CMPE 462

Machine Learning

Assignment 2 Report

Harun Reşid Ergen 2019400141

Meriç Keskin 2019400024

Github Repository:

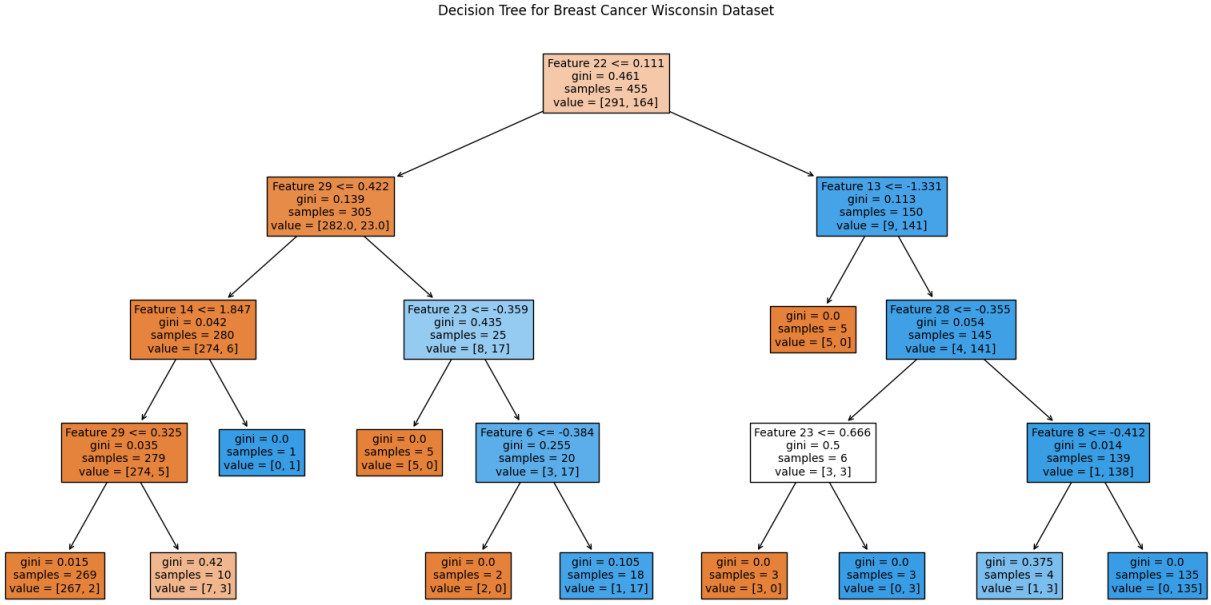
MericKeskin/CmpE462-Spring24 (github.com)

Decision Tree

1) Train a Decision Tree

- We loaded the dataset and split it into training and testing sets.
- The training set has 455 samples, and the test set has 114 samples.
- We tried different tree depths (from 1 to 20) to find the best depth.

The best depth was found to be 4, meaning the tree goes 4 levels deep:



2) Compare Performance

The Naive Bayes classifier had a test accuracy of 95.61%.

The Decision Tree classifier has a test accuracy of 92.11%.

The Naive Bayes classifier performs better on the test set compared to the Decision Tree classifier.

3) The Most Significant Features

Top 5 Features: 21, 28, 12, 22, 5

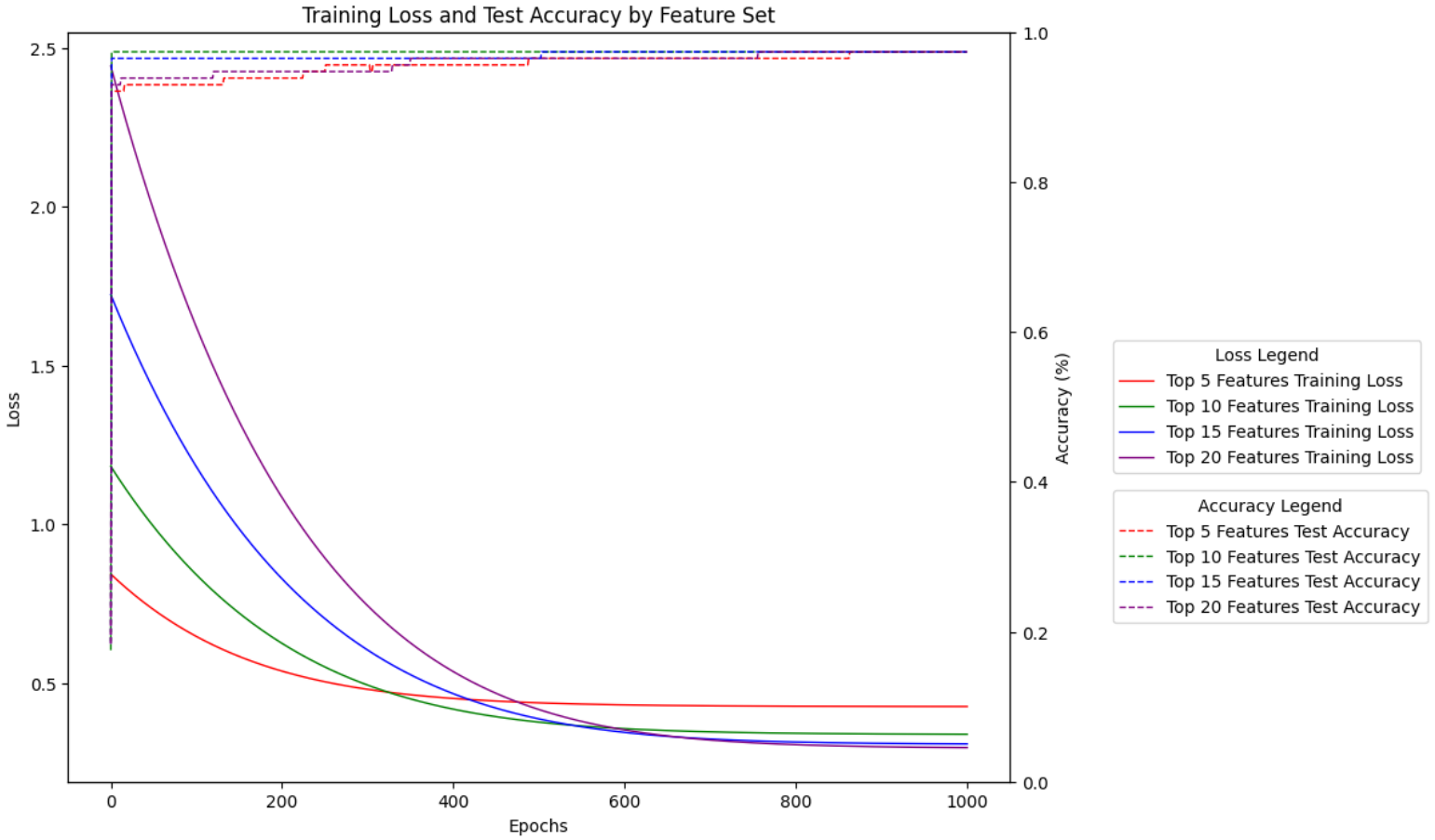
Top 10 Features: 21, 28, 12, 22, 5, 27, 13, 7, 4, 3

Top 15 Features: 21, 28, 12, 22, 5, 27, 13, 7, 4, 3, 2, 1, 6, 8, 9

Top 20 Features: 21, 28, 12, 22, 5, 27, 13, 7, 4, 3, 2, 1, 6, 8, 9, 10, 11, 30, 15, 14

- We used a logistic regression model for each set of top features.
- We plotted the training loss and test accuracy for each feature set.
- The plot shows how the loss decreases and accuracy increases over epochs.

(We got help from ChatGPT to draw the plots better)



- **Top 5 Features:**
  - Training Loss: Reducing steadily.
  - Test Accuracy: 97.4%.
- **Top 10 Features:**
  - Training Loss: Reducing steadily.
  - Test Accuracy: 97.4%.
- **Top 15 Features:**
  - Training Loss: Reducing steadily.

    - Test Accuracy: 97.4%.
- **Top 20 Features:**
    - Training Loss: Reducing steadily.
    - Test Accuracy: 97.4%.

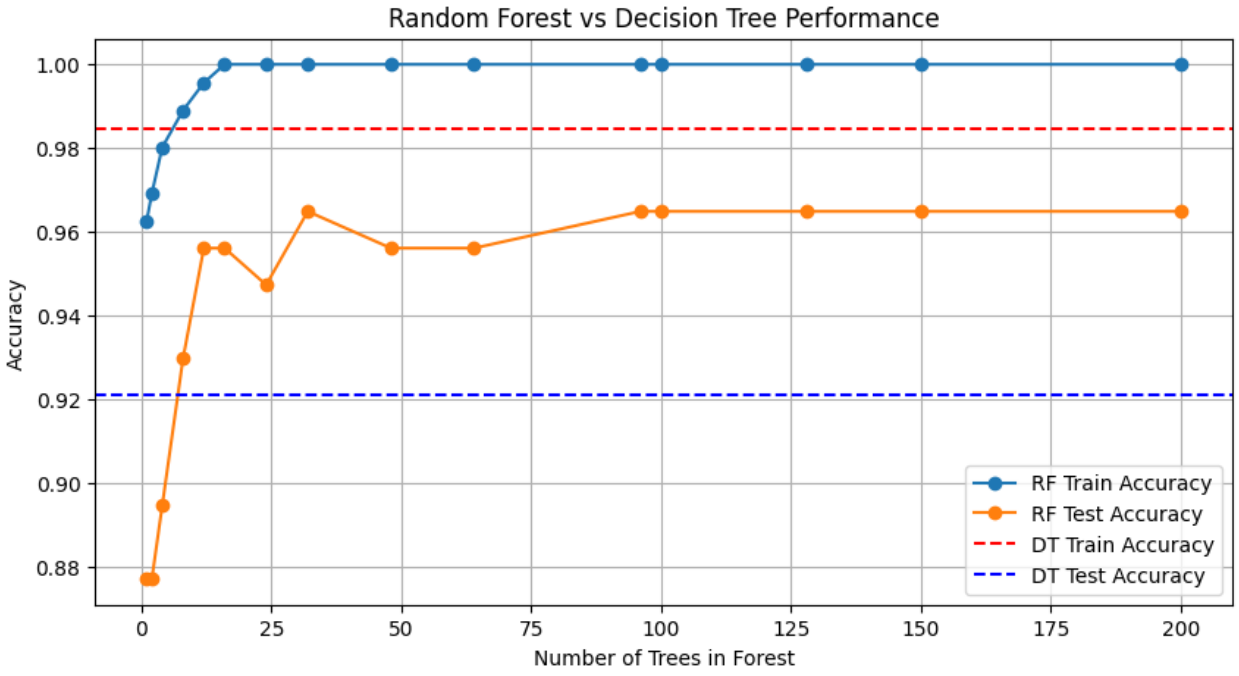
Comments:

- The models with top 10 and 20 features showed consistent high accuracy.
- Including more features generally helps, but after a certain point, the performance gains are minimal.
- At the end of the training, test accuracies reached the same point.

4) Random Forest

- We tried different number of trees [1, 2, 4, 8, 12, 16, 24, 32, 48, 64, 96, 100, 128, 150, 200].
- We measured the accuracy of the Random Forest on both the training and test sets.

(We got help from ChatGPT to draw the plots better)



- The Random Forest generally performed better or similarly to the Decision Tree, especially as the number of trees increased.
- The Decision Tree's test accuracy was 92.11%, while the Random Forest's test accuracy varied with the number of trees and stabilized around 97%.

Support Vector Machine

1) Soft Margin SVM

a) Linear Kernel, Soft Margin SVM

$$Q = \begin{bmatrix} I_{dxd} & 0 & 0 \\ 0 & \ddots & \vdots \\ 0 & 0 & (1e-6)_{NxN} \end{bmatrix}_{(d+1+N)\times(d+1+N)}$$

- The middle block of zeros is 1 x 1 as it corresponds to the scalar b.
- The $(1e-6)_{NxN}$ represents an NxN zero matrix, corresponding to slack variables. The reason of usage of a small value 1e-6 instead of zero is to make Q positive semi-definite.

$$p = (0 \quad \dots \quad C)$$

- p vector composed of $d+1$ zeros and N C

$$A = \begin{bmatrix} -y_1 x_1^T & -y_1 -1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ -y_N x_N^T & -y_N & 0 & \dots & -1 \\ 0 & \dots & 0 & -1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & \dots & -1 \end{bmatrix}_{2N\times(d+1+N)}$$

- Top right is identity NxN, bottom left is identity NxN
- Top left is corresponds to the w and b.
- Multiplied with -1 since cvxopt expects constraint as A*x <= c instead of A*x >= c

$$c = (-1 \quad \dots -1)$$

- *c* is a vector composed entirely of 1's with dimension Nx1, multiplied with -1 to make compatible for cvxopt.

[w1, …, wd, b, slack vars] = qp(Q, p, A, c)

Results:

	Training Accuracy	Test Accuracy	Time
C = 1	95.93 %	94.64%	79m 38s

b) sklearn's Linear Kernel, Soft Margin SVM

	Training Accuracy	Test Accuracy	Time
C = 0.01	94.58%	94.31%	1m 57s

C = 0.1	95.36%	94.61%	1m 46s
C = 1	95.97%	94.61%	5m 14s
C = 10	96.01%	94.54%	32m 41s

Although computation time increases as C increases, there is no big difference in accuracy values and they are almost the same. However, Sklearn's soft margin linear kernel SVM beat ours in terms of computation time. Ours takes about 1.3 hours, while sklearn's takes a few minutes, except for the scenario where C=10.

c) Polynomial Kernel, Soft Margin SVM

qp(Q, p, G, h, A, c)

$$Q_{i,j} = y_i y_j K(x_i, x_j)$$

- Q is a symmetric NxN matrix.

$$p = (-1 \ldots -1)$$

- p is a vector of −1's of length N, indicating that the dual problem is a maximization problem that negates the original minimization form

constraint $\sum_{n=1}^{N} y_n a_n = 0$

To capture this constraint, we built matrix A and vector c for Ax = c

$$A_{1i} = y_i$$

A encodes the constraint that the weighted sum of dual coefficients by their labels equals zero. The dimensions of A is 1 x N

$$c = [0]$$

constraint $0 \leq alpha \leq C$

To capture this constraint, we built matrix G and vector h for Gx <= h

Negative Identity Part (G and h) lower bound $0 \leq alpha$:

- Matrix G is constructed as a negative identity matrix to transform the $0 \leq alpha$ constraint into a linear inequality form $G * alpha \leq h$
- Vector h for this part is a zero vector of length N, indicating that $-alpha \leq 0$.

Positive Identity Part (G_up and h_up) upper bound $alpha \leq C$:

- Matrix G_up is constructed as a positive identity matrix. This setup means that each $alpha_$i is directly compared to C.
- Vector h_up for this part is a vector filled with the constant C

Then, G and G_up, and h and h_up is combined vertically stacking. Hence, G and h captures both constraints.

Kernel is selected as polynomial 2nd degree. $(1 + X_1X_2)^2$

Subsampling: We tried to optimize in every way using sparse matrices or try to exploit symmetry etc. However, it looked like the solver would take over 10 hours to solve. That's why we decided to do subsampling. We trained using 1/10 of the samples.

Results:

	Training Accuracy	Test Accuracy	Time
C = 0.01	96.97%	97.01%	30s
C = 0.1	97.42%	97.21%	30s
C = 1	97.35%	96.98%	30s
C = 10	97.04%	96.90%	34s

Even though we did subsample, we got better results for each C value from the linear kernel. Nonlinear kernel captured data better.

d) sklearn's Polynomial Kernel, Soft Margin SVM

	Training Accuracy	Test Accuracy	Time
C = 0.01	94.42%	95.02%	5m 6s
C = 0.1	97.82%	97.58%	2m 30s
C = 1	99.63%	99.14%	1m 21s
C = 10	100.00%	99.14%	1m 5s

Since we did subsampling in our own SVM, our training times are shorter. However, we can see better in sklearn's SVM that as the C value increases, tolerance decreases and accuracy increases significantly.

2) Feature extraction

For feature extraction we implemented PCA because it effectively reduces the high dimensionality of the MNIST dataset (784 features per image) while preserving most of the variance. Also, it is computationally more efficient and straightforward.

Extracted top 10 features:

C = 1	Training Accuracy	Test Accuracy	Time
Part a	89.16%	89.43%	16m 42s
Part b	88.97%	89.36%	1m 3s

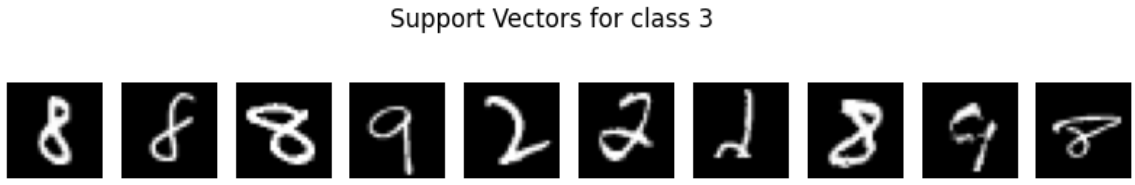
Part c	93.11%	92.98%	59s
Part d	93.64%	93.28%	25s

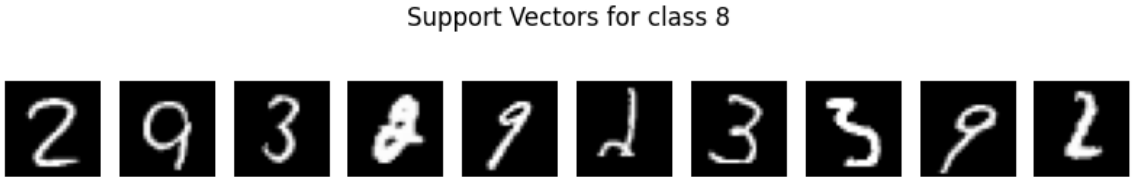
Impact on Accuracies:

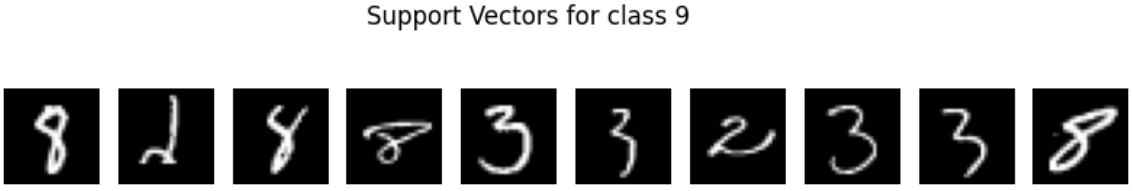
Training Accuracy: The accuracy during the training phase dropped slightly because the model had less detailed information to learn from. However, it still retained most of the important patterns and structures in the data.

Test Accuracy: Similarly, the test accuracy also decreased. This is expected because with fewer features, the model might lose some important details that could help in making more accurate predictions.

3) Support Vectors







- Support vectors are more ambiguous and less distinct compared to other images in the dataset. They are very close to the decision boundary and the margin of the SVM.

- The support vectors for each class sometimes appear like digits of other classes, highlighting their ambiguous nature. For instance, some support vectors for class 3 look like 8 for many instances.

Clustering

1) Normalizing

Yes, normalizing data points before running K-means is important. Here's why:

1. **Equal Importance**: Different features can have different scales. For example, one feature might range from 1 to 10, while another ranges from 1 to 1000. Without normalization, K-means can give more weight to features with larger ranges, even if they are not more important.
2. **Distance Calculation**: K-means relies on distance measurements (like Euclidean distance) to assign points to clusters. If features have different scales, the distance calculations can be skewed, leading to incorrect clustering results.
3. **Better Performance**: Normalized data helps K-means converge faster and find better clusters. It ensures that each feature contributes equally to the distance calculation.

Example

Imagine you have two features:

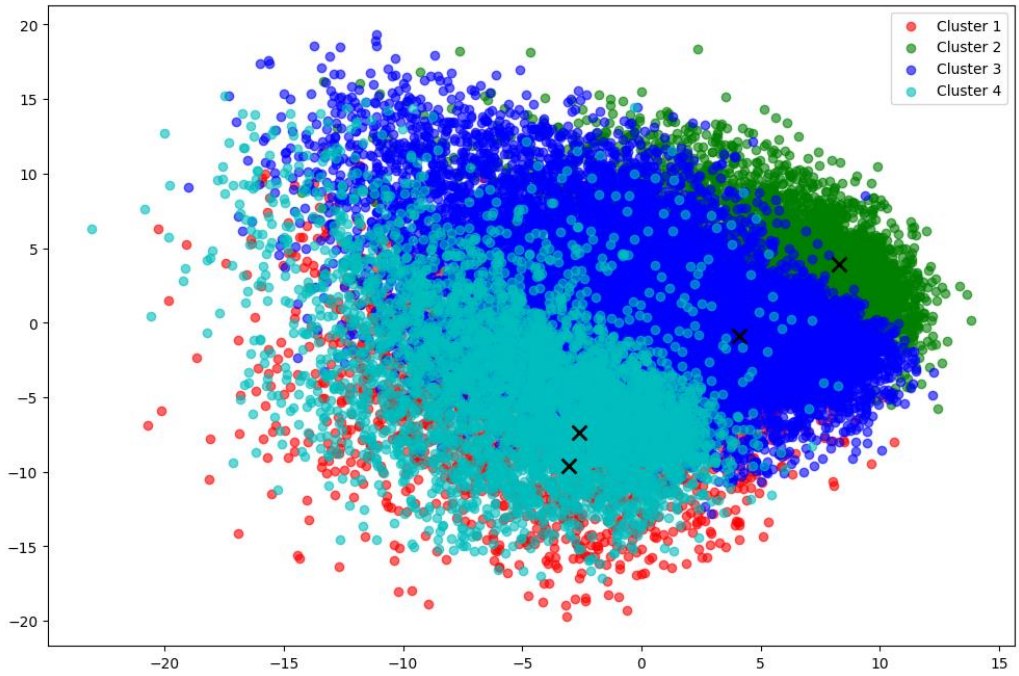
- **Feature A**: Age (1 to 100)
- **Feature B**: Income (1000 to 100000)

Without normalization, the income feature will dominate the distance calculation because its values are much larger. This can cause the algorithm to group data points mainly based on income, ignoring the age feature.

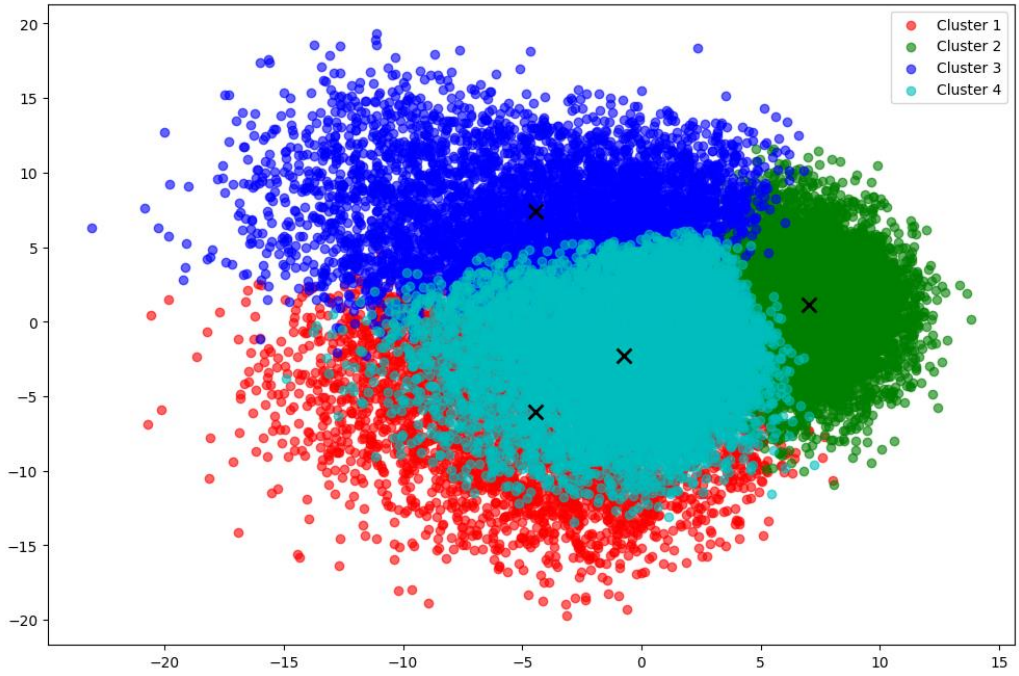
2) Eucledian Distance

Implementing the KMeans algorithm to the extracted featured flattened images with eucledian distance, the random initialized centroids' translation to the 4 clusters can be seen in the plots:

Initial Scattered Dataset:



Final Scattered Dataset:



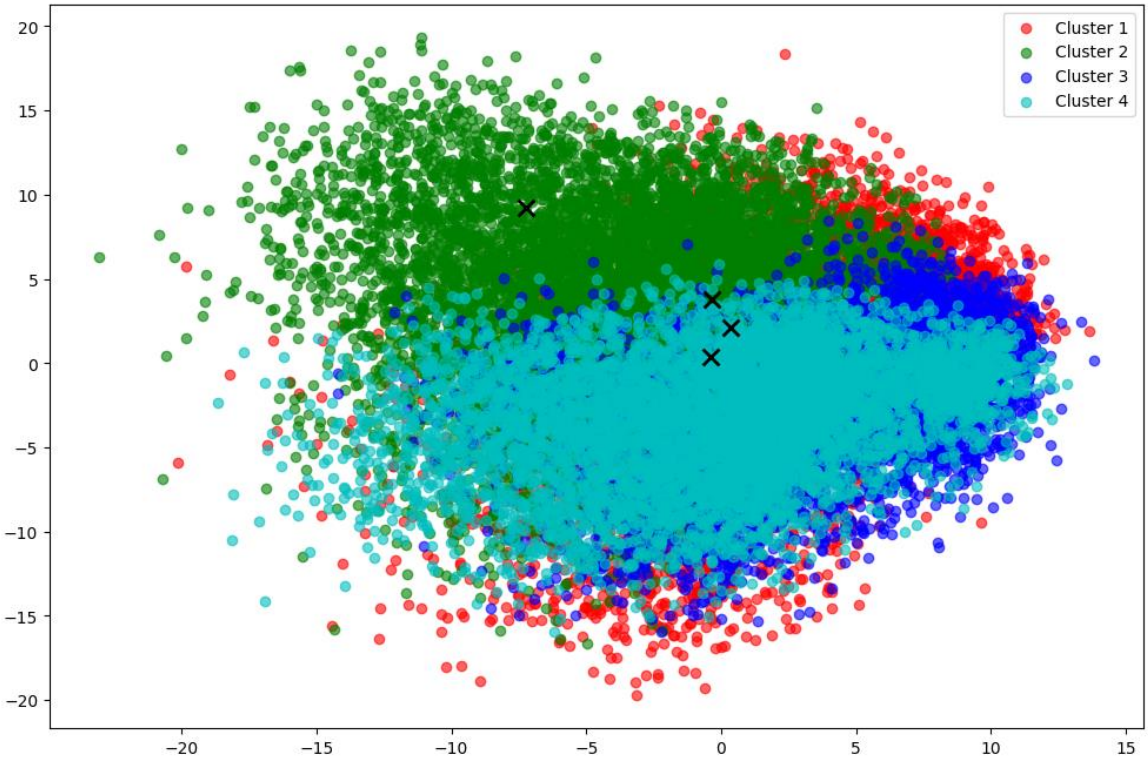
The comparison between this and including extracted features can be shown by clustering accuracy and SSE. By running the algorithm with both train images and computing their metrics, we can observe:

a) Extracting features lowered SSE largely, which means a better fit.
b) Extracting features did not cause much in the case of accuracy. It may depend on data characteristics and KMeans algorithm's limitations.

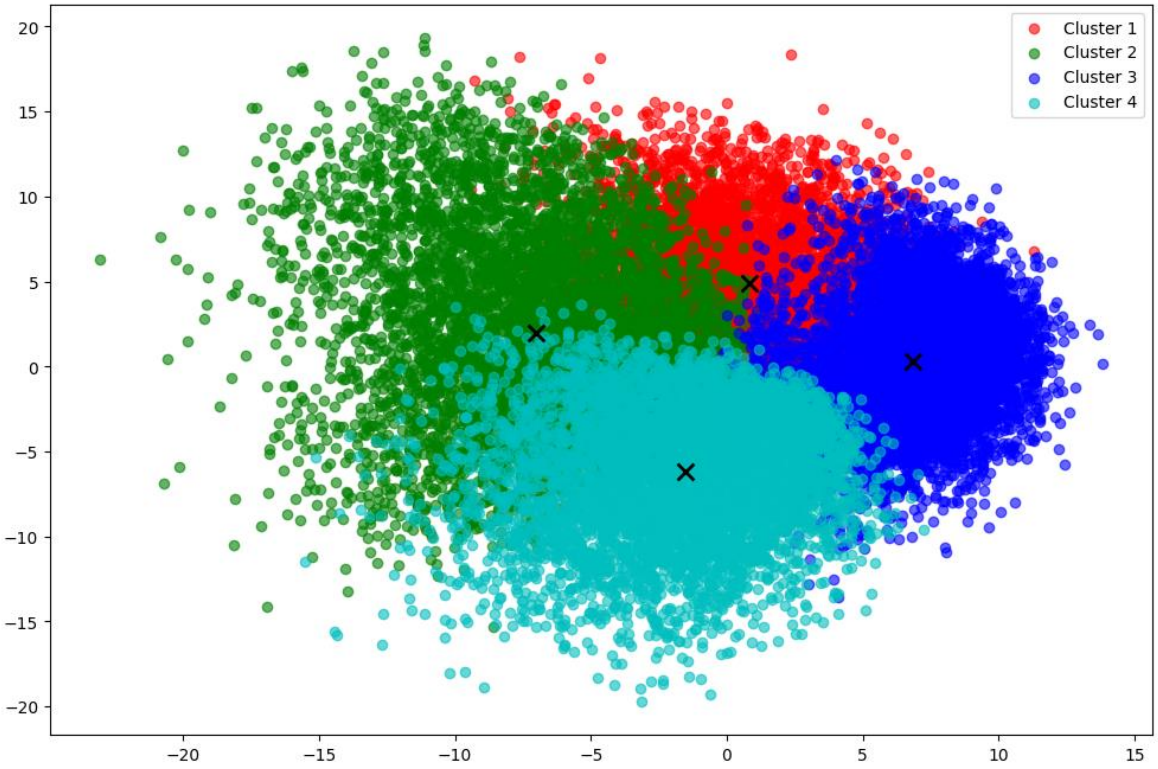
3) Cosine Similarity

Implementing the KMeans algorithm to the extracted featured flattened images with cosine similarity distance, the random initialized centroids' translation to the 4 clusters can be seen in the plots:

Initial Scattered Dataset:



Final Scattered Dataset:



The difference between the plots of eucledian distance KMeans and cosine similarity KMeans is because of the random selection of initial centroids. The fact that there are no significant difference can be shown by comparing the metrics of each approach. The cluster accuracy and SSE values are very similar to each other even with the behaviour when we included the extracted features.