Cmpe462 Project2

Hikmet Can Köseoğlu

Dataset

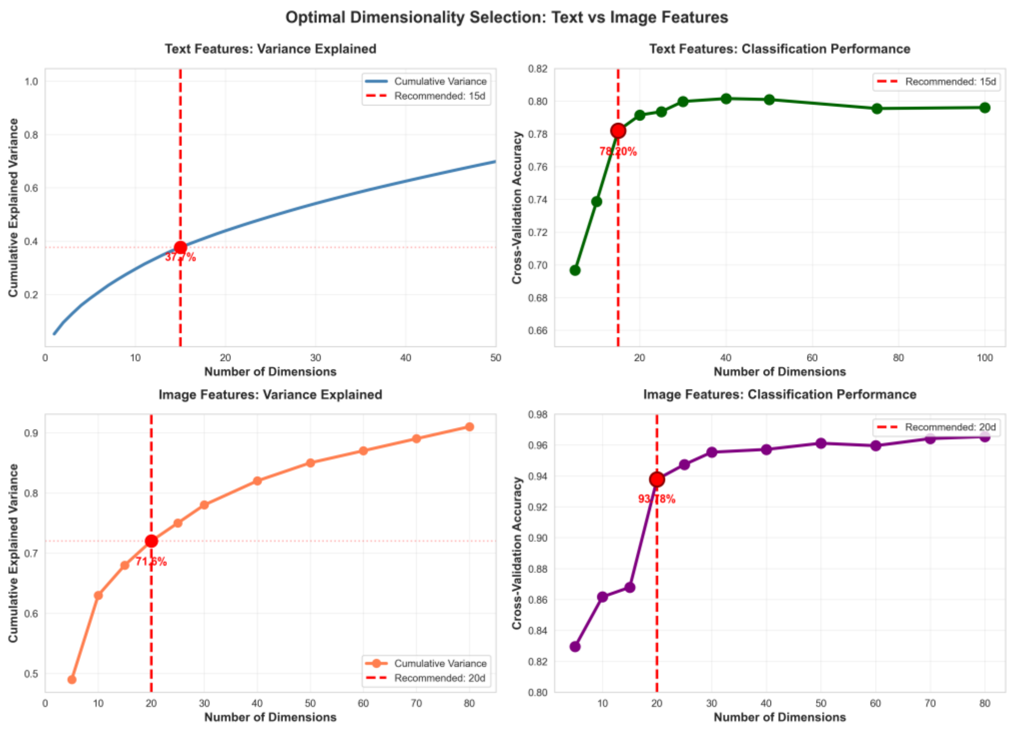
I have mainly made 2 modifications on top of the dataset for the previous project:

1- Dimensionality reduction
2- Addition of incorrectly labeled data

Compared to the number of samples our model had relatively many features, most of them being the text and image features. The abundance of image features caused the model to memorize details that are not necessarily related to the task of fruit classification. This in return caused the model to generalize poorly. Hence I decreased the dimensions of image and text features as follows:

Dimension of image features: 96 => 20
Dimension of text features: 100 => 15

I have made the decision of dimensionality reduction based on elbow technique; I observed the model performance (logistic regression with using single modality) based on accuracy with varying dimensions for different number of features:



The performance of the model stops the substantial improvement after the given dimensions.

Secondly I have added false labeled data, 5 per each category. From each category 5 random samples were selected and their labels were changed to another category. This helped me to observe the specifics of the outlier detection framework and my observation of different support vectors.

After these classifications we have in total 3349 samples:
Apple: 1037,
Banana: 527,
Lemon: 493,
Mandarin: 765,
Orange: 527

Total Dimensions of 50:
Text features: 15,
Numerical features: 4,
Categorical features: 11 (with one hot encoding),
Image features: 20

Please note that our dataset has high intra-class similarity which is a product of the heavy data augmentations used. Another factor is the image taking process during dataset creation: as we took photos of fruits with exactly the same background, angle, camera etc.
Hence most of the test and training results are similar, as the test set doesn't really introduce that much challenge.

Task1: Classification

1.

Sklearn library was used for training the models in this part. Following table depicts the performance and training time of different models:

Classifier	Time (s)	Training Acc.	Test Acc.	Test F1
Logistic Regression	0.016	0.998	0.997	0.997
Logistic Regression (non-linear transformation)	0.366	0.994	0. 982	0. 982
Soft-margin SVM	0.039	0.993	0. 988	0.988
Soft-margin SVM (kernel trick)	0.058	0.993	0.985	0.985

KNN	-	0.994	0.983	0.983
Naïve Bayes	-	0.922	0.925	0.924
Random Forest	0.512	0.994	0.983	0.983

Some comments on the classification choices and hyperparameters:

Please note that the hyperparameters are selected purely on trial and error rather than grid search or random search (these would be overkill).

The C value is selected such that the model neither overfits nor underfits. Too high value of a C causes overfitting (lighter regularization) whereas too low C values causes underfitting (heavy regularization). For the logistic regression I picked C=0.1 as it creates a very small positive generalization gap which implies the model is learning well. The choice of "l2" loss and "lbfgs" as solve is used as a defacto standard.

For logistic regression with non-linear transformation I picked polynomial transformation. Using degree n=2 or n=3 yielded nearly the same result, so I went with degree 2 as it requires less dimensions hence requires less computation. RBF didn't provide the same performance as polynomial, even though varying gamma values helped increase accuracy. Large gamma provides increased locality whereas smaller values creates smoother boundaries.

In SVM the C value was picked with the same approach in logistic regression. However the loss function is "Hinge" loss which allows to penalize for margin violations.

In the SVM with kernel trick I used RBF as kernel, which allowed having non-linear boundaries without making the actual transformations to higher dimensions. I used the argument gamma="scale", which makes gamma inversely proportional with number of dimensions and variance of data.

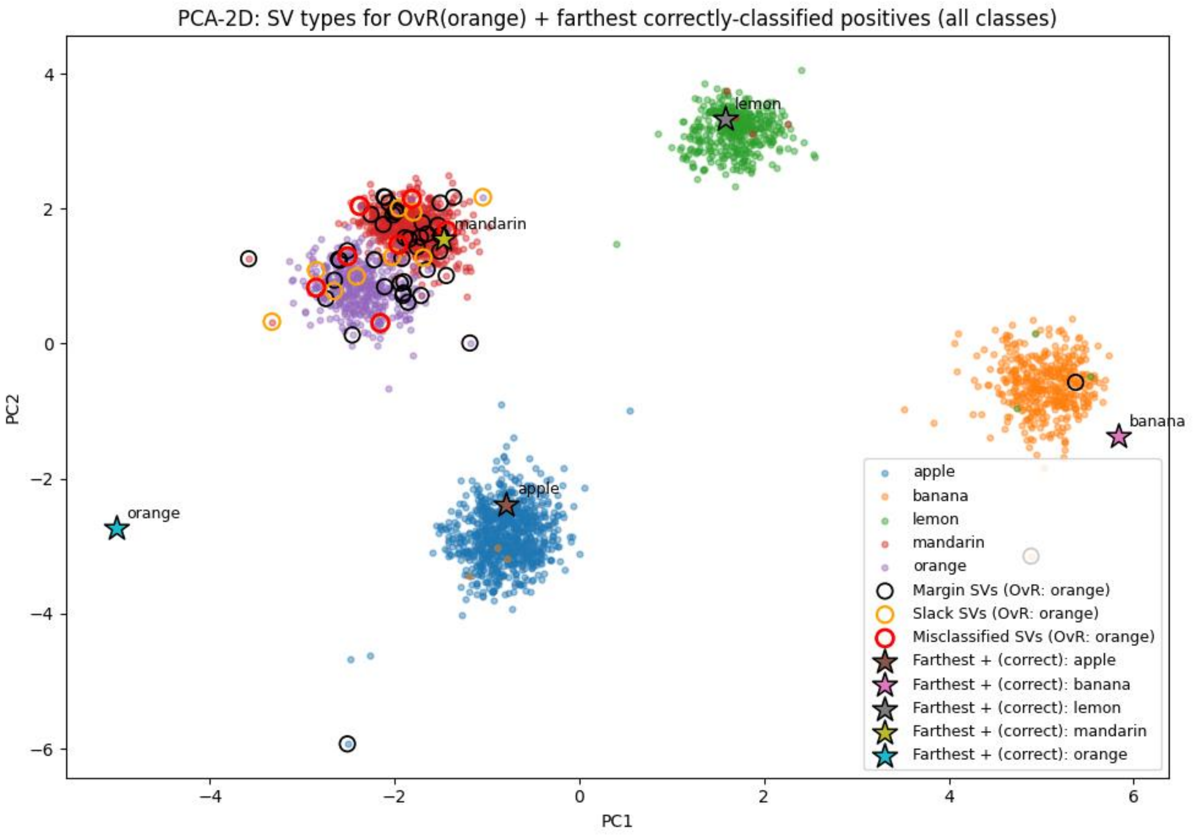
In kNN I picked n_neighbors=15, which yielded a good performance. Decreasing it too much causes the model to overfit, whereas too high values causes underfitting. weights="distance" makes model more robust to variance in data. I used Euclidean distance as a distance metric which is the standard approach.

The "alpha" argument in Naïve bayes smoothes the close to zero probabilities.

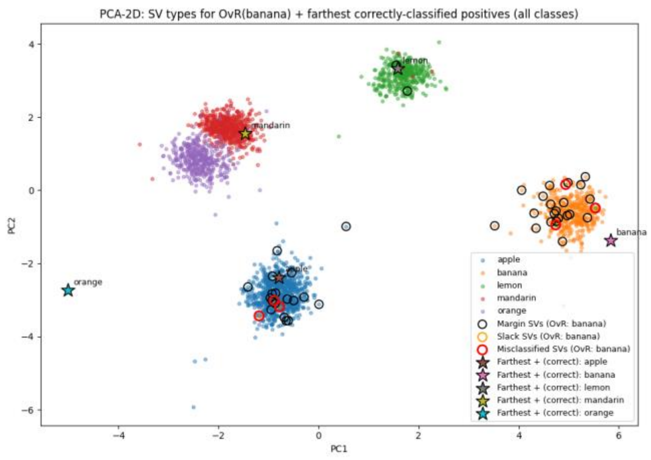
"n_estimators" is set to ensure stable performance which in return training time. Depth and minimum-sample constraints regularize individual trees, and max_features increases tree diversity to reduce variance.

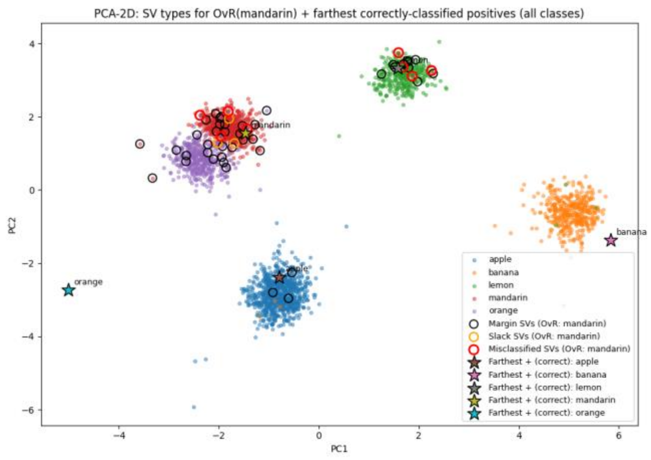
2.a

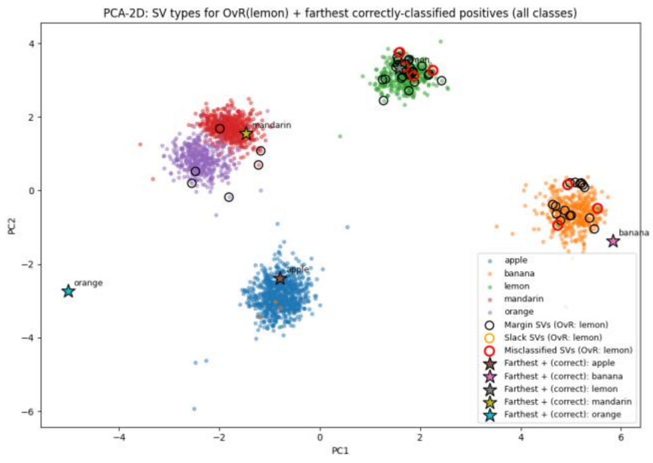
The following visualizations provide the locations of support vectors (distinguished as margin, slack or regular support vector) for each OvR classifier, as well data points that were correctly classified and located furthest away from their OvR classifier. Each visualization is for one of the OvR classifiers:

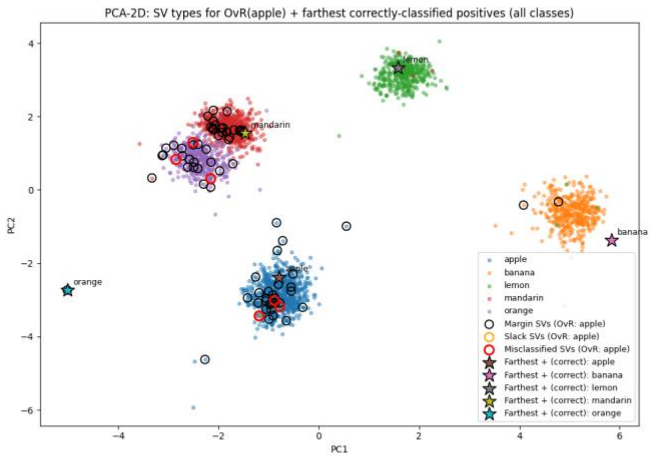


It can be seen that the support vectors are concentrated along the decision boundary (in the case of the orange OvR classifier between orange and mandarin) which indicates they are the samples that play the most significant role for determining the decision boundary. The farthest points are typically located far away from the decision boundary, more close to the class cluster (as can bee seen in apple, lemon and mandarin), hence they are classified confidently and have zero or little contribution to determining the class boundaries. Please note that some support vectors are located at points that doesn't really make sense on our visuals, like in the middle of a cluster, this is because of the dimensionality reduction used for visualizations: we have a high dimensional data, however we try to represent it in 2D; so I believe it is okay to see some distortions (like orange's farthest point being off the cluster).





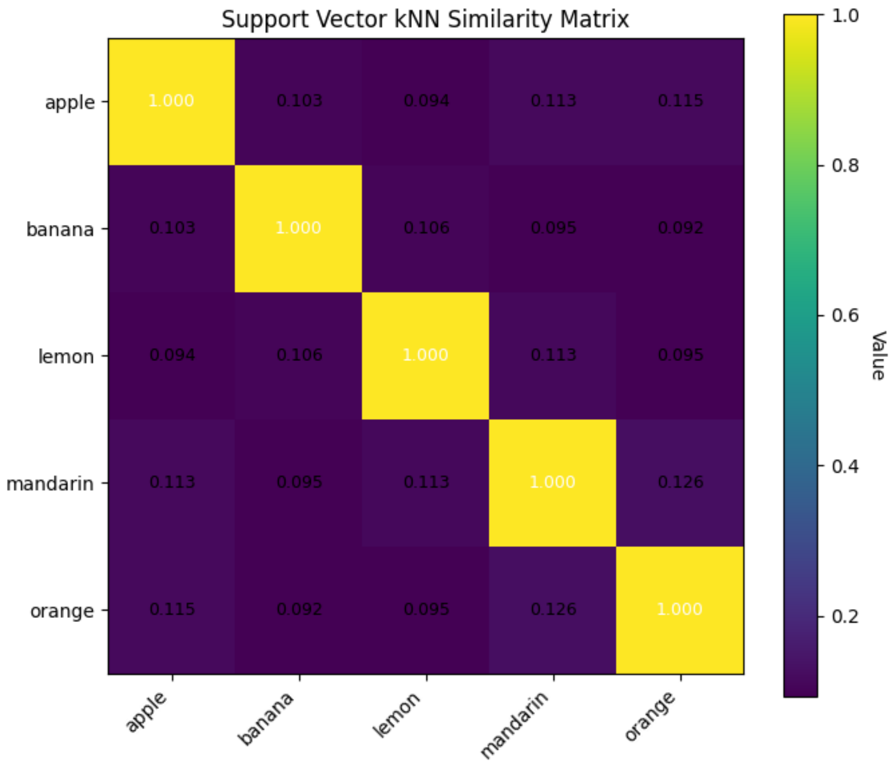


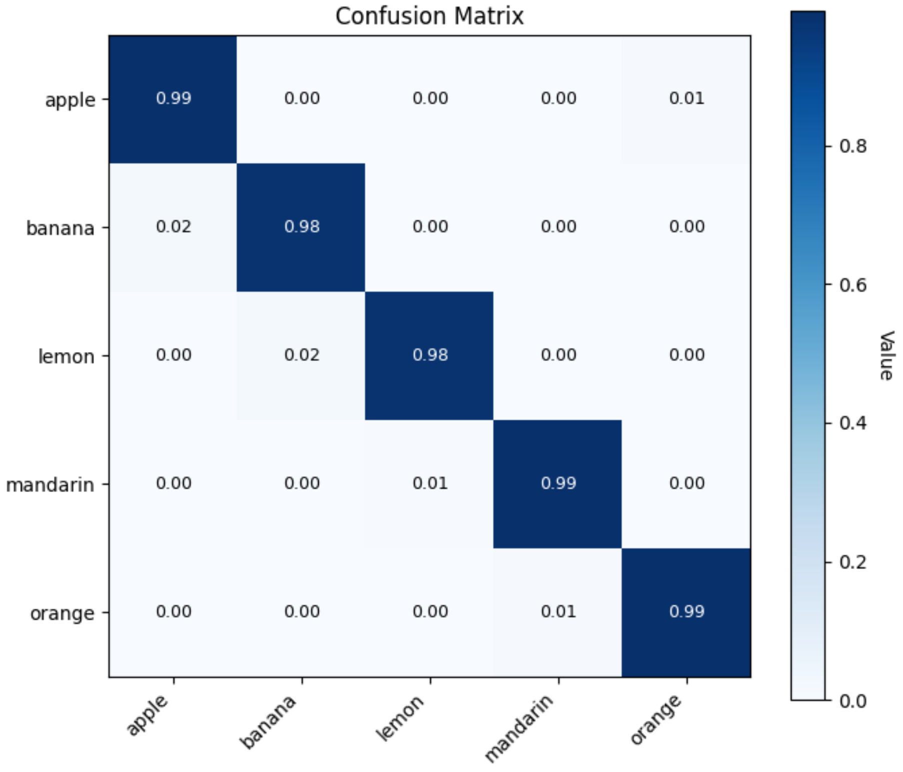


2.b

I have compared the confusion matrix and the kNN similarity matrix of support vectors of different OvR classifiers. The highest similarity in terms of support vectors are mandarin-orange and mandarin-lemon. We can observe the non-zero confusion values in the confusion matrix as opposed to most of the other pairs (which has lower kNN similarity values). The correlation between confusion and support vector similarity lies under the nature of SVs. They are the samples that determine the decision boundaries. If the decision boundaries are closer, it is natural to observe more confusion.

3.
From the trained models we can deduce that our dataset is linearly separable:
We get the highest accuracy with Logistic Regression, the other non-linear methods like Naïve Bayes yield lower performance. Also applying non-linear transformation in logistic regression decreased model's performance. Finally it is possible to observe the linear separability nature of our dataset with the pca2d projection of our dataset.


Support Vector kNN Similarity Matrix
apple
banana
lemon
mandarin
orange
1.000 0.103 0.094 0.113 0.115
0.103 1.000 0.106 0.095 0.092
0.094 0.106 1.000 0.113 0.095
0.113 0.095 0.113 1.000 0.126
0.115 0.092 0.095 0.126 1.000
apple
banana
lemon
mandarin
orange
1.0
0.8
0.6
0.4
0.2
Value


Confusion Matrix
apple
banana
lemon
mandarin
orange
0.99 0.00 0.00 0.00 0.01
0.02 0.98 0.00 0.00 0.00
0.00 0.02 0.98 0.00 0.00
0.00 0.00 0.01 0.99 0.00
0.00 0.00 0.00 0.01 0.99
apple
banana
lemon
mandarin
orange
0.8
0.6
0.4
0.2
0.0
Value

4.a

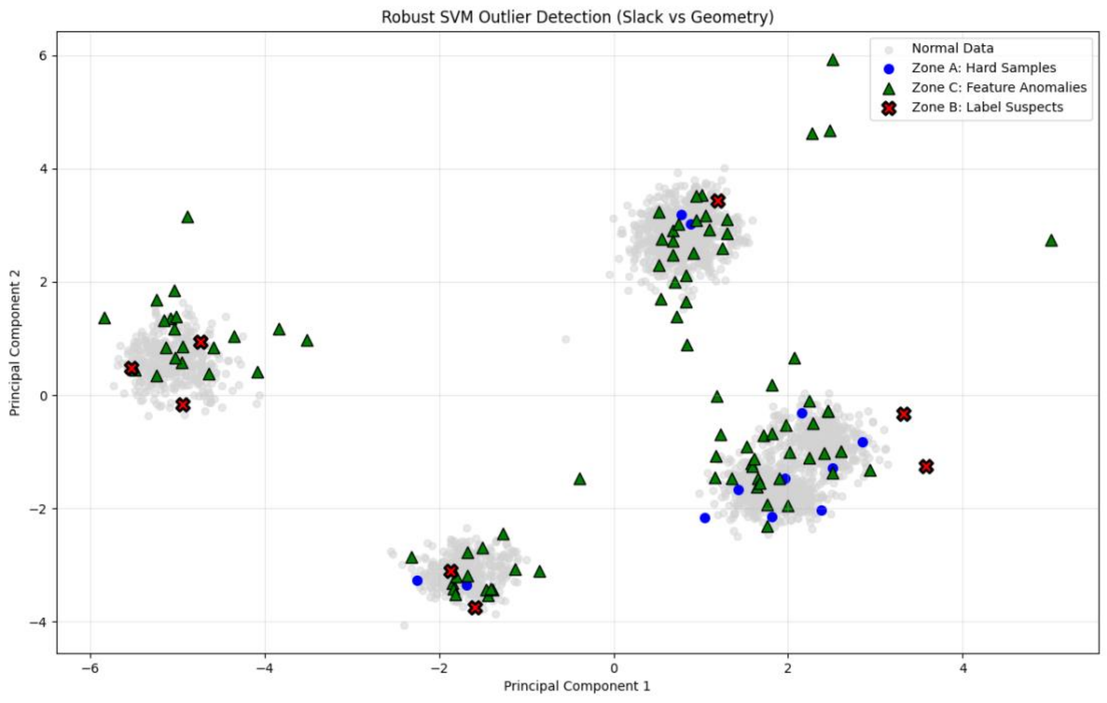
My outlier detection framework is based on two metrics, the slacks corresponding to samples and the distance to each class' centroid.

Value of slack directly tells us the classification result and how hard was it for the model to classify this sample. If the slack value is bigger than 1, then we can deduce the sample was misclassified.

The distance to class centroid tells us about the similarity of individual samples and the overall class representation. If a given sample's distance to class centroids is more than class standard deviation times a threshold than we evaluate this sample as it has a bad geometry.

Based on these metrics there are 3 different categorizations for data points:

1- Hard Samples: these samples have high slack values, but are close to the class cluster. Even though they were misclassified, they are not outliers but some hard examples that svm failed to classify.
2- Incorrect Labels: these samples have high slack values (misclassified) and bad geometry (far away from rest of the class). It is likely that they were labeled incorrectly.
3- Feature anomalies: These data points have normal slack values (classified correctly) but geometric abnormalities (far away from the rest of the class samples). It is likely that this is a true outlier that has unusual features.



Number of hard samples detected: 12
Number of incorrect label points detected: 8
Number of anomaly points detected: 93

In the framework, 1.5 standard deviations is used as the threshold:
I started with threshold of 3 std but it wasn't performing well on the detection of the incorrect labels. Now it has a decent performance of detecting 6 of the 15 incorrect labels (the incorrect labels are mentioned in the first section "Dataset"). Please note that the rest of 8 incorrect label points (8-6=2 samples) are false positives.

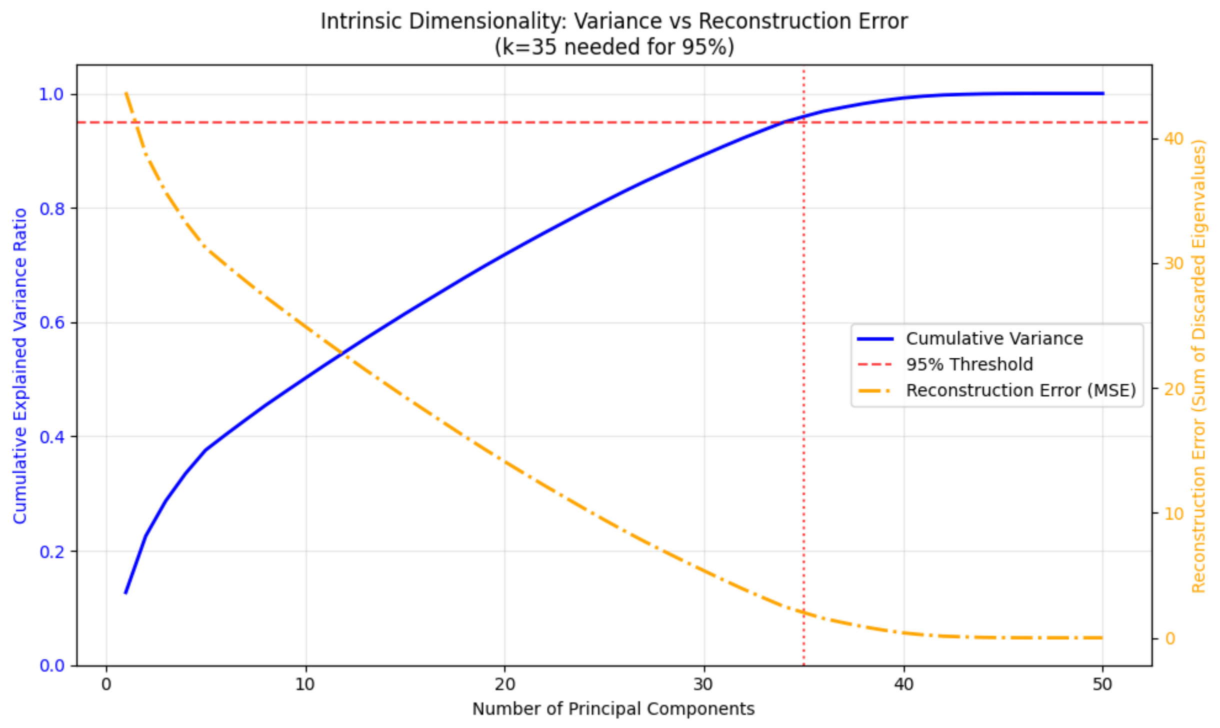
Increasing it further might improve incorrect label detection performance but also increases the detected number of feature anomalies so something between 1.5 and 2 seemed fine. Currently 32 out of 60 outliers with feature anomalies are detected (these feature anomalies were mentioned in the project1 report). Please note that the rest of 93 anomaly points (93-32 = 61 samples) are false positives.

4b.

Link to the video description of the framework: https://drive.google.com/file/d/1R2bTjV9UmJ-piC_M8VjmT_7M5AQkqvRl/view?usp=sharing.

Task1: Unsupervised Learning

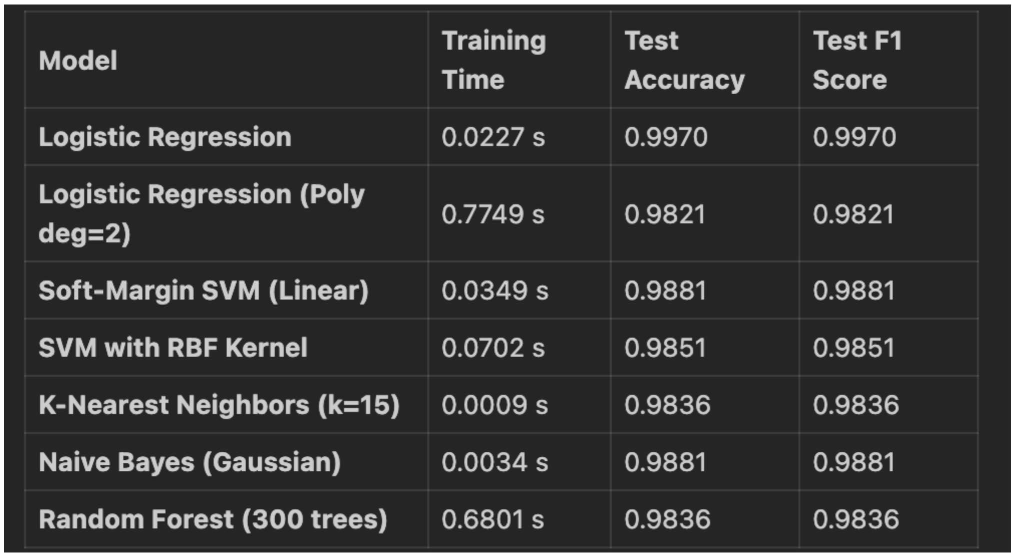
1.



I have aimed for %95 explained variance for dimensionality reduction, I picked such a high number as I already have relatively few number of features, and wanted to keep as much elements that kept the meaning in the data as possible.

Here are the results that can be observed in the figure:
Dimensions needed for 95% Variance: 35
Reconstruction Error (MSE) at k=35: 2.02

The following outputs are gained with dimensionality reduction to 35 from 50



Model	**Training Time**	**Test Accuracy**	**Test F1 Score**
Logistic Regression	0.0227 s	0.9970	0.9970
Logistic Regression (Poly deg=2)	0.7749 s	0.9821	0.9821
Soft-Margin SVM (Linear)	0.0349 s	0.9881	0.9881
SVM with RBF Kernel	0.0702 s	0.9851	0.9851
K-Nearest Neighbors (k=15)	0.0009 s	0.9836	0.9836
Naive Bayes (Gaussian)	0.0034 s	0.9881	0.9881
Random Forest (300 trees)	0.6801 s	0.9836	0.9836

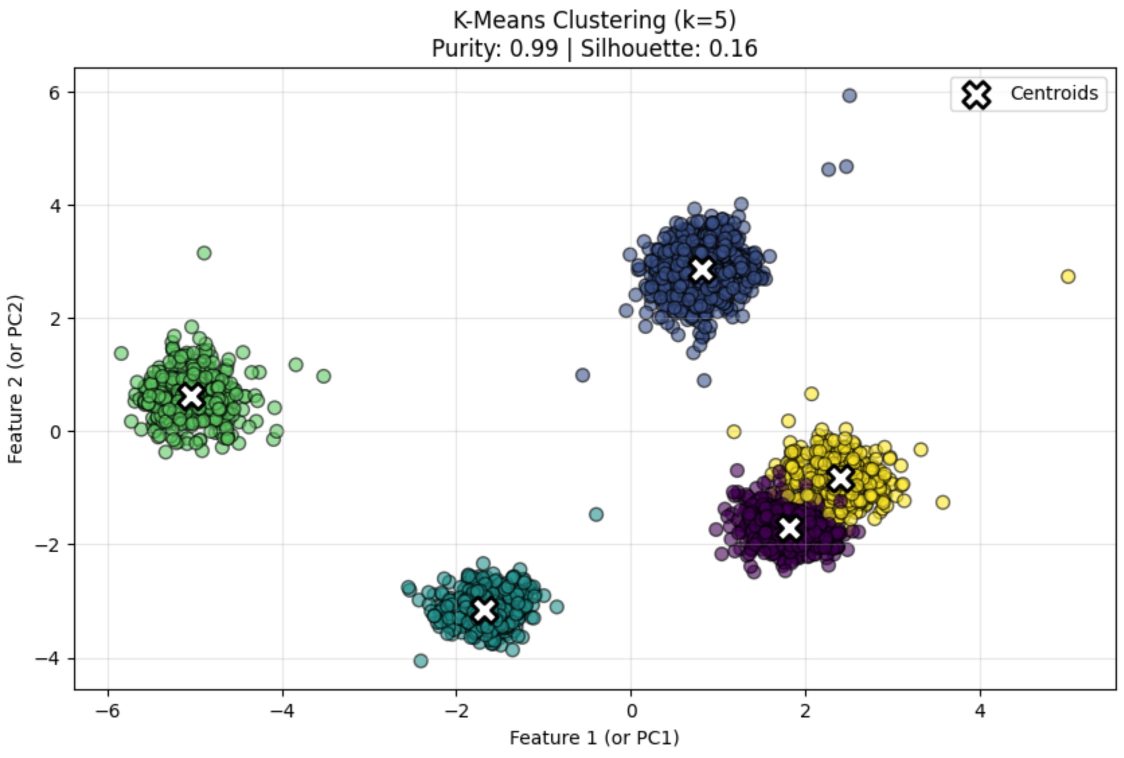
Actually, there are no significant performance improvement, this is highly like because of the already low number of features. The current features are already compactly representing the data, compressing it just a little further doesn't help in terms of representing the data better.

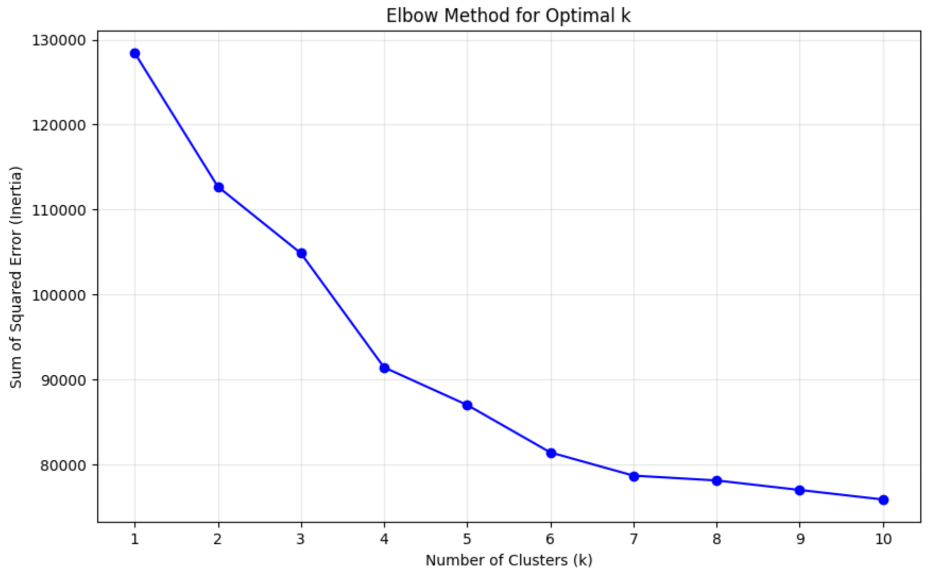
2.

I have used K-Means with k=5. For the clustering algorithm to be more robust PCA with dimensions=35 was applied beforehand. In the plot next page, you can observe the number of clusters versus SSE (Inertia); if we were to apply elbow method directly the method suggests clusters=4 is a better choice, however we know, in reality we have 5 clusters. Nevertheless, I got a pretty good purity score, I believe the plot (number of clusters vs sse) was the result of two clusters (likely mandarin and orange) being too close to each other. Here are the metrics for evaluating the clustering:

Silhouette Score: 0.1554
Purity Score: 0.9918

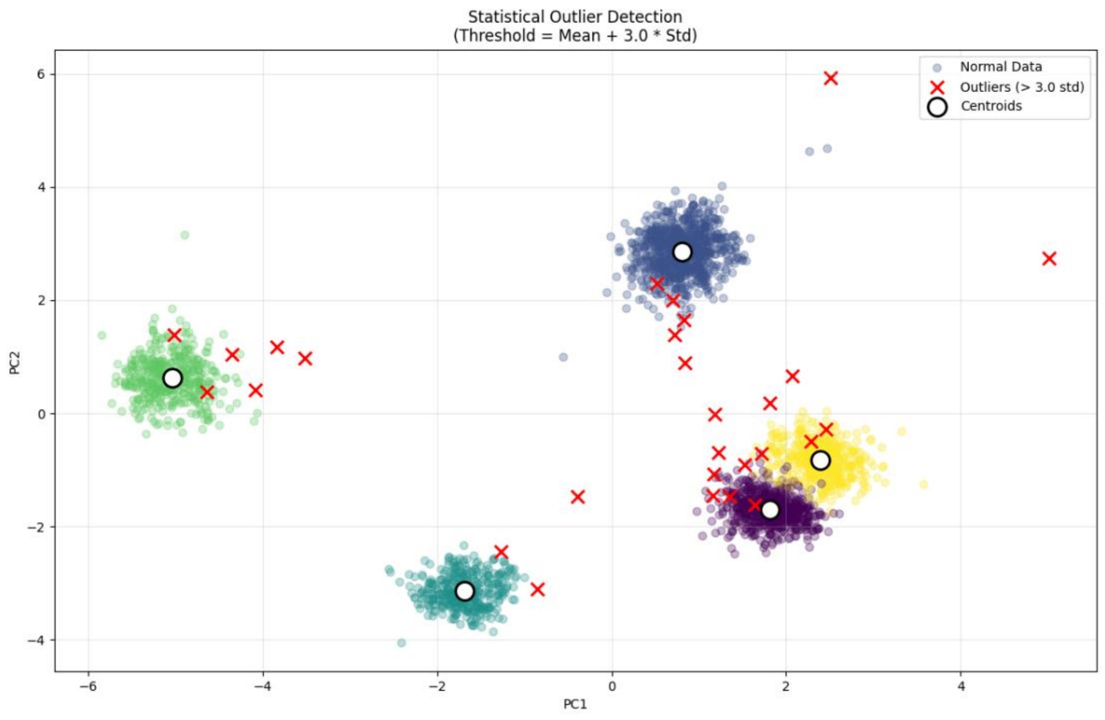
The purity score implies algorithm is successful in terms of matching similarities between samples. In contrast the relatively low purity score might be due to the geometry of the samples. Instead of perfect spheres the samples form a complex shape in high dimensions which decreases the score.





3.

For the outlier detection based on clustering, I used the distance of each sample to the mean of the cluster it was assigned. If the distance is bigger than standard deviation of the distance of that cluster times a threshold than that sample is an outlier. Please note that since different clusters have different standard deviations for the distances, each cluster has a unique outlier detection criteria: some clusters are dense and some are more sparse, hence each has different threshold distance for outliers.



I've compared this framework with the framework based on svm, the followings are the overlap results:
SVM-based outliers (anomaly+incorrect label): 101
Clustering-based outliers: 28
Overlap count: 28

Every point found by cluster-based outlier detection was also found by the svm-based outlier detection (now we can argue the points are outliers with higher confident). This points out that two combined together can make an even more robust outlier detection framework where clustering algorithm eliminates false positives.

Source Code and Dataset

You can access the source code and dataset from here:

https://github.com/hckoseoglu/Fruit-Classifier2
https://drive.google.com/file/d/1dGU1BCBufqMsxJAuvc5PWrisre9iuSWh/view?usp=sharing

Contribution

All the work for project 2, including the modification of dataset, completion of coding tasks, visualizations and reporting were done by Hikmet Can Köseoğlu

AI Statement

Here are the prompts that were used to benefit from AI resources during the making of the project:

Chatgpt: https://chatgpt.com/share/695696f6-1990-8000-bd48-defd2670fd93
Gemini: https://gemini.google.com/share/7fbf1b78f6c2