SAPIENZA
UNIVERSITÀ DI ROMA

Machine Learning - Homework I

Emanuele Frasca 1836098

November 2023

Contents

1 Introduction

In this report, we will analyze and describe the methodology used for classifying two given datasets using Machine Learning algorithms. The code for this report is written in Python using the sklearn module and is available as a Jupyter Notebook.

Our first step is to analyse the datasets, to check their characteristics and see if we have to make a normalization to ensure that all the features follows the same distribution. We also have to check if there are missing values and if we have to adjust the dataset to avoid problems during the training phase.

After having analysed the dataset we have to choose the best algorithms to use for the classification problem. In this report we are going to use three different algorithms: K-Nearest Neighbors, Decision Tree and Random Forest. We are going to use the accuracy on the test dataset as metric to evaluate the performance of each one of them.

Since we want to avoid overfitting we are going to use the k-fold cross validation technique to evaluate the performance of the algorithms. It's important to note that before the training phase we have to split the dataset in two parts: the training set and the test set. The training set is going to be used to train the models and the test set is going to be used to evaluate the performance of the algorithms.

Since we want to find the best hyperparameters for each algorithm we are going to use the grid search technique. This technique is used to find the best hyperparameters for each model. So we are going to use a k-cross validation to evaluate the performance of the algorithms with the hyperparameters found by the grid search.

Finally, we will compare the performance of each algorithm and present our final considerations to determine the most effective approach.

In Figure 1 is possible to see the typical workflow used in Machine Learning problems that we are going to outline and explain in the next chapters.

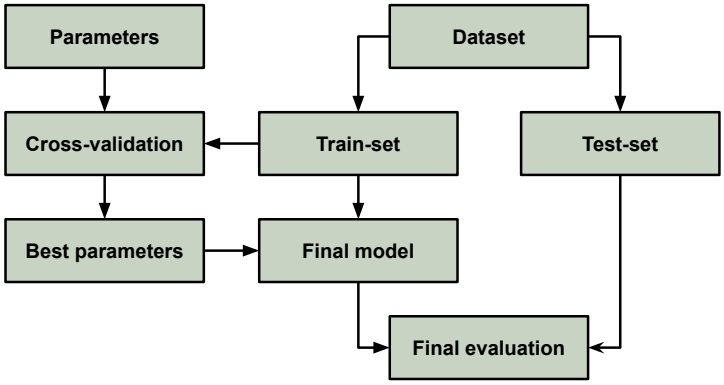

Figure 1: Typical workflow used for Machine Learning problems

2 Dataset Analysis

The dataset analysis is a critical step in any machine learning project. It involves a thorough exploration and understanding of the dataset's characteristics.

Initially, we will check the datasets shapes, including the number of instances and features. This preliminary step helps in understanding the scale of the datasets and guides the choice of the techniques to use.

In this case we will generate a statistical summary of the dataset. This includes calculating mean, median, mode, standard deviation, and range for numerical features. This summary provides insights into the central tendency and dispersion of the data, highlighting any potential anomalies like outliers or skewness in the distribution of the features.

The quality of the datasets will be evaluated by checking for missing values. The presence of missing data can significantly impact the performance of machine learning models, and appropriate strategies for handling such missing data (like imputation, removal, or using algorithms that can handle missing values) will be determined based on this analysis.

Based on the insights gathered from the above analyses, initial data preprocessing steps will be outlined. These may include normalizing or standardizing of numerical data and, potentially, reducing dimensionality (using methods like PCA). This step ensures that the dataset is in an appropriate format for effective model training.

3 Models Used

In a ML problem is important to have a deep understanding of the models that can be used basing on the dataset given. Since we are facing a classification problem we have a large range of models that can be used. In this report we are going to use three different ones: K-Nearest Neighbors, Decision Tree and Random Forest. We choose these models because of their simplicity and their effectiveness so that they don't require a lot of computational power to be trained.

3.1 K-Nearest Neighbors

The K-Nearest Neighbors (KNN) algorithm is a supervised machine learning technique used for classification and regression. It belongs to the family of instance-based, non-parametric learning algorithms. Non-parametric means there is no assumption for the underlying data distribution, which is a useful feature in real-world scenarios where most data does not follow mathematical theoretical assumptions.

KNN works by finding the distances between a query and all the examples in the data, selecting the specified number 'K' of the nearest examples, and then voting for the most frequent label (in the case of classification) or averaging the values (in the case of regression).

Most important parameters of KNN:

- **Number of Neighbors (K):** It represents the number of nearest neighbors to consider for making the prediction.
- **Distance metric:** This describes the metric used to calculate the distance between data points. The most used one are the Euclidean Distance (Standard straight-line distance between two points) and the Manhattan Distance (Sum of the absolute differences of their Cartesian coordinates).
- **Weights:** This parameter is crucial for determining how the nearest neighbors influence the prediction.

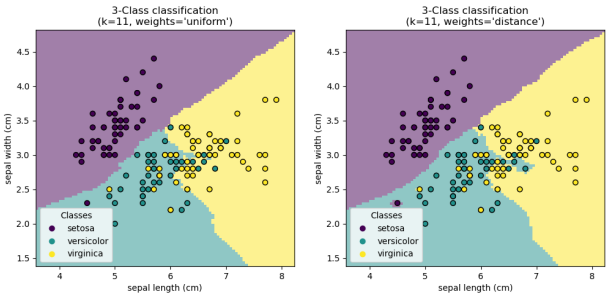

Figure 2: Example of KNN used for classification.

3.2 Support Vector Machines

Support Vector Machines (SVM) are supervised learning models used for classification, regression, and outliers detection, particularly effective in high dimensional spaces. They operate by finding a hyperplane that best divides a dataset into classes, with an emphasis on maximizing the margin between data points of different classes. The core elements of SVM include support vectors, which are the critical data points nearest to the hyperplane, and the margin, defined as the gap between the closest points of different classes.

Most Important Parameters of SVM:

- **Kernel:** The main function of the SVM algorithm, determining the transformation of input data. Common kernels include Linear (for linearly separable data), Polynomial, and Radial Basis Function (RBF), each suitable for different types of data.
- **Regularization Parameter (C):** Controls the trade-off between achieving a low error on the training data and minimizing the norm of the weights. A higher value of C implies less regularization, potentially leading to overfitting, while a lower value increases regularization, possibly leading to underfitting.
- **Gamma (γ) in Kernel Function:** Applicable in non-linear SVMs, particularly with the RBF kernel. It defines the influence of a single training example; a low value indicates 'far' and a high value indicates 'close'.

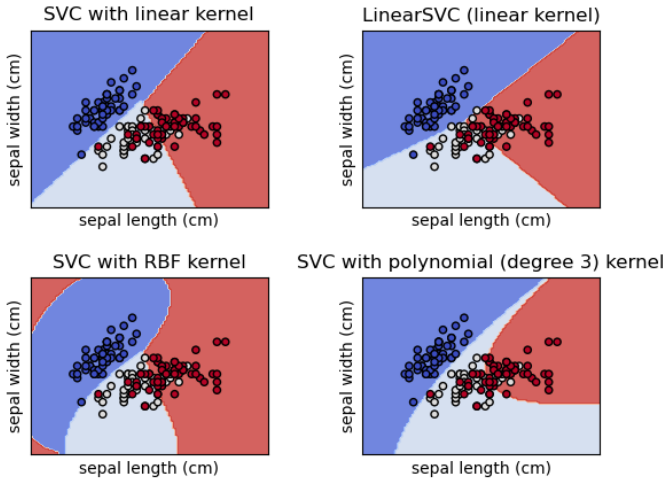

Figure 3: Example of SVC used for classification

3.3 Random Forest

Random Forest is an ensemble learning technique used for both classification and regression tasks, renowned for its accuracy, robustness, and ease of use. It operates by constructing a multitude of decision trees at training time and outputting the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random Forest overcomes the problem of overfitting, common to decision trees, by integrating the predictions from many trees, thereby averaging out biases and providing more reliable results. Each tree in a Random Forest is built on a random subset of the data and considers a random set of features when splitting nodes, introducing diversity among the trees which enhances the model's generalization capabilities.

Most Important Parameters of Random Forest:

- **Number of Trees:** Sets the count of trees in the forest. More trees generally lead to better performance and stability but increase computational load.
- **Criterion:** Determines the function used to measure the quality of a split. 'Gini' implies the use of Gini impurity.
- **Maximum Depth of Trees:** Sets the maximum depth for each tree. 'None' allows unlimited depth, while a set limit like 10 can prevent overfitting.
- **Minimum Samples Split:** The smallest number of samples required to split a node. Set to 2 for high flexibility in splitting.
- **Minimum Samples Leaf:** The minimum number of samples a leaf node must have. Smaller values allow for finer leaf nodes.

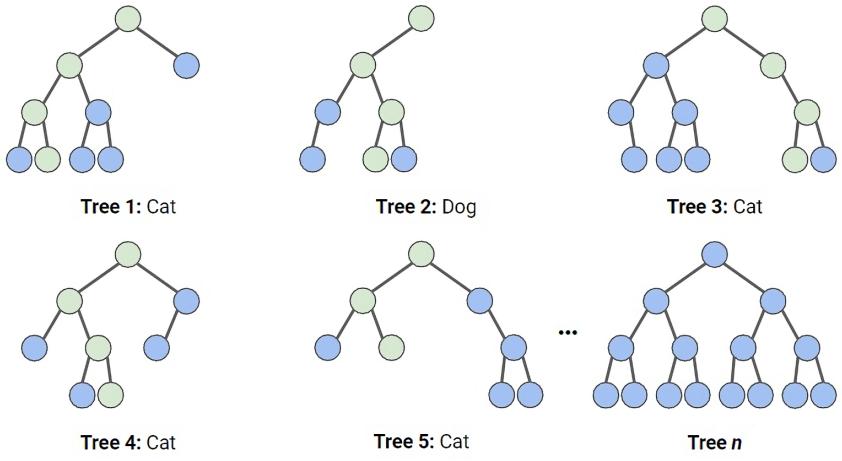

Figure 4: Example of Random Forest used for classification

4 Models Optimization

Since we have outlined our models of interest is now important to make some optimizations to be sure to get the best out of them. In particular we are interested in avoiding the overfitting problem and in the search of the best hyperparameters for our models. To solve both this problems we use a k-fold cross validation and a grid search.

The k-fold cross-validation is a robust method for assessing the performance and generalizability of a machine learning model. In this technique, the dataset is partitioned into 'K' equal-sized subsets or folds. The process involves using one fold as the validation set to test the model and the remaining $K-1$ folds as the training set to train the model. This procedure is repeated 'K' times, with each of the 'K' folds serving as the validation set exactly once. The key advantage of K-fold cross-validation lies in its ability to use every data point for both training and validation, ensuring a comprehensive evaluation of the model.

The results from the 'K' iterations are averaged to produce a single estimation. This method significantly reduces bias as we use all the data for both training and testing, and it also gives a sense of how well the model will generalize to an independent dataset. The choice of 'K' is critical: a lower value of 'K' (such as K=2) can lead to higher variance in the estimation of model performance, while a higher value (like K=10) is often used as a good balance between computational efficiency and reliable estimation. K-fold cross-validation is especially useful when dealing with limited data samples, as it maximizes both the training and testing data available, allowing for a thorough assessment of the model's performance.

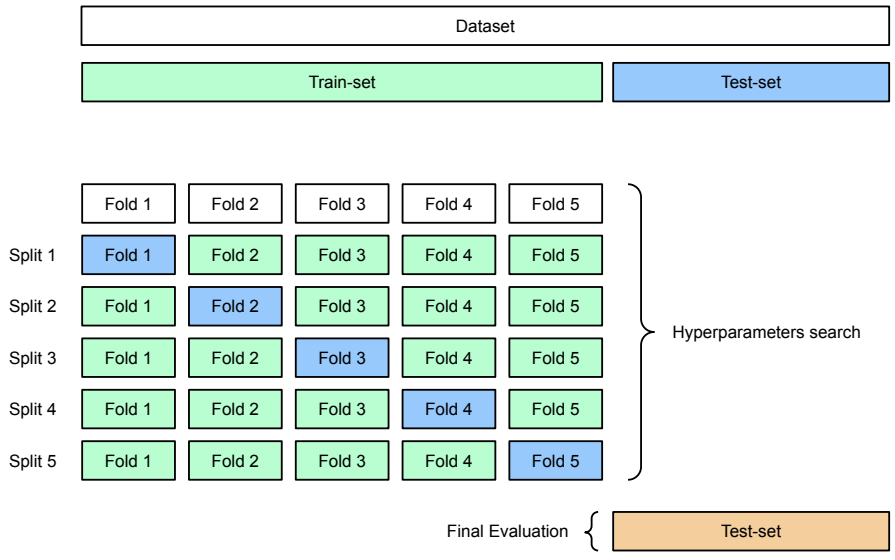

Figure 5: Mechanism of the k-fold Cross Validation

5 Results

5.1 Datasets Analysis and Preprocessing

From the analysis of the data given, analyzing the first dataset, we observe it comprises 50,000 training instances, each characterized by 100 distinct features. The mean across different features varies significantly, suggesting differing scales or units of measurement and the standard deviations varies significantly too meaning that there is a big variability across different features. The min-max range of each feature shows the extent of variation. Some features have a broad range, indicating a wide dispersion of values, while others are more concentrated. For the dataset 2 we can reach the same conclusion but here we have 50,000 training instances, each characterized by 1000 distinct features .

The absence of missing values in both X and y in both datasets is a positive sign, indicating complete datasets. This completeness is essential for ensuring the reliability of the analysis.

This variation in range and standard deviation across features shows the importance of appropriate normalization and scaling during preprocessing to ensure an optimal model training.

Since we want to normalize the dataset we use a StandardScaler. This preprocessing method standardize features by removing the mean and scaling to unit variance.

Given the high dimensionality of the second dataset (1,000 features), we use techniques of dimensionality reduction to reduce the complexity of the model and improve computational efficiency. In particular we are going to perform a Principal Component Analysis to reduce the number of features from 1,000 to 100.

We finally split the datasets in train and test set using as test set the 20% of the dataset. We choose this percentage because we want to have a good amount of data to train and test the algorithms. We also set the random_state to 42 to have the same results every time we run the code so that our result can be reproduced.

5.2 Models Optimization

5.2.1 Support Vector Machine Optimization

For the Support Vector Machine (SVM) optimization, we implemented a k-fold cross-validation with k=3 and conducted a grid search to explore various combinations of parameters. The parameters tested were:

- C: [1, 0.1, 0.01]
- Gamma: [1, 0.1, 0.001]
- Kernel: ['linear', 'poly']

This process resulted in 54 different configurations being evaluated for both datasets.

Optimal Parameters: The optimal parameters identified were 'C': 0.01, 'gamma': 1, 'kernel': 'linear' for both datasets. This choice indicates that a linear kernel with a low regularization parameter (C) and a high gamma value was most effective.

Cross-validation Score: The best cross-validation score achieved for Dataset 1 was 0.9888, indicating a high level of accuracy and suggesting that the model is well-tuned. For Dataset 2, the best score achieved was 0.9725, which is lower, reflecting the increased complexity or variability inherent in this dataset. Additionally, employing Principal Component Analysis (PCA) with higher component values did not result in an increase in the model's score.

Classification Report: The SVM model demonstrated excellent classification performance, with an accuracy of 0.9873 for Dataset 1. The model showed high precision, recall, and f1-scores across all classes, particularly excelling in distinguishing between most classes. A similar trend was observed for Dataset 2, albeit with slightly lower scores of 0.9705 due to the dataset's increased complexity.

Confusion Matrix Analysis: The confusion matrices for both datasets revealed a high degree of accurate classifications with minimal misclassifications. Notably, classes with similar feature spaces showed some degree of confusion, suggesting an area for potential improvement through feature engineering or model adjustment.

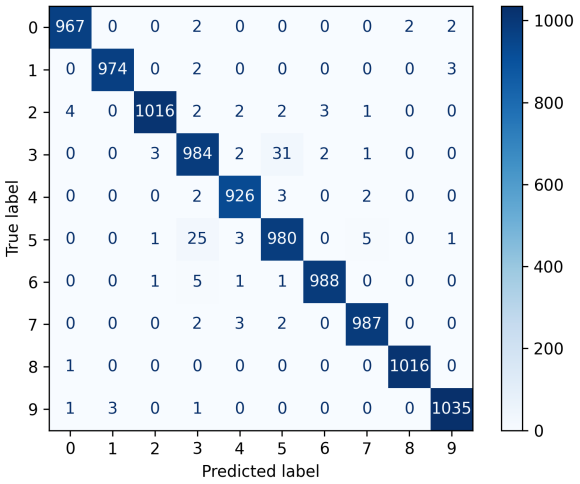

Figure 6: Confusion Matrix for SVM on Dataset 1

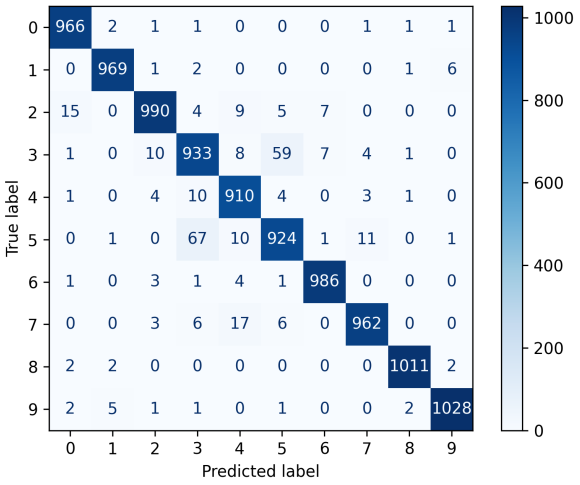

Figure 7: Confusion Matrix for SVM on Dataset 2

5.2.2 K-Nearest Neighbors Optimization

The K-Nearest Neighbors (KNN) model was optimized using a similar approach:

- Neighbors: [6, 7, 8]
- Metric: ['euclidean', 'manhattan']
- Weights: ['uniform', 'distance']

This approach led to the evaluation of 60 different model configurations.

Optimal Parameters: The best parameters were found to be 'metric': 'manhattan', 'n_neighbors': 7, 'weights': 'distance' for Dataset 1, indicating the effectiveness of the Manhattan distance metric and a slightly higher number of neighbors. For Dataset 2, the Euclidean metric with 8 neighbors and uniform weights showed better performance.

Cross-validation Score: The highest cross-validation score for KNN was 0.9873 for Dataset 1 and 0.9710 for Dataset 2, reflecting the increased challenge posed by the higher dimensionality of the second dataset.

Classification Report: The KNN model demonstrated strong classification capabilities with an accuracy of 0.9850 for the first dataset and 0.9679 for the second one.

Confusion Matrix Analysis: The confusion matrices indicated high accuracy in classification with minimal misclassifications. Similar to SVM, the KNN model also showed some confusion among classes with overlapping feature spaces, which could be an area for further model refinement or feature selection.

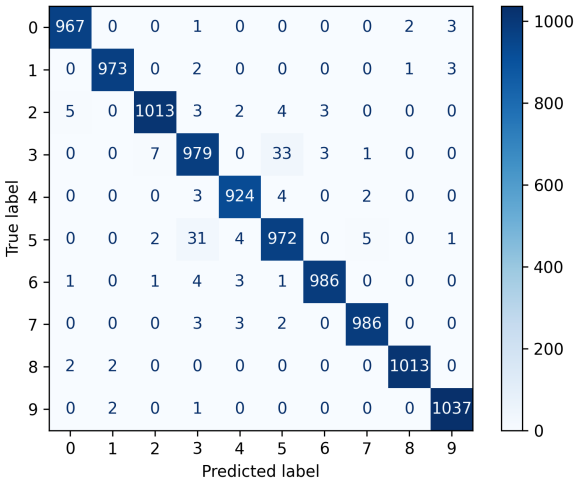

Figure 8: Confusion Matrix for KNN on Dataset 1

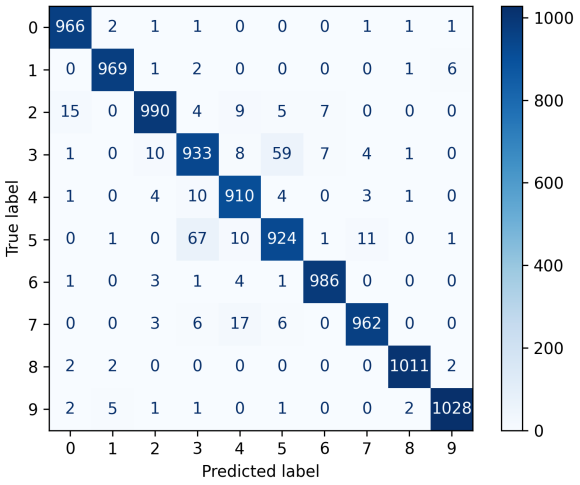

Figure 9: Confusion Matrix for KNN on Dataset 2

5.2.3 Random Forest Optimization

The Random Forest model was optimized through:

- Estimators: [5, 10]
- Max depth: [None, 3, 5]
- Min samples split: [2, 3]
- Min samples leaf: [1, 2]

This approach led to the evaluation of 72 different model configurations.

Optimal Parameters: The grid search for Random Forest resulted in the selection of 'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 10 as the best parameters for both datasets, highlighting the model's preference for more complex tree structures with a moderate number of estimators.

Cross-validation Score: The best score achieved was 0.9859 for Dataset 1, indicating a high level of model accuracy. The score achieved for Dataset 2 was 0.9653, which is consistent with the increased complexity of the dataset.

Classification Report: The Random Forest model showed high accuracy, with notable precision, recall, and f1-scores across different classes. The accuracy on first dataset was 0.9843 and on the second one 0.9679. The model performed consistently well across both datasets, confirming its robustness and adaptability to different data complexities.

Confusion Matrix Analysis: The confusion matrices for both datasets indicated a strong ability of the Random Forest model to accurately classify instances with very few misclassifications. The model demonstrated balanced performance across various classes, with slight confusions in classes with overlapping features.

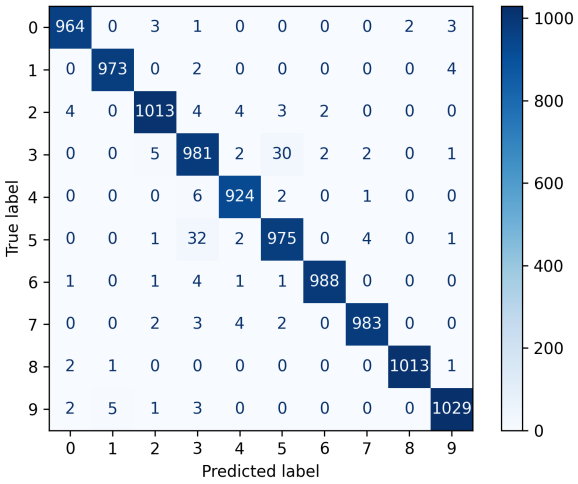

Figure 10: Confusion Matrix for Random Forest on Dataset 1

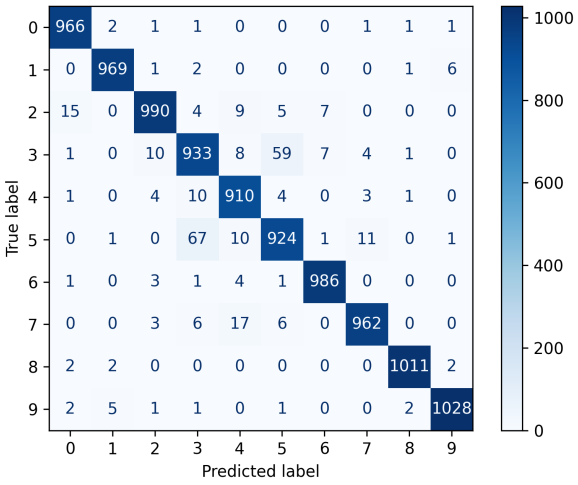

Figure 11: Confusion Matrix for Random Forest on Dataset 2

6 Conclusion

The comprehensive analysis of the first and second datasets using different Machine Learning models demonstrates a significant variance in performance across the models. The application of Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Random Forest algorithms, each optimized using grid search and cross-validation, yielded insightful results.

6.1 Overall Model Performance

Dataset 1:

- The best **Support Vector Machine (SVM)** model emerged as the best model showing an accuracy of 0.9873, demonstrating its effectiveness in handling the dataset with 100 features.
- The **K-Nearest Neighbors (KNN)** algorithm, despite its simplicity, achieved a commendable max accuracy of 0.985, proving its capability in classification tasks.
- The **Random Forest** algorithm showed also an impressive accuracy of 0.9843. Its ensemble approach provided a balanced classification across different classes.

Dataset 2:

- The **SVM** model again showed again the best performance with an accuracy of 0.9705.
- The **KNN** model, with optimized parameters, managed to achieve an accuracy of 0.9679.
- The **Random Forest** model paralleled its Dataset 1 performance with a similar accuracy of 0.9679, indicating its consistent effectiveness across datasets of varying complexity.

6.2 Insights and Recommendations

Model Selection: The SVM model stands out as the most effective across both datasets, particularly for Dataset 1. Its ability to handle high-dimensional data and provide consistent accuracy makes it a favorable choice.

Model Robustness: The consistency in the performance of these models across both datasets, indicates their robustness. However, it's crucial to consider the computational complexity, especially for large datasets. For the first dataset the computational time used for the cross-validations was about 5 minutes, and for dataset 2 about 8 minutes, consistently with the higher features vector dimensions.

6.3 Final Thoughts

In conclusion, the selection of a Machine Learning model greatly depends on the nature of the dataset and the specific requirements of the task. While Random Forest excels in these scenarios, a comprehensive approach that includes a variety of models can provide a more nuanced understanding and effective solutions to classification problems.

A Datasets statistical analysis

A.1 Dataset 1

	0	1	2	3	4
count	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000
mean	0.378181	1.317949	0.009003	1.391512	1.814728
std	0.516047	1.496512	0.050664	1.230377	1.508633
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.248946	0.000000	0.409221	0.463299
50%	0.075048	0.910430	0.000000	1.076016	1.624676
75%	0.608842	1.645869	0.000000	2.084849	2.857240
max	3.206600	8.212224	0.748968	6.523547	6.554948

	...	90	91	92	93
count	...	50000.000000	50000.000000	50000.000000	50000.000000
mean	...	2.272932	0.296990	3.042890	0.339678
std	...	2.104927	0.495357	2.883971	0.446056
min	...	0.000000	0.000000	0.000000	0.000000
25%	...	0.483817	0.000000	0.683260	0.000000
50%	...	1.973584	0.000000	2.138780	0.023752
75%	...	3.381041	0.460844	4.736313	0.722400
max	...	12.077601	2.320764	13.996863	2.380158

	94	95	96	97	98
count	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000
mean	2.625464	0.842655	1.926343	2.506435	0.035964
std	1.673630	1.331757	2.016545	2.694744	0.134504
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.323167	0.000000	0.020918	0.108381	0.000000
50%	2.541312	0.036439	1.263868	1.690138	0.000000
75%	3.915264	0.973730	3.399809	3.824580	0.000000
max	8.275156	6.449932	8.829996	13.680472	1.375294

	99
count	50000.000000
mean	0.009226
std	0.057310
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.057508

A.2 Dataset 2

	0	1	2	3	4
count	5.000000e+04	5.000000e+04	5.000000e+04	5.000000e+04	5.000000e+04
mean	4.556711e-15	-5.460947e-15	1.630696e-16	-5.187815e-15	6.064091e-15
std	1.000010e+00	1.000010e+00	1.000010e+00	1.000010e+00	1.000010e+00
min	-7.328495e-01	-8.806891e-01	-1.777065e-01	-1.130976e+00	-1.202908e+00
25%	-7.328495e-01	-7.143364e-01	-1.777065e-01	-7.983746e-01	-8.958060e-01
50%	-5.874186e-01	-2.723151e-01	-1.777065e-01	-2.564249e-01	-1.259775e-01
75%	4.469816e-01	2.191249e-01	-1.777065e-01	5.635212e-01	6.910379e-01
max	5.480991e+00	4.606942e+00	1.460556e+01	4.171150e+00	3.142095e+00

	...	90	91	92	93
count	...	5.000000e+04	5.000000e+04	5.000000e+04	5.000000e+04
mean	...	-6.558167e-15	-8.637357e-16	3.743708e-15	5.516938e-15
std	...	1.000010e+00	1.000010e+00	1.000010e+00	1.000010e+00
min	...	-1.079826e+00	-5.995520e-01	-1.055115e+00	-7.615222e-01
25%	...	-8.499739e-01	-5.995520e-01	-8.181962e-01	-7.615222e-01
50%	...	-1.422144e-01	-5.995520e-01	-3.134983e-01	-7.082725e-01
75%	...	5.264410e-01	3.307845e-01	5.871901e-01	8.580211e-01
max	...	4.658007e+00	4.085524e+00	3.798264e+00	4.574533e+00

	94	95	96	97	98
count	5.000000e+04	5.000000e+04	5.000000e+04	5.000000e+04	5.000000e+04
mean	-2.760174e-15	-1.207781e-15	7.516121e-15	5.102407e-16	3.888445e-16
std	1.000010e+00	1.000010e+00	1.000010e+00	1.000010e+00	1.000010e+00
min	-1.568740e+00	-6.327457e-01	-9.552784e-01	-9.301292e-01	-2.673805e-01
25%	-7.781348e-01	-6.327457e-01	-9.449052e-01	-8.899092e-01	-2.673805e-01
50%	-5.028150e-02	-6.053840e-01	-3.285232e-01	-3.029249e-01	-2.673805e-01
75%	7.706685e-01	9.842372e-02	7.306956e-01	4.891590e-01	-2.673805e-01
max	3.375746e+00	4.210477e+00	3.423539e+00	4.146646e+00	9.957623e+00

	99
count	5.000000e+04
mean	-2.941647e-17
std	1.000010e+00
min	-1.609823e-01
25%	-1.609823e-01
50%	-1.609823e-01
75%	-1.609823e-01
max	1.829157e+01