DRY-BEANS CLASSIFICATION

Sasu Alexandru-Cristian (407), Barbu Sebastian-Marian (411), Plăian Georgiana (411)

Grigore Dragos-Gabriel (412), Zibileanu Sabin (412)

I. INTRODUCTION

For this project, we have chosen a Data Science one, where the task we tackled was: **classification**. To successfully fulfil the needs of the project, we made multiple experiments using different Machine Learning models and dimensionality reduction techniques.

Our main motivation regarding this project was to develop some Machine Learning models that can help in the agricultural field by classifying types of beans, which also according to the authors are "*the most produced ones among the edible legume crops in the world*".

II. DATASET

The dataset we chose was found on "*UCI Machine Learning*" and it is about the classification of dry beans. It was first introduced in the paper "*Multiclass classification of dry beans using computer vision and machine learning techniques*", by M. Koklu and I.A. Ozcan (2020).

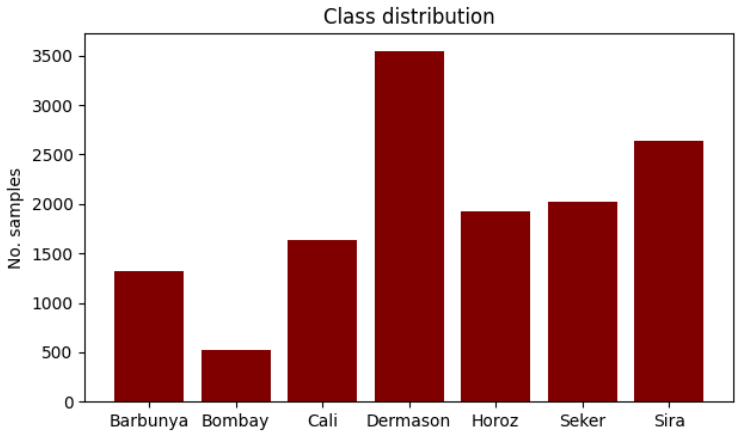
The size of the dataset was 3.452 KB and it contains images of 13.611 grains of 7 different registered dry beans that were taken with a high-resolution camera. The grains were distributed along 7 categories:

- BARBUNYA
- BOMBAY
- CALI
- DERMASON
- HOROZ
- SEKER
- SIRA

The dataset also contains 16 features, out of which 12 describe the dimensions of the bean and 4 describe their shape. Some of them are:

- Area – the area of a bean zone and the number of pixels within its boundaries;
- Perimeter – bean circumference is defined as the length of its border;
- Major axis length – the distance between the ends of the longest line that ca be drawn from a bean;
- Minor axis length – the longest line that ca be drawn from the bean while standing perpendicular to the main axis;
- Convex area – number of pixels in the smallest convex polygon that can contain the area of the bean;
- Roundness ;
- Compactness;
- Class;

  etc.



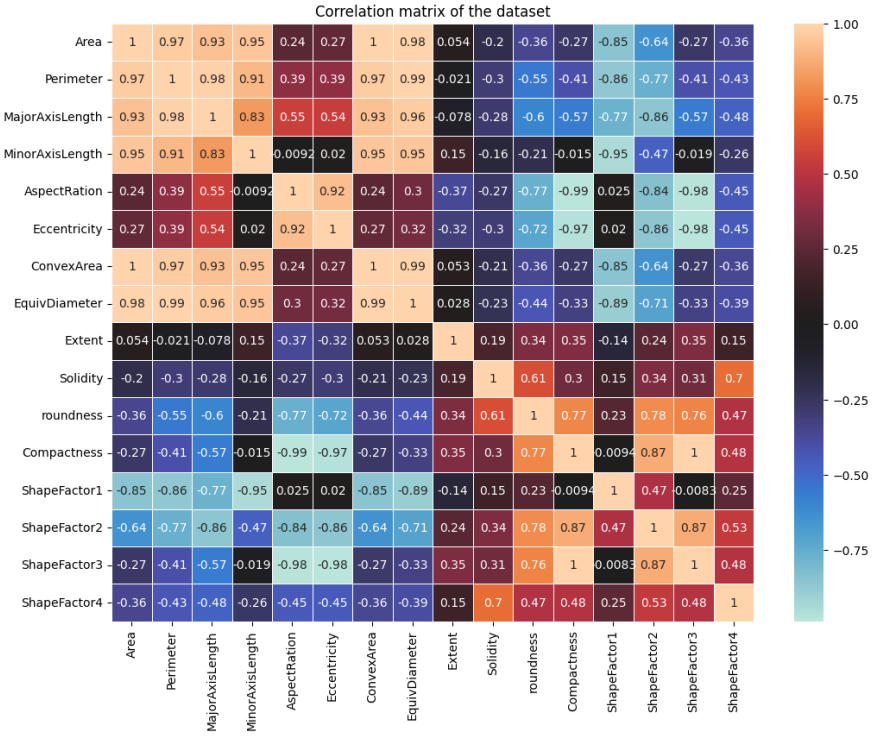
As shown above, our dataset was imbalanced. To address potential issues, we employed measures at the level of training machine learning algorithms. For example, we used balanced weights for Random Forest, instead of a weight of 1 for each class.

Throughout the project, we aimed to observe the features that are the most important for our classification, using various models such as:

- Logistic Regression
- K-Nearest Neighbor (KNN)
- Random Forest Classifier
- Support Vector Machines (SVM)
- Naïve Bayes

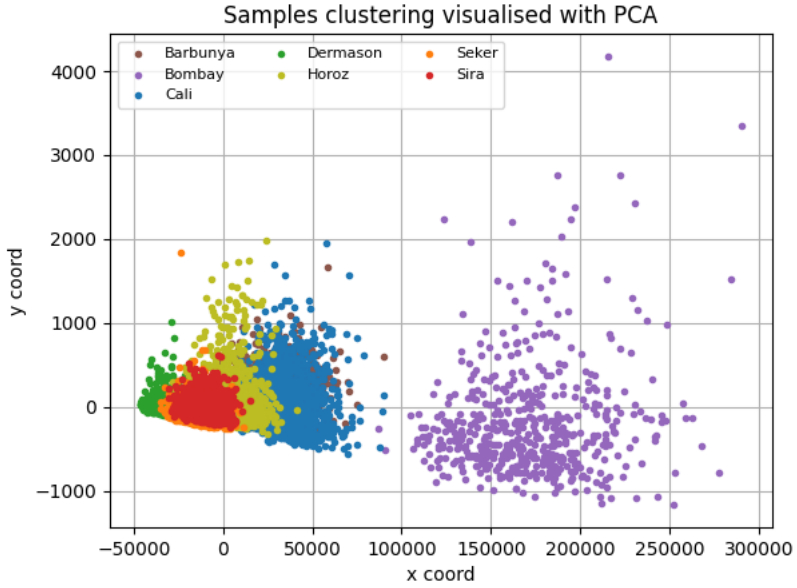
III. EXPLORATORY DATA ANALYSIS

Firstly, we plotted the correlation matrix where we can observe the relationships between our features.



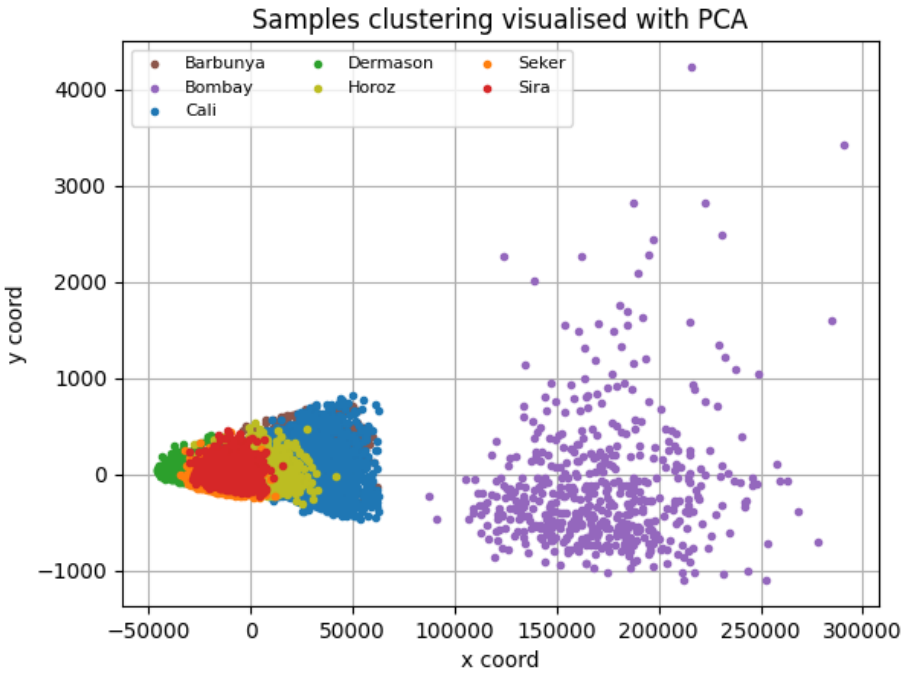
Now, it is clear that the dimension features are in a strong inverse relationship with the shape features. This opens possibilities for removing multiple features that are similar to others, though this will be realized in our project in an automatic fashion, by our dimensionality reduction algorithms.

To observe our initial dataset in 2D, we can use the PCA algorithm and plot the resulting two coordinates.



From the previous plot, we can observe how the samples from the “Bombay” class are very different from the rest of the samples. This may prove beneficial for training our models, as they could differentiate more easily between “Bombay” samples and samples from other classes.

In addition, we performed an automatic removal of outliers from our data, by making use of the z-score method. Thus, we removed only samples that were more than three standard deviations away from the mean, having eliminated in total 669 samples. During this process, we did not eliminate any samples from the “Bombay” class, as it already had the lowest number of samples from all classes, by a substantial amount, and we feared that it would impact the training of our models. The data could then be represented like this:



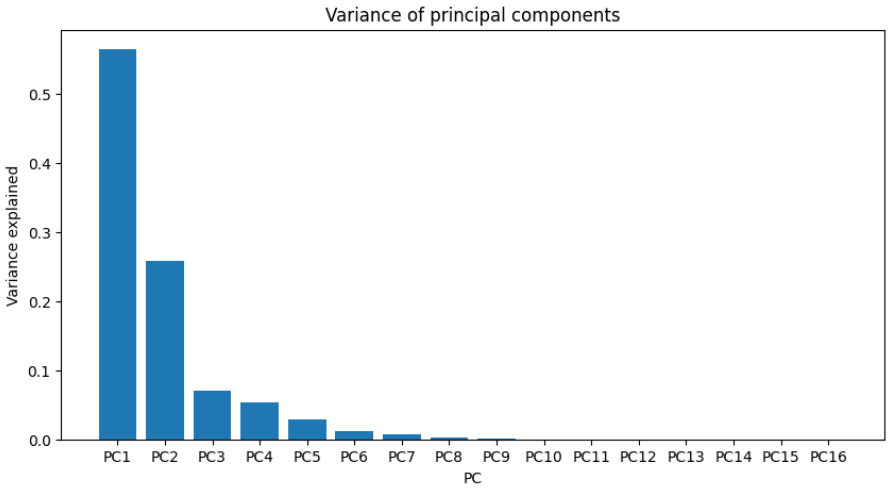
IV. DIMENSIONALITY REDUCTION METHODS

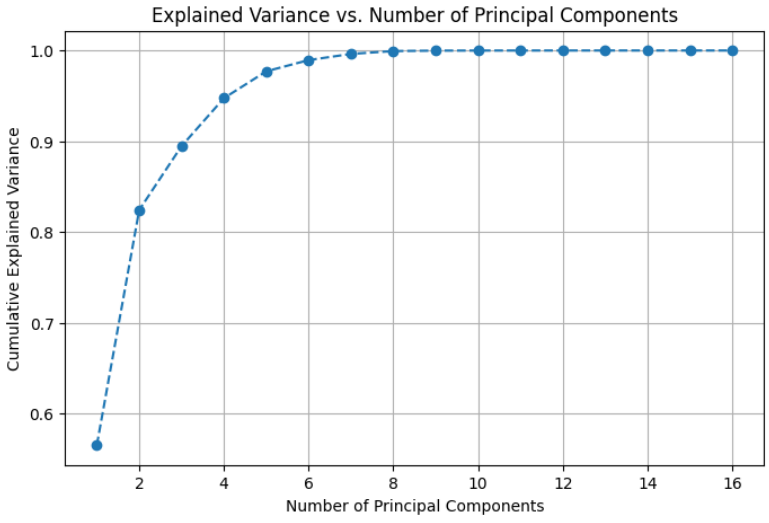
As a means of both normalizing our data and surpassing the sensitivity to mean and variance of the PCA, we used *StandardScaler* to preprocess our dataset so that the features would have a mean of 0 and variance of 1.

1. Principal Component Analysis (PCA)

The first dimensionality reduction technique that we used was Principal Component Analysis, also known as PCA. Having to work around with 16 features, a dimensionality reduction technique such as PCA played a very important role in our project, reducing the data to its essential features.

To find the best number of components for PCA, we decided to look at the variance of the PCA model and plot the graph for both variance and cumulative explained variance to have a better look at what number of components needs to be chosen. We decided to keep the first ***'n'*** principal components that explained at least 95% of the variance, so the number of components we chose was 5, as it can be deducted from the next 2 figures:

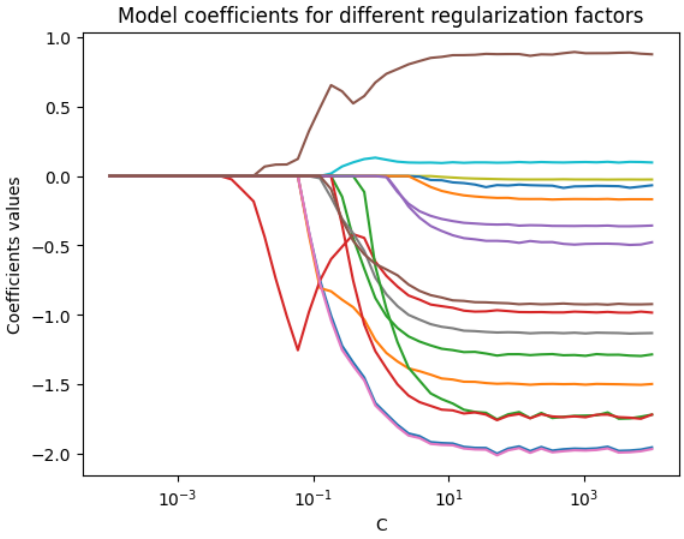




2. Lasso variable selection property

For this method, we employed a feature selection method, namely the variable selection property of the L1 (Lasso) regularization factor. The Lasso variable selection works by imposing an L1 penalty on the coefficients, which shrinks some coefficients to zero and selects a simpler model with fewer predictors. This helps in improving model interpretability and prediction accuracy by reducing overfitting.

In order to make use of this method, we used a Logistic Regression model. First we wanted to look at how the model shrinks the coefficients based on a wide range of values for lambda (the C parameter of the model in Python), and since our task is a multinomial one, we plotted the values of the coefficients only for the model that predicted the largest amount of correct labels for the most difficult class, in this case the class “Sira”, or the seventh class.



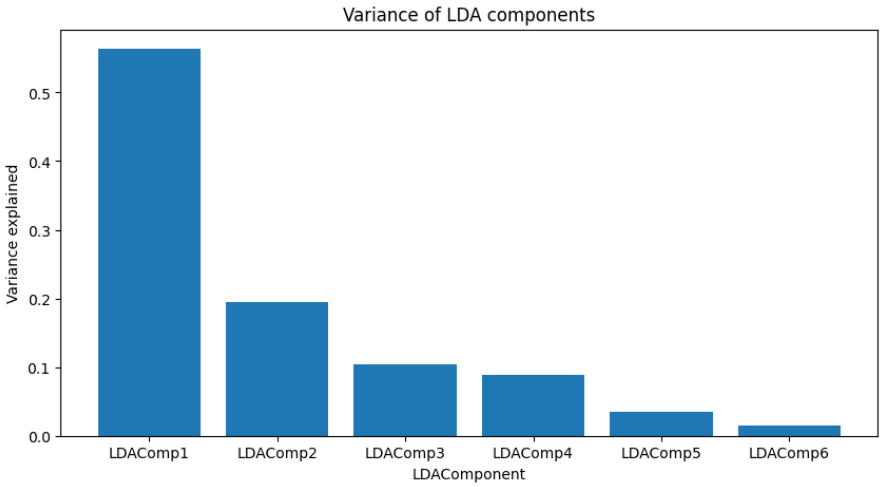
Upon observing the resulting plot, we decided that the range of lambda values $10^{-1.5}$ – 10^{1} holds the optimal value for C for variable selection. Then, we proceeded to use the Grid Search algorithm with multiple lambda values from this optimal range, after which we selected the resulting value of the C parameter of the model that had an acceptable macro F1 score, namely over 0.933, which selected the lowest number of features. This way, we came up with a satisfactory C of 0.188 and the following 8 selected features: Area, Perimeter, MinorAxisLength, Eccentricity, ConvexArea, EquivDiameter, Solidity, ShapeFactor4.

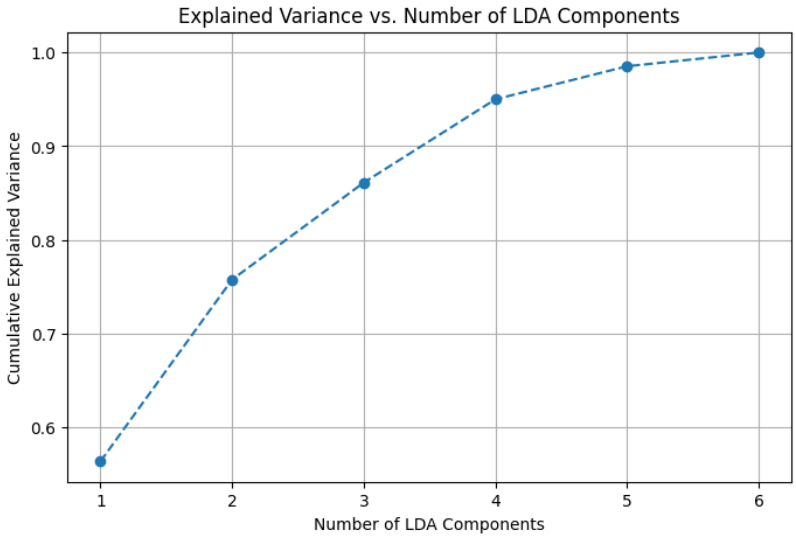
3. Linear Discriminant Analysis (LDA)

Just like PCA, LDA reduces dimensionality, but the main difference between the two is that LDA focuses solely on maximizing separability among the known categories, which, in our case, are the seven different categories of beans.

For a binary classification task, LDA reduces dimensionality by minimizing the distance between the means of the data and also minimizing the variation of the data. When it comes to multiple categories, LDA finds the central point of the whole data and then finds the central points for each category and measures the distance between them. The purpose is maximizing the distance between each category and minimizing the variation for each category.

In order to find the optimum number of components, we used the same steps as we did before with PCA, and the result was 4, as it can be seen from the figures below:





V. THE MODELS

1. Logistic Regression

This model can be used to determine the probability of an event, given some input data, being different than linear regression where the predicted output is a continuous one. Logistic regression can be binary or multinomial (here it is multinomial).

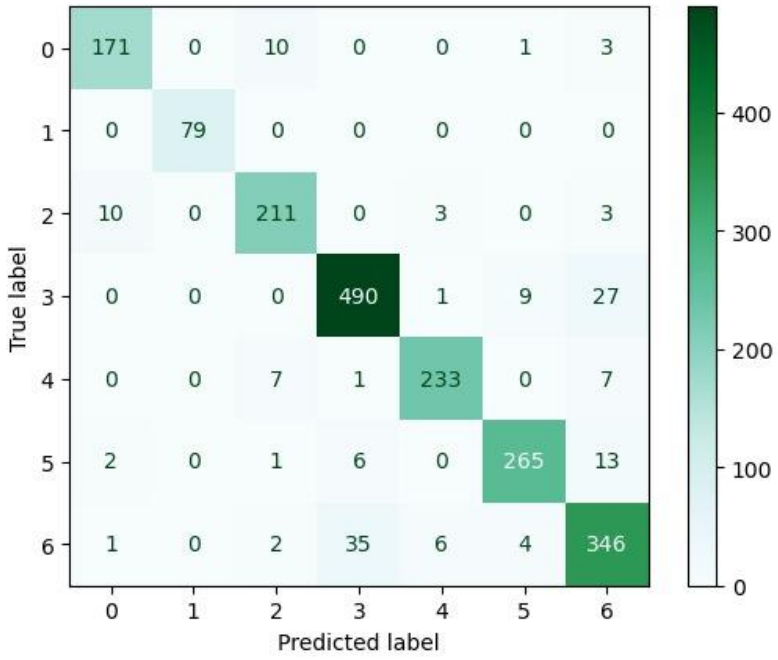
In terms of preprocessing steps, the outliers were removed from the data frame and then scaled using S*tandard Scaler*. We have made 4 different experiments when using the logistic regression model, precisely, we have tested the model's behavior when different dimensionality reduction techniques were applied.

1.1 Logistic Regression without any dimensionality reduction technique

The first experiment involved applying the logistic regression model to the scaled data without any dimensionality reduction applied to it. The best parameters for the model were quickly found using grid search:

- C: 10
- Class weight: none
- Max iter: 1000
- Solver: lbfgs
- Time elapsed: 36.68 seconds
- Accuracy: 0.92

Label	Precision	Recall	F1-score
0	0.93	0.92	0.93
1	1	1	1
2	0.91	0.93	0.92
3	0.92	0.93	0.93
4	0.96	0.94	0.95
5	0.95	0.92	0.94
6	0.87	0.88	0.87

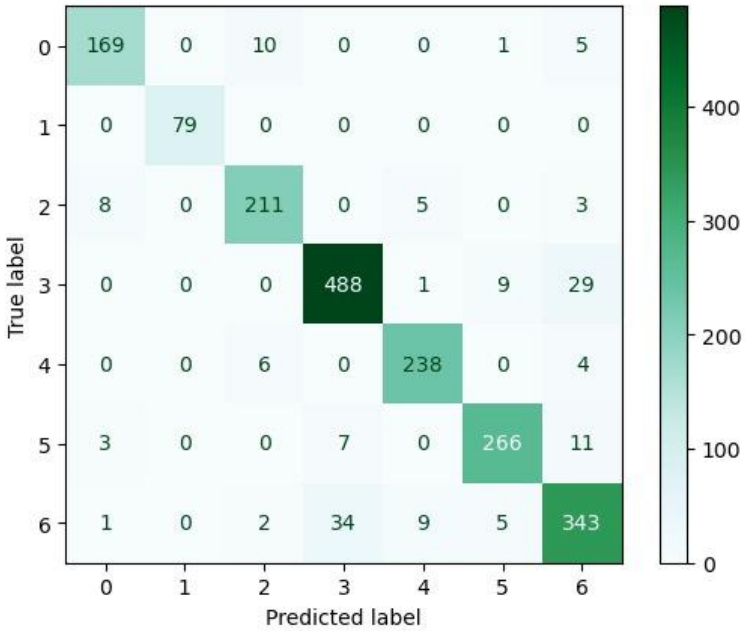


1.2 Logistic Regression + LVS

The best parameters obtained from the grid search were the same as the ones obtained where no dimensionality reduction techniques were applied:

- C: 10
- Class weight: none
- Max iter: 1000
- Solver: lbfgs
- Time elapsed: 19.74 seconds
- Accuracy: 0.92

Label	Precision	Recall	F1-score
0	0.93	0.91	0.92
1	1	1	1
2	0.92	0.93	0.93
3	0.92	0.93	0.92
4	0.94	0.96	0.95
5	0.95	0.93	0.94
6	0.87	0.87	0.87



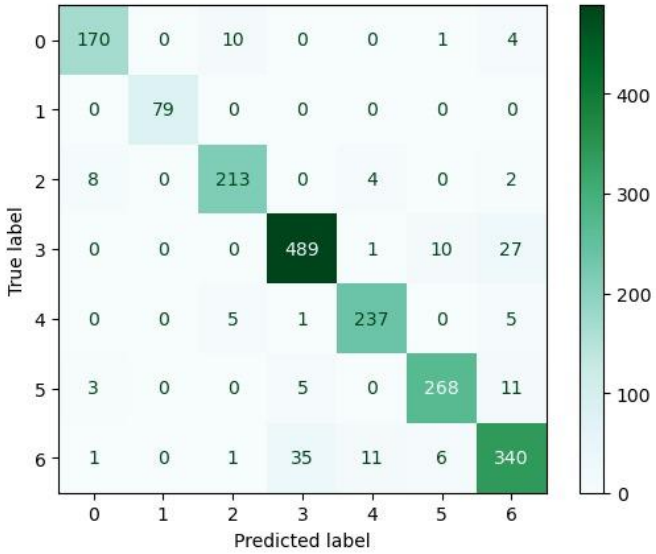
1.3 Logistic Regression + PCA

The first dimensionality reduction technique applied to our data was Principal Component Analysis. After finding the optimum number of components and that was 4 initially and then 5 after the outliers were removed. The best parameters obtained from grid search are:

- C: 1.0
- Class weight: none
- Max iter: 1000
- Solver: lbfgs
- Time elapsed: 7.76 seconds
- Accuracy: 0.92

We fit the logistic regression model on our modified data and the results are:

Label	Precision	Recall	F1-score
0	0.93	0.92	0.93
1	1	1	1
2	0.93	0.94	0.93
3	0.92	0.93	0.93
4	0.94	0.96	0.95
5	0.94	0.93	0.94
6	0.87	0.86	0.87



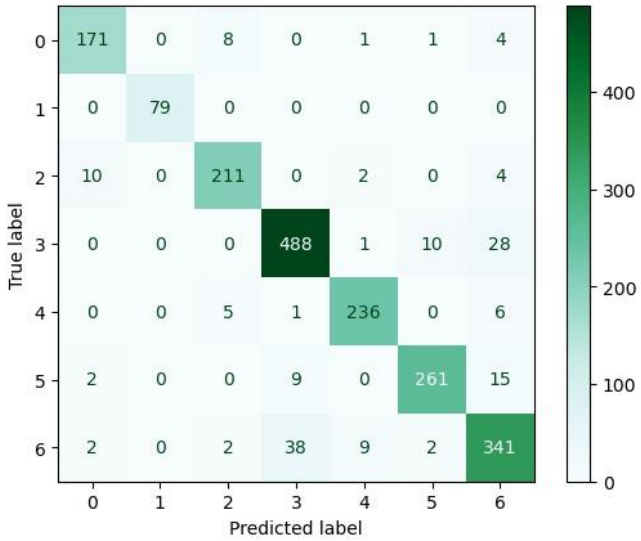
1.4 Logistic Regression + LDA

The final experiment that we did was applying Linear Discriminant Analysis (LDA), because we wanted to see how the model behaves when using a dimensionality reduction technique which focuses solely on maximizing the separability of the classes. The best parameters obtained from the grid search are the following:

- C: 10
- Class weight: none
- Max iter: 1000
- Solver: lbfgs
- Time elapsed: 12.35 seconds
- Accuracy: 0.92

We fit the logistic regression model on our modified data and the results are:

Label	Precision	Recall	F1-score
0	0.92	0.92	0.92
1	1	1	1
2	0.93	0.93	0.93
3	0.91	0.93	0.92
4	0.95	0.95	0.95
5	0.95	0.91	0.93
6	0.86	0.87	0.86



2. K-Nearest Neighbor

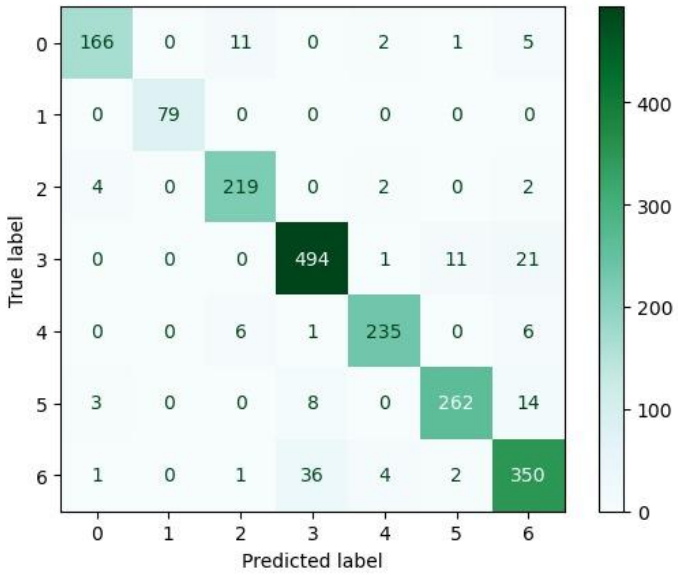
K-Nearest Neighbor is a simple and widely used machine learning algorithm, particularly in the context of classification and regression tasks. This model is used for supervised learning tasks and thus it is a 'lazy learning algorithm', the predictions are made based on the proximity of the new data point.

2.1 K-Nearest Neighbor without any dimensionality reduction technique

The first experiment involved applying the KNN model to the scaled data without any dimensionality reduction applied to it. The best parameters for the model were quickly found using grid search:

- N neighbors: 15
- Metric: euclidean
- Time elapsed: 5.27 seconds
- Accuracy: 0.93

Label	Precision	Recall	F1-score
0	0.95	0.90	0.92
1	1	1	1
2	0.92	0.96	0.94
3	0.92	0.94	0.93
4	0.96	0.95	0.96
5	0.95	0.91	0.93
6	0.88	0.89	0.88

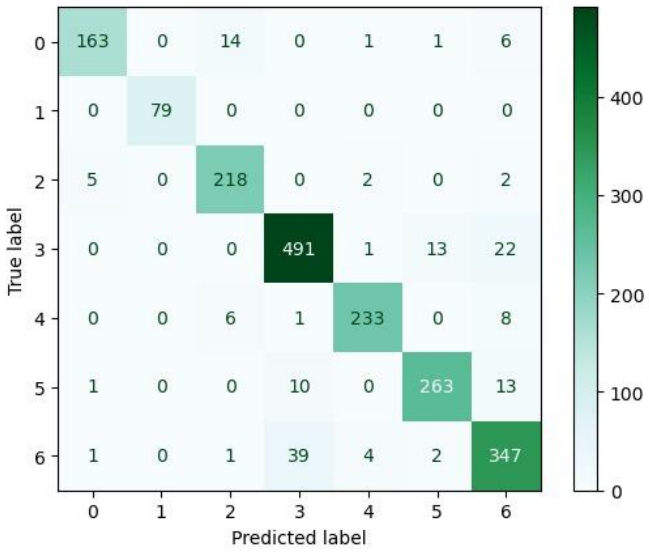


2.2 K-Nearest Neighbor + PCA

The first dimensionality reduction technique applied to our data was Principal Component Analysis. The best parameters for the model were quickly found using grid search:

- N neighbors: 21
- Metric: manhattan
- Time elapsed: 1.91 seconds
- Accuracy: 0.92

Label	Precision	Recall	F1-score
0	0.96	0.88	0.92
1	1	1	1
2	0.91	0.96	0.94
3	0.91	0.93	0.92
4	0.97	0.94	0.95
5	0.94	0.92	0.93
6	0.87	0.88	0.88

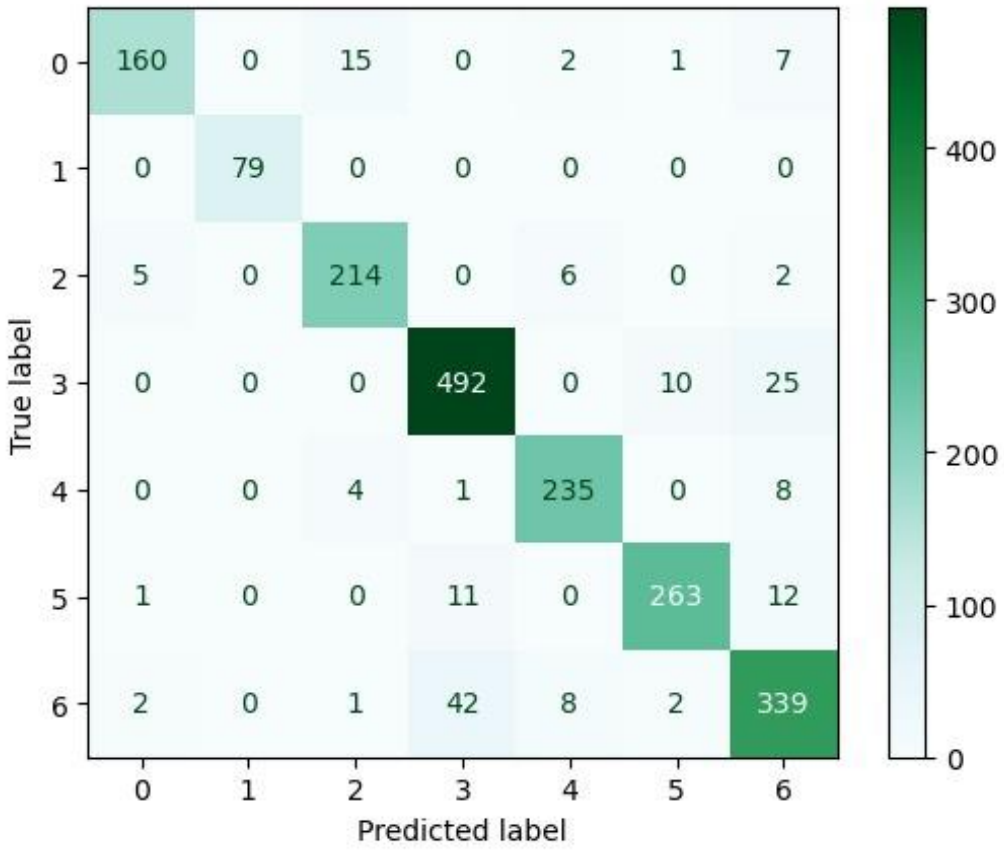


2.3 K-Nearest Neighbor + LVS

The best parameters for the model were quickly found using grid search:

- N neighbors: 15
- Metric: euclidean
- Time elapsed: 1.80 seconds
- Accuracy: 0.92

Label	Precision	Recall	F1-score
0	0.95	0.86	0.91
1	1	1	1
2	0.91	0.94	0.93
3	0.90	0.93	0.92
4	0.94	0.95	0.94
5	0.95	0.92	0.93
6	0.86	0.86	0.86

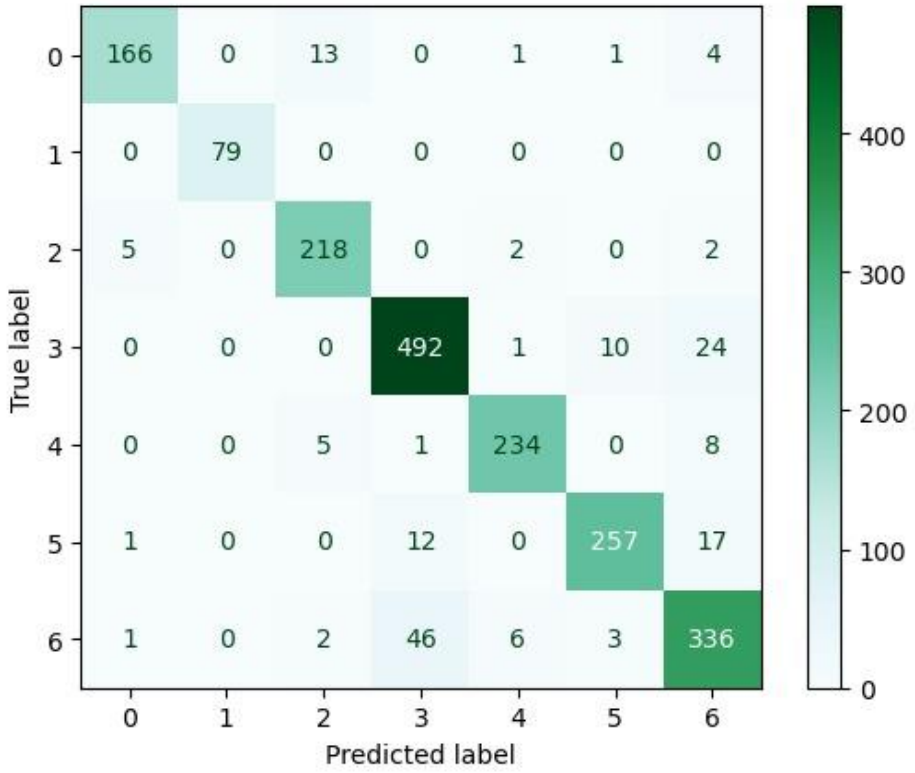


2.4 K-Nearest Neighbor + LDA

The best parameters for the model were quickly found using grid search:

- N neighbors: 21
- Metric: euclidean
- Time elapsed: 1.41 seconds
- Accuracy: 0.91

Label	Precision	Recall	F1-score
0	0.96	0.90	0.93
1	1	1	1
2	0.92	0.96	0.94
3	0.89	0.93	0.91
4	0.96	0.94	0.95
5	0.95	0.90	0.92
6	0.86	0.85	0.86



3. Random Forest

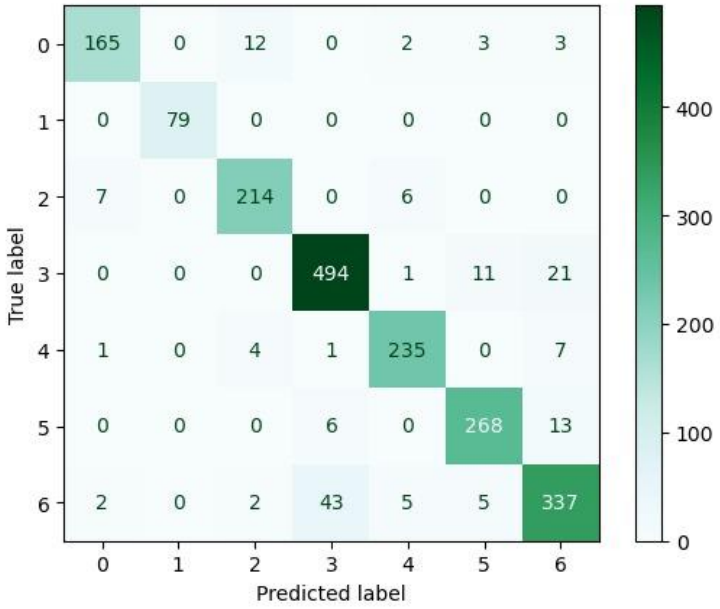
Random Forest is an ensemble machine learning algorithm that constructs multiple decision trees during training and combines their outputs to improve predictive accuracy and robustness. Each decision tree is built using a random subset of the training data and a random subset of features at each split, a process known as *"bootstrap aggregating"* or *"bagging"*. Random Forest reduces overfitting, increases accuracy, and handles large datasets with higher dimensionality more effectively than individual decision trees.

3.1 Random Forest without any dimensionality reduction technique

For the process of choosing a suitable set of hyperparameters for our Random Forest models, we made use of the grid search. The best parameters for the model were quickly found using grid search:

- N estimators: 175
- Max features: none
- Max depth: 20
- Min sample split: 10
- Min sample leaf: 2
- Class weight: balanced
- Time elapsed: 181.02 seconds
- Accuracy: 0.92

Label	Precision	Recall	F1-score
0	0.94	0.89	0.92
1	1	1	1
2	0.92	0.94	0.93
3	0.91	0.94	0.92
4	0.94	0.95	0.94
5	0.94	0.94	0.94
6	0.89	0.86	0.87

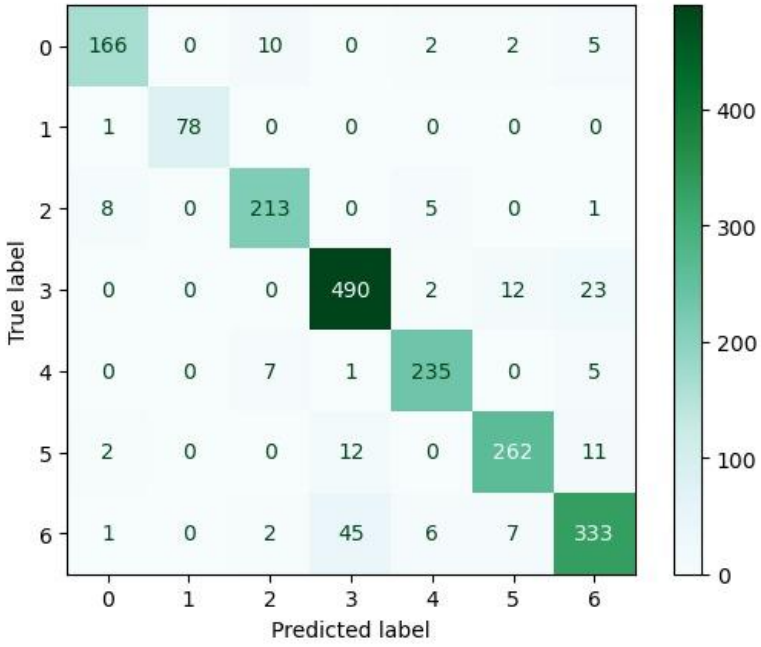


3.2 Random Forest + PCA

The best parameters for the model were quickly found using grid search:

- N estimators: 175
- Max features: none
- Max depth: 20
- Min sample split: 2
- Min sample leaf: 1
- Class weight: balanced
- Time elapsed: 57.28 seconds
- Accuracy: 0.91

Label	Precision	Recall	F1-score
0	0.94	0.89	0.91
1	1	1	1
2	0.92	0.94	0.93
3	0.89	0.93	0.91
4	0.94	0.95	0.94
5	0.93	0.91	0.92
6	0.88	0.84	0.86

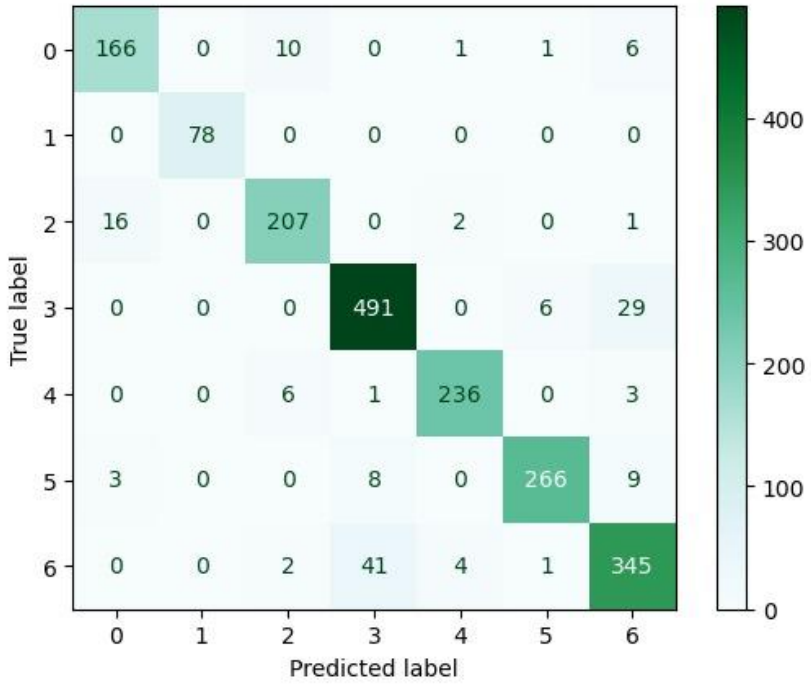


3.3 Random Forest + LVS

The best parameters for the model were quickly found using grid search:

- N estimators: 175
- Max features: none
- Max depth: 20
- Min sample split: 10
- Min sample leaf: 2
- Class weight: balanced
- Time elapsed: 92.16 seconds
- Accuracy: 0.92

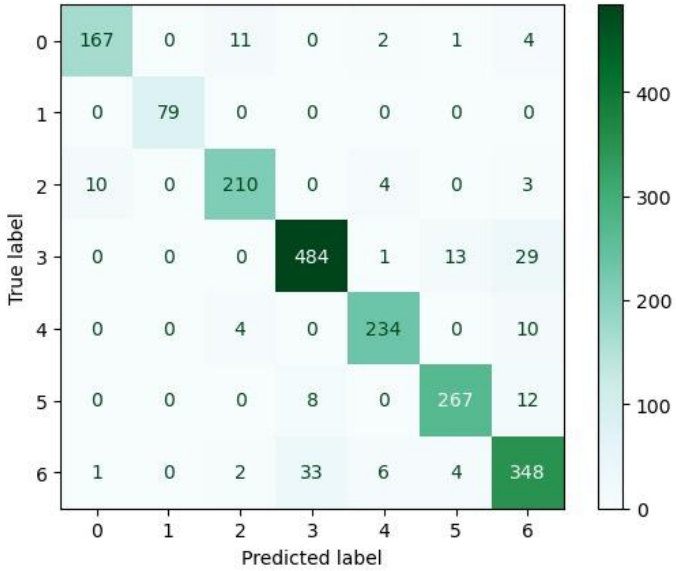
Label	Precision	Recall	F1-score
0	0.95	0.88	0.91
1	1	1	1
2	0.91	0.93	0.92
3	0.92	0.94	0.93
4	0.94	0.94	0.94
5	0.95	0.93	0.94
6	0.87	0.87	0.87



3.4 Random Forest + LDA

The best parameters for the model were quickly found using grid search:

- N estimators: 100
- Max features: none
- Max depth: 10
- Min sample split: 5
- Min sample leaf: 1
- Class weight: balanced
- Time elapsed: 48.80 seconds
- Accuracy: 0.92



Label	Precision	Recall	F1-score
0	0.95	0.90	0.93
1	1	1	1
2	0.92	0.94	0.93
3	0.92	0.93	0.92
4	0.95	0.94	0.95
5	0.94	0.92	0.93
6	0.86	0.87	0.86

4. Support Vector Machines (SVM)

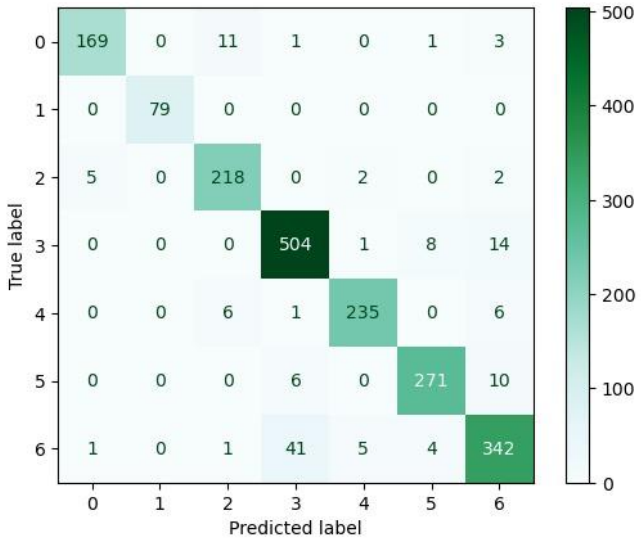
At its core, an SVM aims to find the optimal hyperplane that maximally separates the data points of different classes in a dataset. The hyperplane is a decision boundary that helps classify new data points. In two-dimensional space, this hyperplane is simply a line, but in higher dimensions, it becomes a plane or a hyperplane.

4.1 Support Vector Machines without any dimensionality reduction technique

The first experiment involved applying the SVM model to the scaled data without any dimensionality reduction applied to it. The best parameters for the model were quickly found using grid search:

- C: 100
- Gamma: 0.01
- Kernel: rbf
- Time elapsed: 66.72 seconds
- Accuracy: 0.93

Label	Precision	Recall	F1-score
0	0.97	0.91	0.94
1	1	1	1
2	0.92	0.96	0.94
3	0.91	0.96	0.93
4	0.97	0.95	0.96
5	0.95	0.94	0.95
6	0.91	0.87	0.89

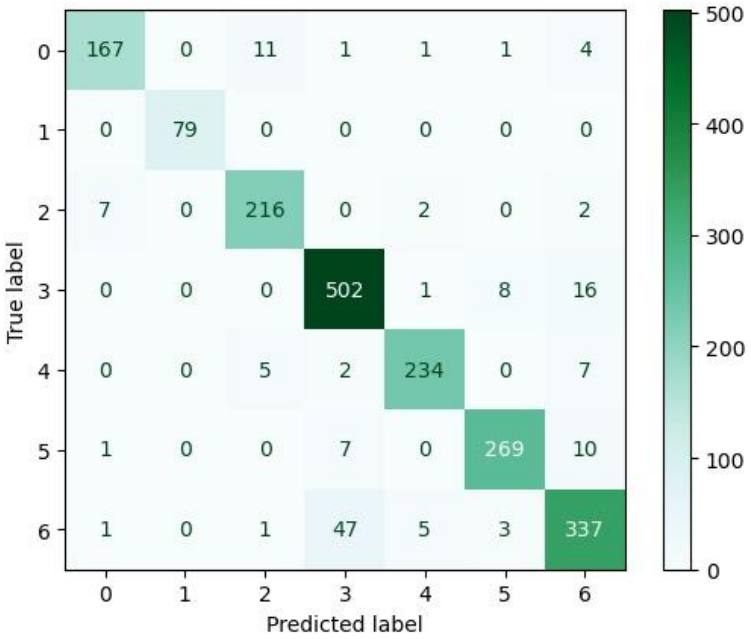


4.2 Support Vector Machines + PCA

The best parameters for the model were quickly found using grid search:

- C: 10
- Gamma: 0.1
- Kernel: rbf
- Time elapsed: 70.51 seconds
- Accuracy: 0.92

Label	Precision	Recall	F1-score
0	0.95	0.90	0.93
1	1	1	1
2	0.93	0.95	0.94
3	0.90	0.95	0.93
4	0.96	0.94	0.95
5	0.96	0.94	0.95
6	0.89	0.86	0.87

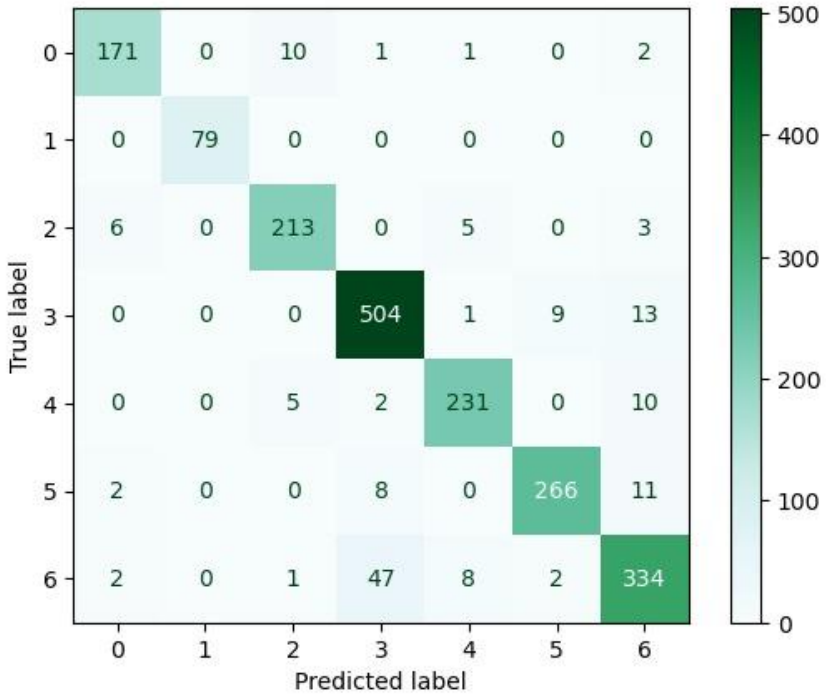


4.3 Support Vector Machines + LVS

The best parameters for the model were quickly found using grid search:

- C: 100
- Gamma: 0.01
- Kernel: rbf
- Time elapsed: 66.27 seconds
- Accuracy: 0.92

Label	Precision	Recall	F1-score
0	0.94	0.92	0.93
1	1	1	1
2	0.93	0.94	0.93
3	0.90	0.96	0.93
4	0.94	0.93	0.94
5	0.96	0.93	0.94
6	0.90	0.85	0.87

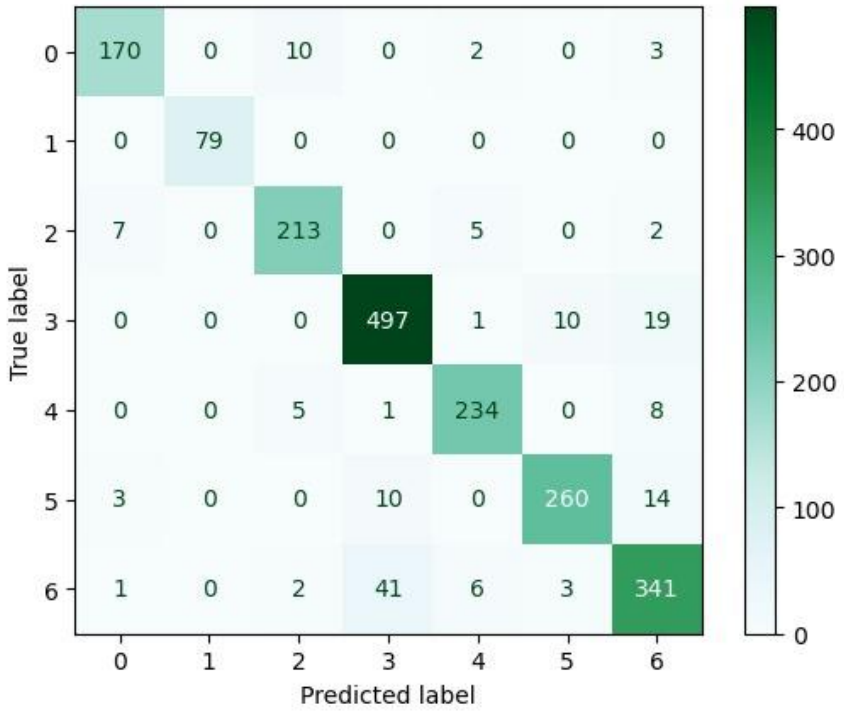


4.4 Support Vector Machines + LDA

The best parameters for the model were quickly found using grid search:

- C: 10
- Gamma: 0.1
- Kernel: rbf
- Time elapsed: 47.49 seconds
- Accuracy: 0.92

Label	Precision	Recall	F1-score
0	0.94	0.92	0.93
1	1	1	1
2	0.93	0.94	0.93
3	0.91	0.94	0.92
4	0.94	0.94	0.94
5	0.95	0.91	0.93
6	0.88	0.87	0.87



5. Naïve Bayes

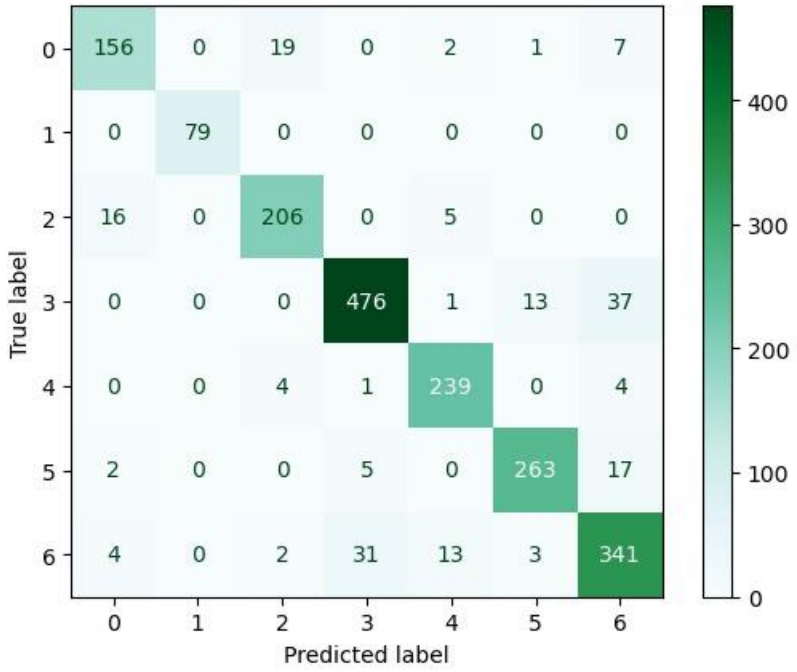
Naïve Bayes is a classification algorithm based on Bayes 'Theorem, which is used in machine learning and statistics. The "naïve" part comes from the assumption that all the features in the dataset are independent of each other, which is rarely true in real-world data, but it simplifies the computation significantly.

5.1 Naïve Bayes without any dimensionality reduction technique

The first experiment involved applying the Bayesian model to the scaled data without any dimensionality reduction applied to it. For this model, we do not need to apply a grid search, because there are no hyperparameters that need to be tuned.

- Accuracy: 0.90

Label	Precision	Recall	F1-score
0	0.88	0.84	0.86
1	1	1	1
2	0.89	0.91	0.90
3	0.93	0.90	0.92
4	0.92	0.96	0.94
5	0.94	0.92	0.93
6	0.84	0.87	0.85

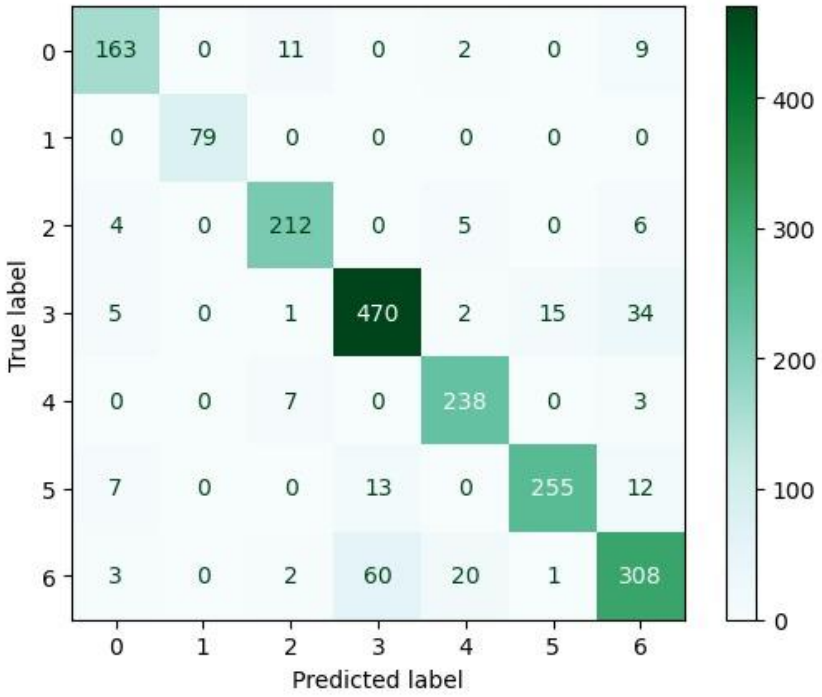


5.2 Naïve Bayes + PCA

The first dimensionality reduction technique applied to our data was Principal Component Analysis.

- Accuracy: 0.89

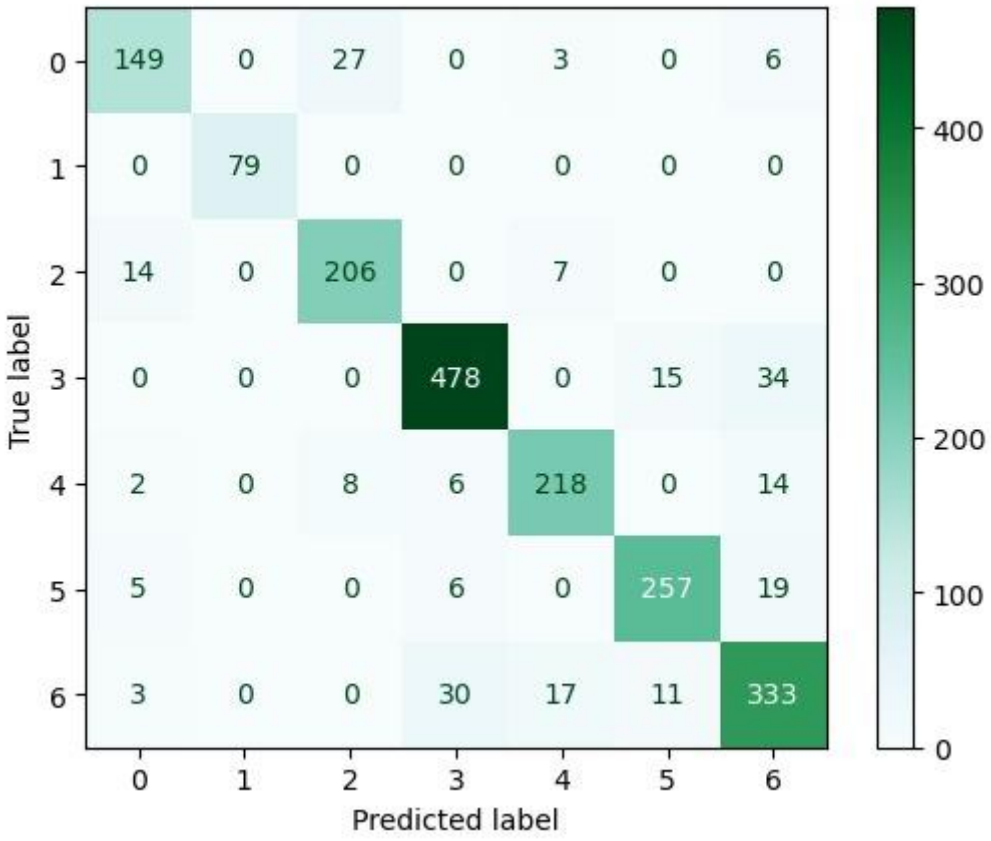
Label	Precision	Recall	F1-score
0	0.90	0.88	0.89
1	1	1	1
2	0.91	0.93	0.92
3	0.87	0.89	0.88
4	0.89	0.96	0.92
5	0.94	0.89	0.91
6	0.83	0.78	0.80



5.3 Naïve Bayes + LVS

- Accuracy: 0.88

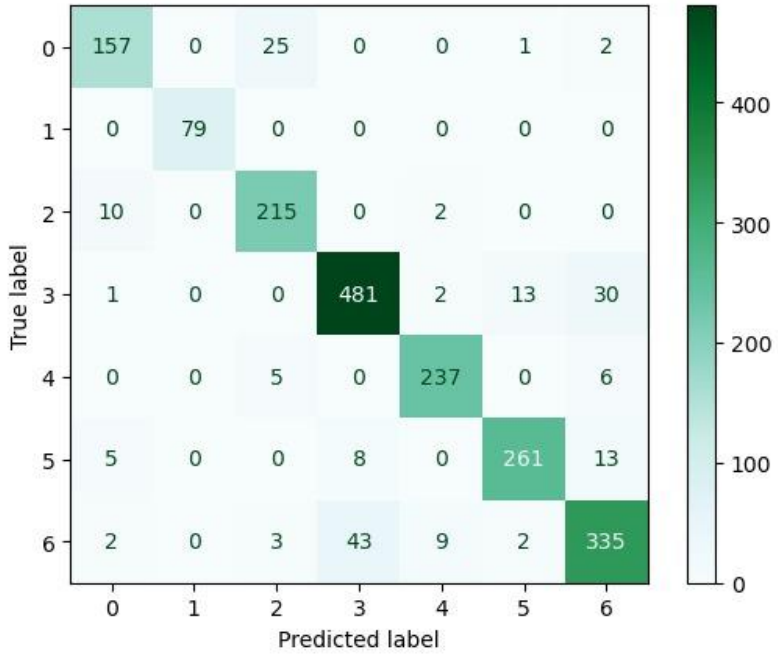
Label	Precision	Recall	F1-score
0	0.86	0.81	0.83
1	1	1	1
2	0.85	0.91	0.88
3	0.92	0.91	0.91
4	0.89	0.88	0.88
5	0.91	0.90	0.90
6	0.82	0.85	0.83



5.4 Naïve Bayes + LDA

- Accuracy: 0.91

Label	Precision	Recall	F1-score
0	0.90	0.85	0.87
1	1	1	1
2	0.87	0.95	0.91
3	0.90	0.91	0.91
4	0.95	0.96	0.95
5	0.94	0.91	0.93
6	0.87	0.85	0.86



VI. CONCLUSIONS

As we could see throughout this project, we obtained great results with all the algorithms we have worked with. For a better understanding and visualization, we created a table showing the weighted average for each model.

	Weighted avg. Precision	Weighted avg. Recall	Weighted avg. F1 score
LR – no dim.	0.94	0.94	0.94
KNN – no dim.	0.93	0.93	0.93
RF – no dim.	0.92	0.92	0.92
SVM – no dim.	0.93	0.93	0.93
NB – no dim.	0.90	0.90	0.90
LR – PCA	0.93	0.93	0.93
KNN – PCA	0.92	0.92	0.92
RF – PCA	0.91	0.91	0.91
SVM – PCA	0.93	0.93	0.93
NB – PCA	0.89	0.89	0.89
LR – LVS	0.94	0.94	0.94
KNN – LVS	0.93	0.93	0.93
RF – LVS	0.92	0.92	0.92
SVM – LVS	0.93	0.93	0.93
NB – LVS	0.90	0.90	0.90
LR – LDA	0.93	0.93	0.93
KNN – LDA	0.92	0.92	0.92
RF – LDA	0.92	0.92	0.92
SVM – LDA	0.92	0.92	0.92
NB – LDA	0.91	0.91	0.91

Our results demonstrate significant improvements when applying various dimensionality reduction techniques such as PCA (Principal Component Analysis), LVS (Lasso Variable Selection) and LDA (Linear Discriminant Analysis), compared to when no such techniques were employed. These methods helped in simplifying the feature space, which in turn facilitated more effective training of our models.

Because this is an important criterion, we also added the computer's specifications on which we ran all the code:

- Intel core I7-1255U (12th generation), 1.70 GHz
- 6 GB RAM
- Windows 11 Home
- Intel Iris Xe Graphics

For a better visualization, we also added a table showing the elapsed times for each combination of algorithm and dimensionality reduction technique:

	No dim. red.	PCA	LVS	LDA
LR	36.68	7.76	19.74	12.35
KNN	5.27	1.91	1.80	1.41
RF	181.02	57.28	92.16	48.80
SVM	66.72	70.51	66.27	47.49
NB	-	-	-	-

Without shuffling the data before applying these techniques, our results would have been suboptimal. This is because the data originally had features in ascending order, which introduced biases and inconsistencies in the training process, leading to poor model performance. Shuffling ensured a more representative distribution of the data, allowing the models to generalize better.

By transforming the feature space, PCA helped in capturing the most variance, LVS focused on selecting the most relevant features and LDA aimed at maximizing class separability. These transformations enhanced the efficiency of our classification algorithms.

The classification models used in this project - Logistic Regression, K-Nearest Neighbors, Random Forest, Support Vector Machine and Naive Bayes—each benefited from the reduced and more relevant feature sets.

Overall, the combined approach of data shuffling, dimensionality reduction and the application of diverse classification algorithms led to robust and high-performing models. The synergy between these techniques ensured that we extracted the most meaningful patterns from the data, resulting in superior predictive performance across various classifiers. Future work could explore the integration of other dimensionality reduction methods and advanced ensemble techniques to further enhance model accuracy and generalization.