classification modeling.

Next I proceeded to analyze the distribution of values of the first 10 features, I chose to focus on these because in a Dataset with a large number of features a unique graph can be confusing, displaying them in this way can simplify visualization for the purpose of analysis.



(a) Dataset 1 (b) Dataset 2

Figure 4: Distribution of features

3.2 Z-score Standardization

Z-score standardization is a scaling technique provided by scikit-learn's *StandardScaler* to normalize the features of a dataset. Each feature is transformed to have a mean (μ) of zero and a standard deviation (σ) of one. The process follows the Z-score formula:

$$Z = \frac{(X - \mu)}{\sigma} \tag{1}$$

where X is the original feature value. This technique is especially beneficial for features that follow a normal distribution, as it aligns them to a common scale, allowing algorithms that are sensitive to the feature scale, such as Support Vector Machines and K-Nearest Neighbors, to perform better.

3.3 Min-Max Normalization

MinMaxScaler is another feature scaling method implemented in scikit-learn. This approach rescales the range of features to fit within a specified range. The formula used for Min-Max scaling is:

$$X_{\text{scaled}} = \frac{(X - X_{\min})}{(X_{\max} - X_{\min})} \tag{2}$$

where $X_{\min}$ and $X_{\max}$ represent the minimum and maximum values of the feature, respectively. Min-Max scaling is particularly sensitive to outliers and is often chosen when the extreme values of the features are known and carry important information. It ensures that all features contribute equally to the result of distance-based algorithms.

4 Split Data

In the process of splitting the Datasets, it was imperative to ensure a consistent and reproducible partitioning of data into training and testing sets. To achieve this, allocating 33% of the data for testing, which aligns with standard practices for maintaining a balance between training and validation. This is a fundamental aspect since the splitting size does affect the final performance of the model. Another crucial aspect of this step was the implementation of a *random state* set to 42.

$$\big((33500, 1000), (16500, 1000), (33500,), (16500,)\big) \tag{3}$$

5 Evaluation Metrics

In the process of assessing the performance of the model, I have considered a variety of metrics to gain a comprehensive understanding of its effectiveness. These metrics, each offering unique insights into different aspects of the model's performance, have been analyzed to extract the most pertinent information relevant to the specific problem at hand. By doing so, it allows for a nuanced understanding of how well the model performs in various scenarios, ensuring that the conclusions drawn are well-aligned with the unique requirements and challenges of the problem being addressed. These are the evaluation metrics I decided to use, in the sections below you can elaborate on them with a small introductory section on what each metric represents and the formula associated with it.

5.1 Confusion Matrix

The confusion matrix is a table used to describe the performance of a classification model on a set of test data for which the true values are known. It compares the actual target values with those predicted by the machine learning model. The key components of a confusion matrix are:

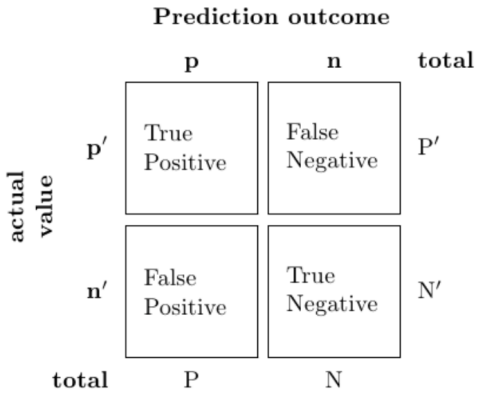


Figure 5: Confusion Matrix definition

5.2 Accuracy

This metric is critically important as it indicates the proportion of correctly predicted observations to the total observations. The formula for accuracy is derived from the error rate and is given as follows:

$$\text{Error Rate} = \frac{\text{Errors}}{\text{Instances}} = \frac{\text{FN} + \text{FP}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4}$$

$$\text{Accuracy} = 1 - \text{Error Rate} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{5}$$

where: TP = True Positives, FP = False Positives, TN = True Negatives, FN = False Negatives.

5.3 Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false positive rate. It is defined as:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} \tag{6}$$

5.4 Recall

Recall, also known as sensitivity, is the ratio of correctly predicted positive observations to all observations in the actual class. It is defined as:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \tag{7}$$

5.5 F1-Score

F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is between [0, 1]. It tells how precise your classifier is (how many instances it classifies correctly), as well as how robust it is. High precision but lower recall, gives you an extremely accurate, but it then misses a large number of instances that are difficult to classify. The greater the F1 Score, the better is the performance of our model. Mathematically, it can be expressed as :

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

6 Models Performances

After the first phase of Preprocessing I have chosen 6 different models to train and test. Each model has been evaluated according to the metrics expressed in (See Section 5). Also for each model I decided to use *GridSearchCV* for an exhaustive search of the best parameter combinations, integrated with cross-validation.
The core of my approach is the param-grid, a dictionary specifying parameters to test for each classifier. This setup allowed me to systematically explore a wide range of configurations. In the image below you can see all the parameters used and chosen.

Listing 1: Parameter Grid for Model Tuning

```
param_grid = {
    'LogisticRegression': {
        'logisticregression__C': [0.001, 0.01, 0.1, 1, 10]
    },

    'KNeighborsClassifier': {
        'kneighborsclassifier__n_neighbors': [3, 5, 7, 9],
        'kneighborsclassifier__weights': ['uniform', 'distance'],
        'kneighborsclassifier__metric': ['euclidean', 'manhattan']
    },

    'SVC': {
        'svc__C': [0.1, 1, 10],
        'svc__gamma': [0.001, 0.01, 0.1, 1],
        'svc__kernel': ['rbf', 'linear']
    },
```

```
    'DecisionTreeClassifier': {
        'decisiontreeclassifier__max_depth': [10, 20],
        'decisiontreeclassifier__min_samples_split': [2, 5, 10],
        'decisiontreeclassifier__min_samples_leaf': [1, 2, 4],
    },

    'RandomForestClassifier': {
        'randomforestclassifier__n_estimators': [100, 200, 300],
        'randomforestclassifier__max_depth': [10, 20],
        'randomforestclassifier__min_samples_split': [2, 5, 10],
        'randomforestclassifier__min_samples_leaf': [1, 2, 4],
    }
}
```

6.1 Introduction to the graphs

It is right before observing the graphs make a premise, based on all the graphs obtained it is possible to carry out two types of comparative analysis:

- **Comparison of Scaler within the same Dataset**: This comparison was useful to understand which preprocessing method is most effective for a particular machine learning model on one of the two datasets, to understand how much this affects the performance of the model itself.
- **Comparison between Datasets using the same Scaler**: Further useful since it allowed me to evaluate the generalizability of the model and the impact of the different characteristics of the datasets on performance, this ensures that the differences in performance are due to the differences in the data rather than the method used.

To make both analyses possible, the images of each model will be divided into 2x2 matrices.

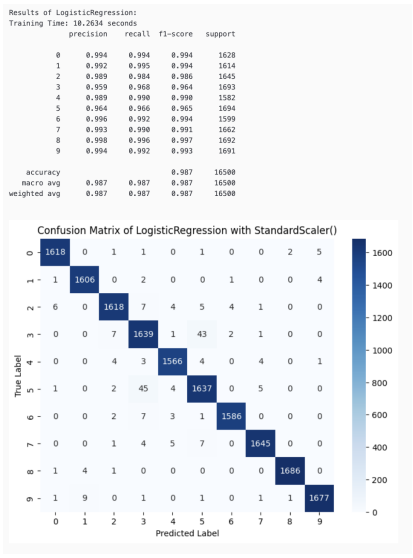
Observation In my analysis of the classification performance metrics, I have noticed a significant trend: the metrics for **classes 3 and 5** are markedly lower than for other classes. This indicates that there might be similarities or overlaps in the features of these classes that are challenging for the models to distinguish. This observation prompted me to perform a more detailed examination of the feature space to identify the exact reasons for this confusion, after several visualizations of the data itself I noticed that many of the samples between two classes are very similar and especially have redundancies.

6.2 Logistic Regression

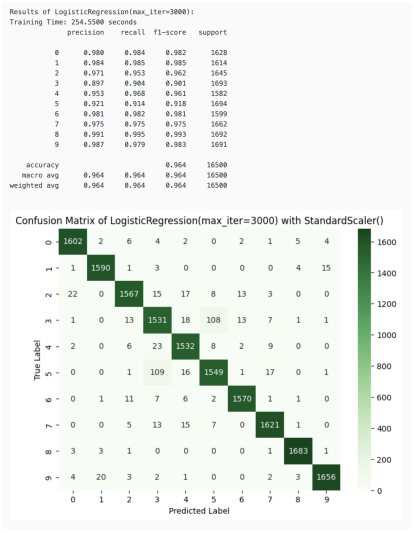
Definition Logistic regression is a statistical model used in machine learning for binary classification tasks. It estimates the probability that a given input belongs to a certain class. The core idea is to model the odds of the probability via a linear combination of the input features, which is then passed through a logistic (sigmoid) function to ensure the output lies between 0 and 1.

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}} \tag{9}$$

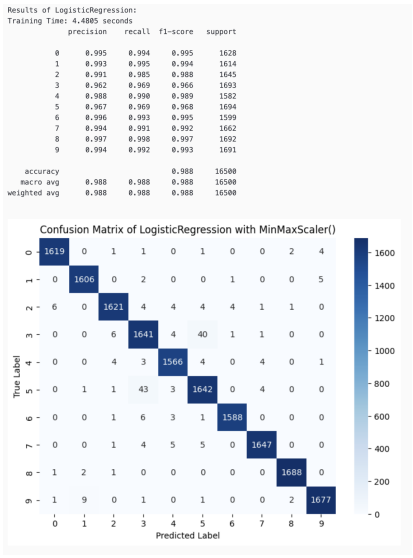
Default settings It can be seen that the only parameter set differently from the default settings is the number of *max-iter = 3000* (default = 100) i.e. the maximum number of iterations taken for the solvers to converge.



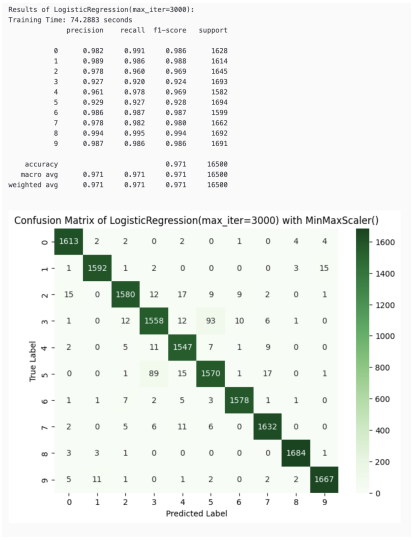
(a) Dataset 1 - Standard Scaler



(b) Dataset 2 - Standard Scaler



(c) Dataset 1 - Min Max Scaler

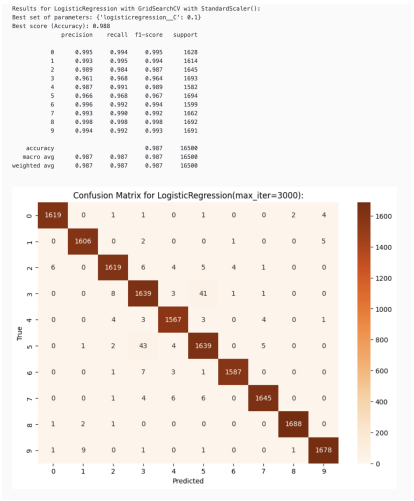


(d) Dataset 2 - Min Max Scaler

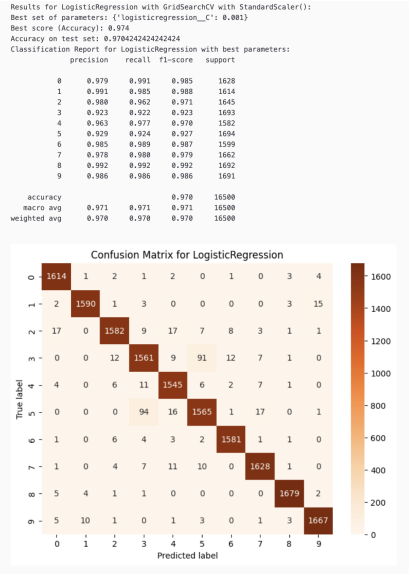
Figure 6: Initial performances of Logistic Regression for each Datasets

Considerations about Default settings The logistic regression models exhibit high and very similar classification performance. The model using MinMaxScaler is marginally more accurate and took less time for training compared to the dataset one with StandardScaler, yet the difference is minimal except for the time, indicating that both scaling methods are effective for this dataset (obviously the difference in time depends a lot on the difference in features of the two datasets). Also in the second dataset it performs much better the MinMax Scaler.

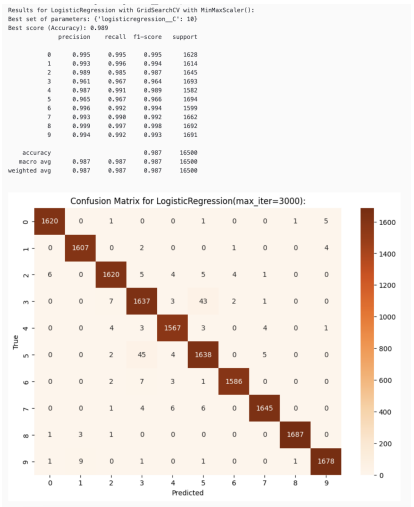
Hypertuned settings I chose to utilize the parameter C, a crucial hyper-parameter that manages the equilibrium between accurately fitting the training data and maintaining the model's simplicity to ensure effective generalization to new, unseen data.
Another adjustment made was to increase the maximum number of iterations, as I observed that during the tuning process, the model tended to converge when the iteration count was set below 4000.



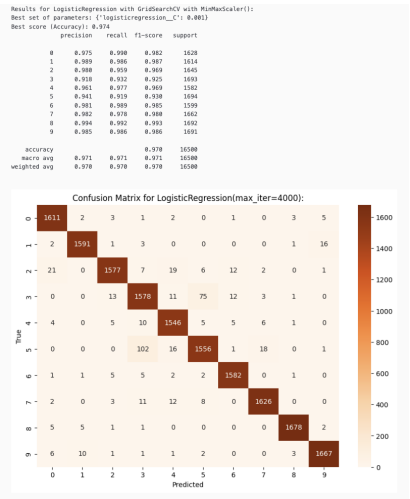
(a) Dataset 1 - Standard Scaler



(b) Dataset 2 - Standard Scaler



(c) Dataset 1 - Min Max Scaler



(d) Dataset 2 - Min Max Scaler

Figure 7: Hypertuned performances of Logistic Regression for each Datasets

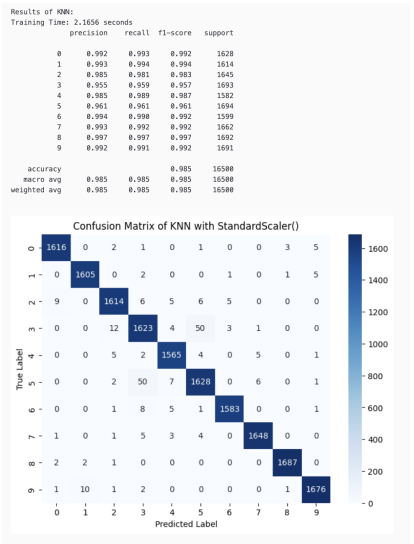
Considerations about Hypertuned settings It can be seen that the only significant improvement over the default settings occurred in the case of Dataset 2 with Standard Scaler, with an improvement from 0.964 to 0.974 in best accuracy, with a parameter set with *C=0.001*.
The classification reports and confusion matrices suggest that the Logistic Regression models perform exceptionally well on both datasets with high precision, recall, and F1-scores, indicating accurate and consistent predictions across all classes. Overall, the model shows high predictive power and stability for the given datasets.

6.3 KNN

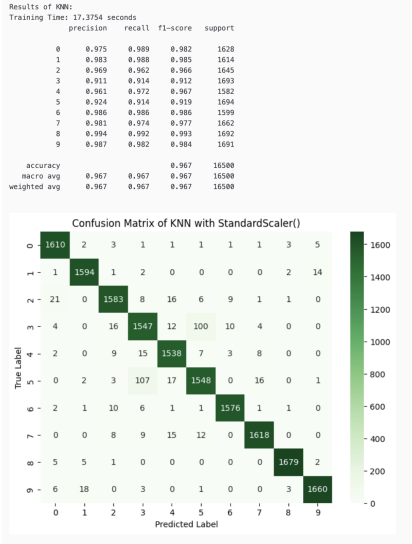
Definition K-Nearest Neighbors (KNN) is a simple, non-parametric algorithm used in machine learning for classification and regression. It operates on the principle that similar things exist in close proximity. In KNN, the output is determined by the majority label among the k-nearest neighbors of a point.

$$y = \frac{1}{k} \sum_{i \in N_k(x)} y_i \tag{10}$$

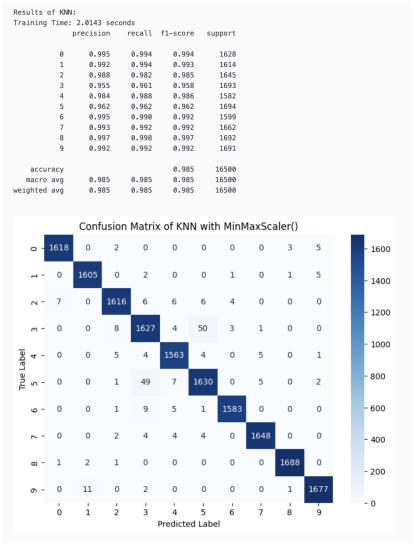
For readability purposes, the default settings and related considerations will be combined into a single paragraph on the following page.



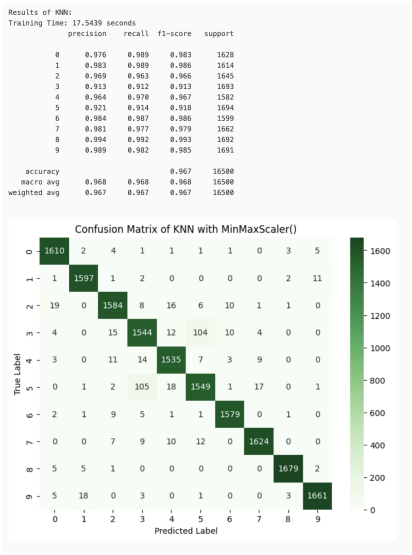
(a) Dataset 1 - Standard Scaler



(b) Dataset 2 - Standard Scaler



(c) Dataset 1 - Min Max Scaler



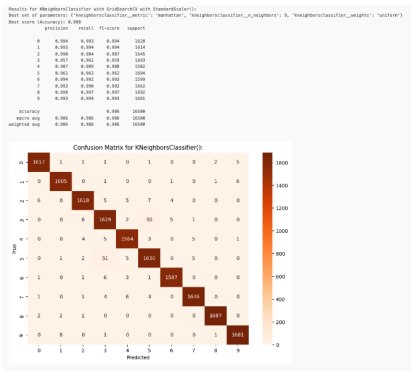
(d) Dataset 2 - Min Max Scaler

Figure 8: Initial performances of KNN for each Datasets

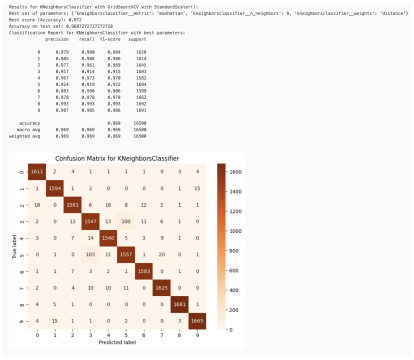
Default settings and Considerations In the default configuration of the k-Nearest Neighbors (kNN) algorithm, the settings are as follows:

- The *n_neighbors* parameter is set to 5, which specifies the number of neighbors to be used in the k-nearest neighbors voting process.
- The *weights* parameter is set to 'Uniform', dictating the weighting function used in making predictions. With the 'uniform' setting, all points within each neighborhood are assigned equal weight.
- The *metric* is configured to 'Minkowski', which defines the distance metric used for the tree. The Minkowski distance with 'p=2' is equivalent to the standard Euclidean distance.

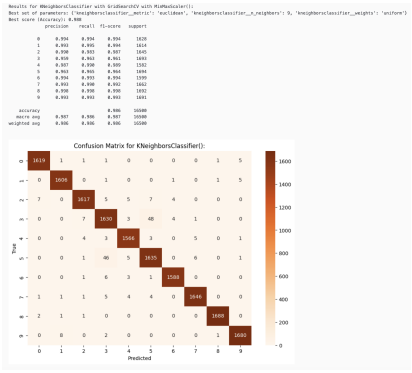
In this case, the performance for both types of scalers for both datasets are about equal, both having very high performance in all metrics. Is possible to see that MinMaxScaler is slightly better for model accuracy, but StandardScaler is much more time efficient.



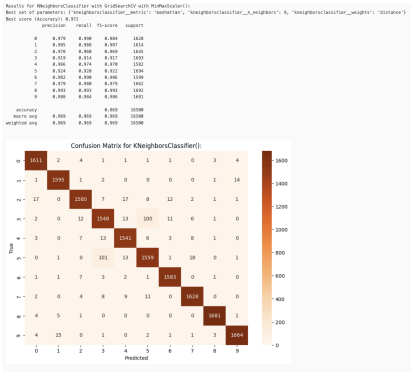
(a) Dataset 1 - Standard Scaler



(b) Dataset 2 - Standard Scaler



(c) Dataset 1 - Min Max Scaler



(d) Dataset 2 - Min Max Scaler

Figure 9: Hypertuned performances of KNN for each Datasets

Hypertuned settings and considerations In the hypertuned configuration, the variations from the default settings, apart from several options for the number of neighbors, also include adjustments to parameters like:

- Setting 'weights' to *distance*, which implies that the influence of each neighbor will be inversely proportional to their distance from the query point. This means neighbors closer to the query point will have a more significant impact on the result compared to those further away.

- Choosing the 'metric' as *manhattan'*, which changes the method of distance calculation to the Manhattan distance. This approach calculates distance based on the sum of the absolute differences in their Cartesian coordinates, differing from the default Minkowski or Euclidean distance.

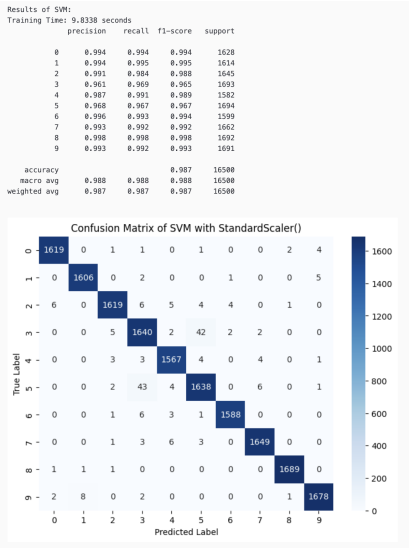
The performance also improves slightly for this model, and it can be seen that for Dataset 1, the optimal number of neighbors is 9 with the 'Euclidean' metric and uniform weights, while for Dataset 2, the 'manhattan' metric with distance-based weights proved more effective. This suggests that the nature of the data in this case has enough influence on the choice of distance metric and type of weighting for neighbors.

6.4 Support Vector Machines

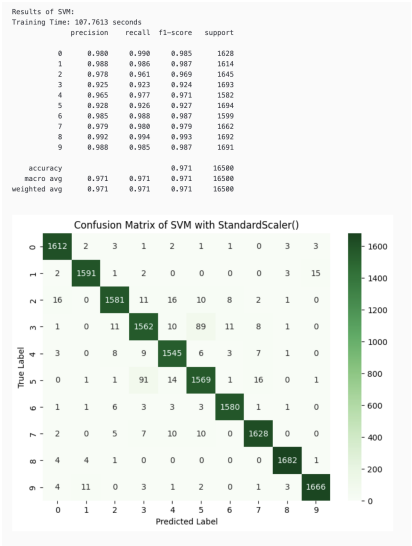
Definition Support Vector Machine (SVM) is a powerful, supervised machine learning algorithm used for classification and regression. It seeks to find the hyperplane that best separates the classes in the feature space. The optimal hyperplane maximizes the margin between the closest points of different classes, known as support vectors.

$$\max_{\mathbf{w},b} \frac{1}{||\mathbf{w}||} \min_{i} y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \tag{11}$$

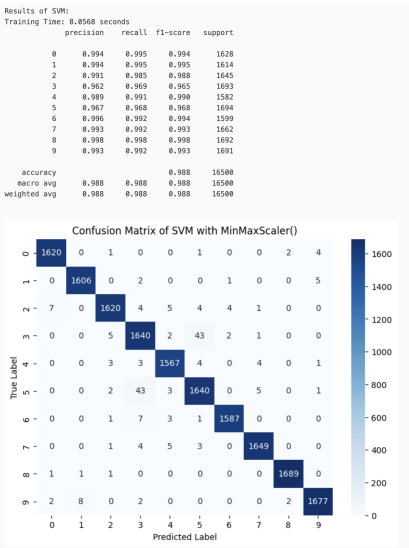
Default settings and Considerations Regarding the default kernel of the SVM used to generate these graphs, the *Radial Basis Function (RBF)* operates by mapping the input data into a higher-dimensional space that facilitates the separation of classes in a non-linear way.
SVM is one of the best performing models, especially in the case of dataset 1 and in general, it seems that MinMaxScaler slightly outperformed StandardScaler in the two datasets.



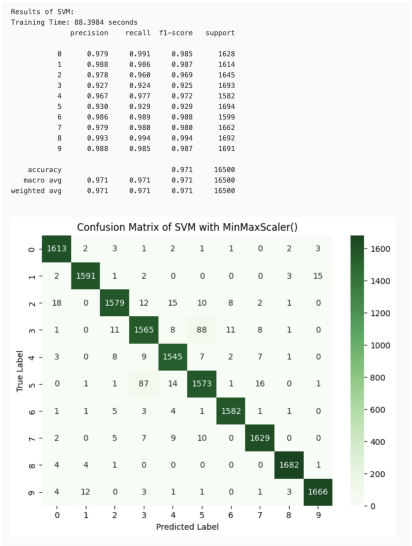
(a) Dataset 1 - Standard Scaler



(b) Dataset 2 - Standard Scaler



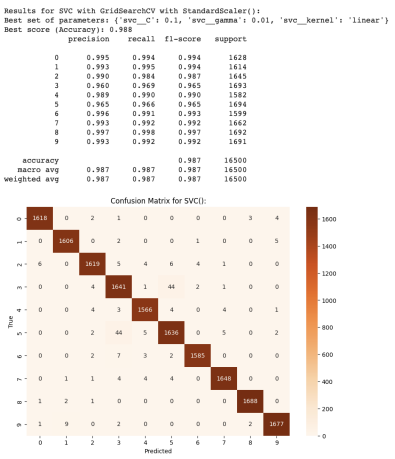
(c) Dataset 1 - Min Max Scaler



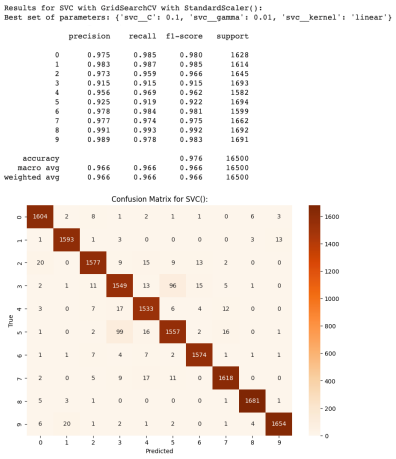
(d) Dataset 2 - Min Max Scaler

Figure 10: Initial performances of SVM for each Datasets

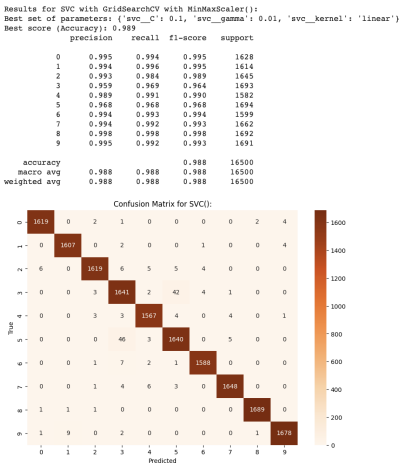
Hypertuned settings For the hypertuning process, a *Linear kernel* was employed. In contrast to the RBF kernel, the linear kernel does not project the data into a higher-dimensional space. Instead, it identifies the optimal linear divider within the existing feature space.
Subsequently, other regularisation parameters such as C, which controls the trade-off between achieving a low error on the training data and minimising the norm of the weights, and another parameter *Gamma*, which influences the decision boundary.



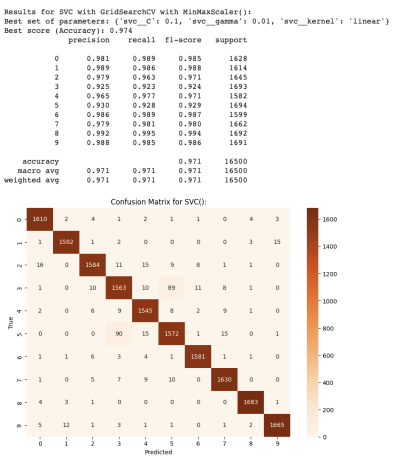
(a) Dataset 1 - Standard Scaler



(b) Dataset 2 - Standard Scaler



(c) Dataset 1 - Min Max Scaler



(d) Dataset 2 - Min Max Scaler

Figure 11: Hypertuned performances of SVM for each Datasets

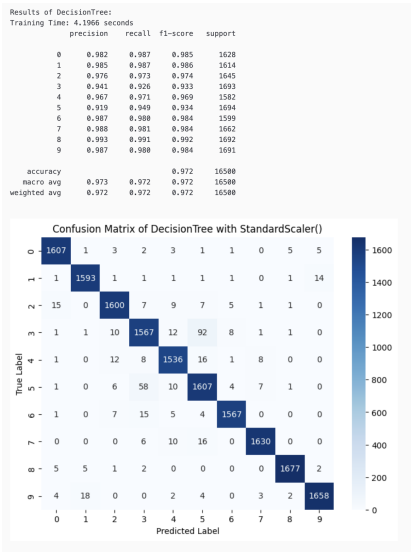
Considerations Here it is possible to observe that for dataset 1, the performance improves slightly even with the same configuration among scalers. As for dataset 2, it is noticeable that the Standard scaler surpasses the Min-Max with a best accuracy of 0.976, which so far is the best compared to the previous Hypertuned models.

6.5 Decision Tree

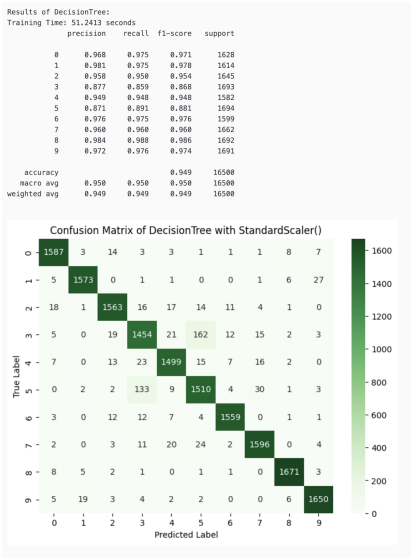
Definition Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features, forming a tree-like structure.

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v) \tag{12}$$

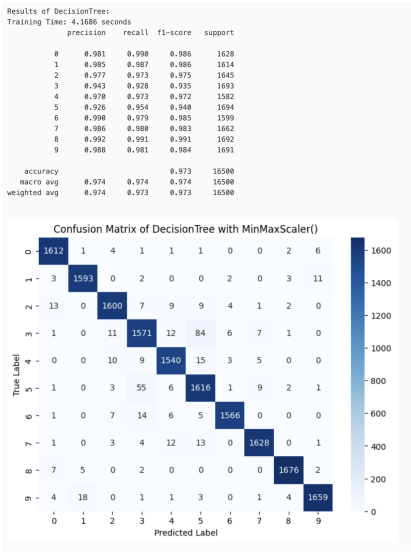
Also here for readability purposes, the default settings and related considerations will be combined into a single paragraph on the following page.



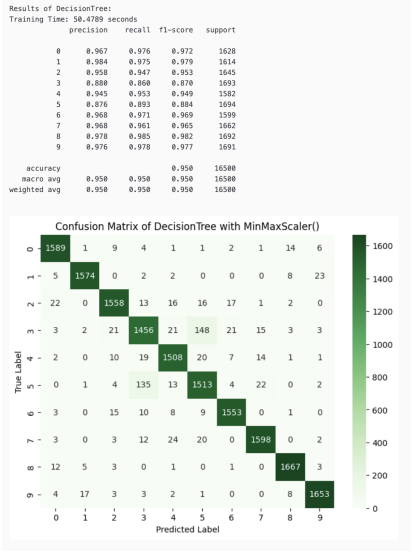
(a) Dataset 1 - Standard Scaler



(b) Dataset 2 - Standard Scaler



(c) Dataset 1 - Min Max Scaler



(d) Dataset 2 - Min Max Scaler

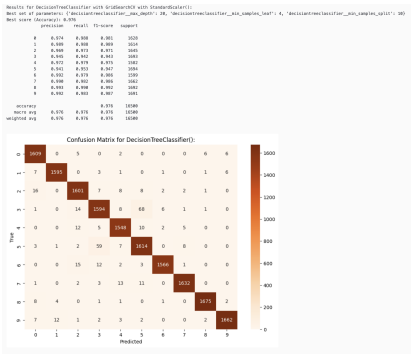
Figure 12: Initial performances of Decision Tree for each Datasets

Default settings and Considerations In this scenario, the default parameters for the decision tree, which will be changed during the hyperparameter setting process, include:

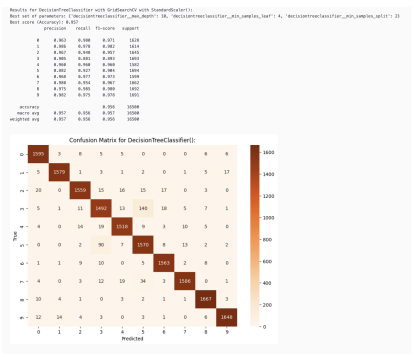
- *max_depth*: set to *None* by default, this parameter allows the tree to expand without limit until all leaves are pure or contain fewer than the number specified by *min_samples_split*.
- *min_samples_split*: with a default value of *2*, it determines the minimum number of samples required to consider a split at an internal node.
- *min_samples_leaf*: set by default to *1*, this specifies the minimum number of samples that must be present in a leaf node.

The Decision Tree's performance metrics indicate minimal differences when using MinMaxScaler versus StandardScaler, with slight variations in accuracy and classes-specific scores. The training times are comparable, suggesting insensitivity to the scaling method.

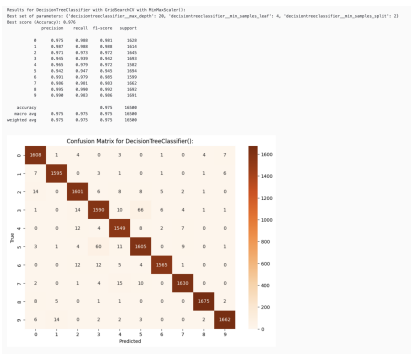
Hypertuned settings The hypertuned Decision Tree models demonstrate enhanced performance, with notable improvements in accuracy reaching up to 0.976. Hyperparameter adjustments, such as increased *min_samples_leaf*, have contributed to a more balanced classification across all classes, as evidenced by the more uniform confusion matrices.



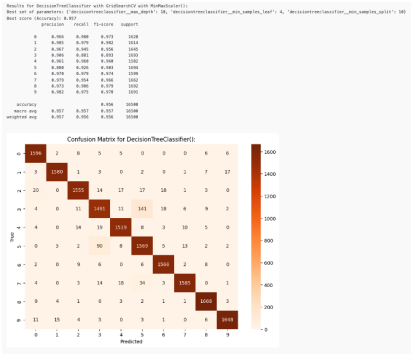
(a) Dataset 1 - Standard Scaler



(b) Dataset 2 - Standard Scaler



(c) Dataset 1 - Min Max Scaler



(d) Dataset 2 - Min Max Scaler

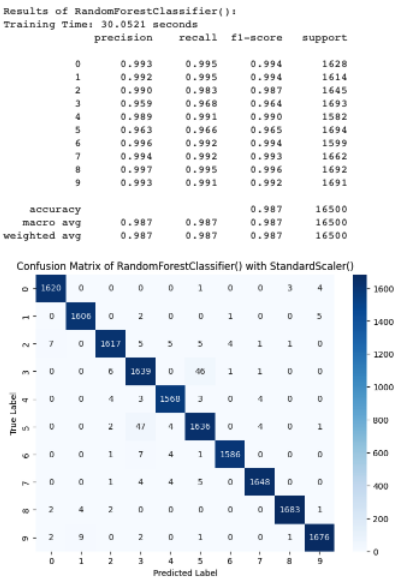
Figure 13: Hypertuned performances of Decision Tree for each Datasets

6.6 Random Forest

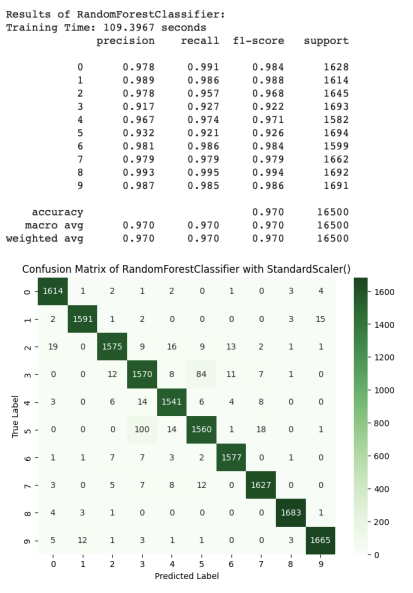
Definition Random Forest is an ensemble learning technique for classification, regression, and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output is the mode of the classes predicted by individual trees.

$$\text{Random Forest Prediction} = \text{Majority Vote}\{\text{Predictions of individual trees}\} \tag{13}$$

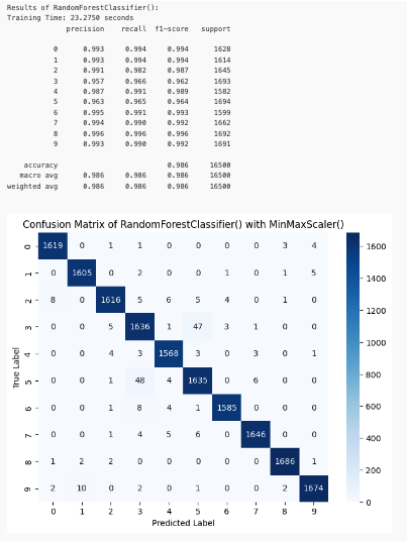
Default settings Regarding the default settings, three out of four of the chosen parameters will be like the previous one, namely: *max_depth*, *min_samples_split*, and *min_samples_leaf*. The 'new' parameter is *n_estimators* set to 100 initially.



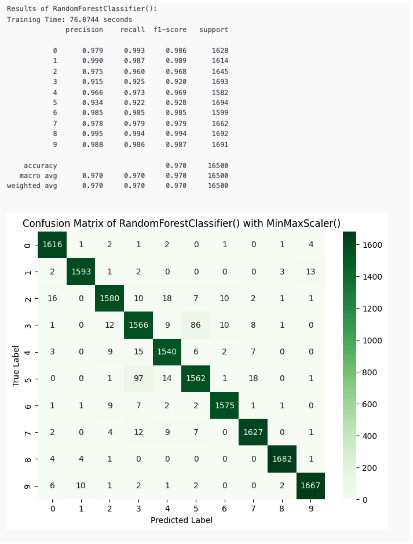
(a) Dataset 1 - Standard Scaler



(b) Dataset 2 - Standard Scaler



(c) Dataset 1 - Min Max Scaler

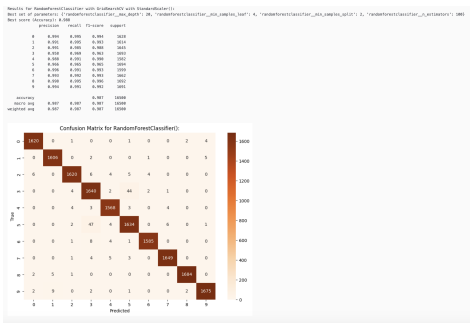


(d) Dataset 2 - Min Max Scaler

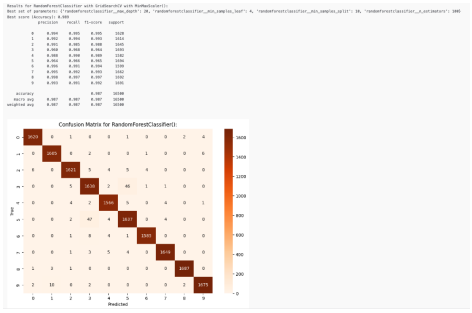
Figure 14: Initial performances of Random Forest for each Datasets

Considerations about default settings The models trained with MinMaxScaler tend to have slightly higher accuracy and slightly more balanced confusion matrices, indicating a better handling of the range of features.
The StandardScaler, while still performing well, shows a slightly higher number of misclassifications, which could be due to the fact that retains outliers that could cause overfitting on some trees in the forest.

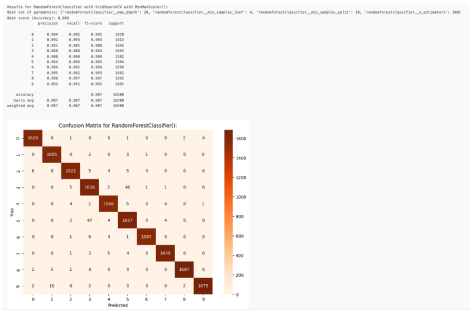
Hypertuned settings I analysed the hyperparameter tuning of Random Forest models and found that both scalers provided very similar results. The confusion matrices confirm the high classification accuracy on all classes. I did not detect a decisive superiority of one scaler over the other, which indicates that both work well. The optimal parameters vary in that the 'n_estimators' parameter selects '200' twice and '100' twice more, but all models demonstrate robustness and reliability, with consistently high accuracy and F1-score.



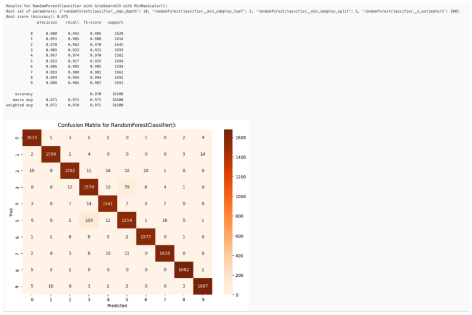
(a) Dataset 1 - Standard Scaler



(b) Dataset 2 - Standard Scaler



(c) Dataset 1 - Min Max Scaler



(d) Dataset 2 - Min Max Scaler

Figure 15: Hypertuned performances of Random Forest for each Datasets

6.7 Evaluation score

This section concerns the Cross Validation Score on the entire dataset for the models analysed in the previous sections. Based on these results, the two models were chosen for the optional blind tests. In general, it can be seen that the performance of the different models is more or less similar and perfectly mirrors the performance trend for these data, except for the decision tree, which proves to be the worst of these results, even if only slightly.

Table 1: Evaluation of the models (on entire dataset)

	Logistic Regression	KNN	SVM	DecisionTree	RandomForest
			Dataset 1		
Std	0.988	0.987	0.988	0.977	0.988
MinMax	0.987	0.987	0.987	0.976	0.988
			Dataset 2		
Std	0.971	0.972	0.968	0.967	0.972
MinMax	0.973	0.971	0.973	0.955	0.973

For blind tests 1 and 2 respectively, I decided to use Random forest and Logistic Regression.

7 Training Time Analysis

Configuration for the tests All tests, and all graphs obtained within this project report were performed on a MacBook Pro M1 machine (so times are rightly affected), with the following configuration:

- RAM: 8 GB unified memory;
- CPU: Apple M1 chip with an 8-core CPU;
- GPU: Integrated 8-core GPU;
- Neural Engine: 16-core NE;

It's also important to note that the machine's performance can be influenced by factors beyond the hardware specifications, such as system load during testing, the specific software versions used for machine learning libraries, and background processes running on the machine.

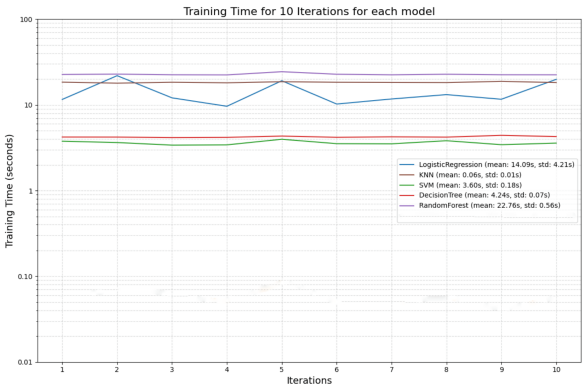


Figure 16: Comparison of Training Times

8 Conclusion and other considerations

In conclusion, this study highlighted the exceptional analytical and processing capabilities of the machine learning models used. Has underlined the critical importance of each phase in data preparation, demonstrating how each step is essential for effective processing.
Thanks to this approach, not only were reliable results obtained, but also areas for further improvement were identified, for example by using other algorithms with completely different approaches.