CIS 419/519: Applied Machine Learning — Fall 2024

Homework 3

Handed Out: October 23 — *Due: 7:59 pm November 6*

Name: Alan Wu

PennKey: alanlwu

PennID: 41855518

1 Declaration

- **Person(s) discussed with:** *Your answer*
- **Affiliation to the course: student, TA, prof etc.** *Your answer*
- **Which question(s) in coding / written HW did you discuss?** ***Your answer***
- **Briefly explain what was discussed.** *Your answer*

2 Multiple Choice & Written Questions

1. (a) To draw the two principal components of both graphs, we have used black to represent the first principal component and red to represent the second principal component:

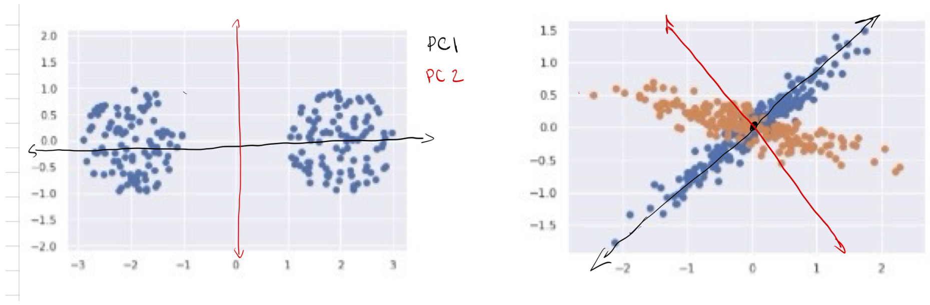

Figure 1: PCA Components

(b) If we want to retain 75% of the variance, we would need to get the number of principal components that capture greater than 75% of the sum of the covariance matrix eigenvalues.

The sum of the eigenvalues of the covariance matrix represent the total variance of the data and thus that computation is $2.1 + 1.8 + 1.3 + 0.9 + 0.4 + 0.2 + 0.15 +$

$0.02 + 0.001 = 6.871$. To retain 75% of the variance, we would need to get the number of principal components that capture greater than 75% of the sum of the covariance matrix eigenvalues. This would be $6.871 * 0.75 = 5.153$.

The largest amount of variance will be captured by the first components and the least number of components to get a sum greater than 5.153 is 3 components. $2.1 + 1.8 + 1.3 = 5.2$

Therefore $K = 3$.

(c) **A**

2. (a) To compute two iterations of the k-means algorithm we start with the following samples:

A = (2,3), B = (4,6), C = (5,1), and D = (10,12)
Our first initialized centroids are (6,9) for cluster 1 and (8,4) for cluster 2.

Iteration 1:

- A
  - $d(A,1) = \sqrt((2-6)^2 + (3-9)^2) = \sqrt(52) = 7.211$
  - $d(A,2) = \sqrt((2-8)^2 + (3-4)^2) = \sqrt(37) = 6.083$
- B
  - $d(B,1) = \sqrt((4-6)^2 + (6-9)^2) = \sqrt(13) = 3.606$
  - $d(B,2) = \sqrt((4-8)^2 + (6-4)^2) = \sqrt(20) = 4.472$
- C
  - $d(C,1) = \sqrt((5-6)^2 + (1-9)^2) = \sqrt(65) = 8.062$
  - $d(C,2) = \sqrt((5-8)^2 + (1-4)^2) = \sqrt(18) = 4.243$
- D
  - $d(D,1) = \sqrt((10-6)^2 + (12-9)^2) = \sqrt(25) = 5$
  - $d(D,2) = \sqrt((10-8)^2 + (12-4)^2) = \sqrt(80) = 8.246$
- Cluster 1 members: B, D
- Cluster 1 updated centroid = (7,9)
  updated centroid = midpoint(B,D) = ((4+10)/2, (6+12)/2) = (7,9)
- Cluster 2 members: A, C
  updated centroid = midpoint(A,C) = ((2+5)/2, (3+1)/2) = (3.5,2)

Iteration 2:

- A
  - $d(A,1) = \sqrt((2-7)^2 + (3-9)^2) = \sqrt(61) = 7.810$

  - $d(A,2) = \sqrt((2-3.5)^2 + (3-2)^2) = \sqrt(3.25) = 1.803$
- B
  - $d(B,1) = \sqrt((4-7)^2 + (6-9)^2) = \sqrt(18) = 4.243$
  - $d(B,2) = \sqrt((4-3.5)^2 + (6-2)^2) = \sqrt(16.25) = 4.031$
- C
  - $d(C,1) = \sqrt((5-7)^2 + (1-9)^2) = \sqrt(68) = 8.246$
  - $d(C,2) = \sqrt((5-3.5)^2 + (1-2)^2) = \sqrt(3.25) = 1.803$
- D
  - $d(D,1) = \sqrt((10-7)^2 + (12-9)^2) = \sqrt(18) = 4.243$
  - $d(D,2) = \sqrt((10-3.5)^2 + (12-2)^2) = \sqrt(142.25) = 11.927$
- Cluster 1 members: D
- Cluster 1 updated centroid = (10,12)
  updated centroid = midpoint(D) = (10,12)
- Cluster 2 members: A, B, C
  updated centroid = midpoint(A,B,C) = ((2+4+5)/3, (3+6+1)/3) = (3.67,3.33)

3. (a) **B**

   Since the output of the first filter will just simply be the sum of all the pixels weighted by 1/9, we can expect that the filter's behavior is that of an averaging over the 8 neighboring pixels and the pixel itself. This will result in a blurring effect because sharper parts of the image will be averaged out with the surrounding pixels. The image generally looks 'smoother' than it did before the filter was applied.

   (b) **A**

   The kernel itself is all zeroes except for the center, which is 2. This means that for every kernel application, the output will double the pixel value of the center pixel. This generally means that when the filter is applied across the image, the image will be brighter across the entire image.

   (c) **C**

   When applying the kernel we noticed that it is darkening the image and also capturing a vertical line pattern.

4. (a) **27** output size dimension 1

   To solve this problem, we will need two equations that help us reduce and count the dimension reduction/shift between convolutional layers.

   Given a convolutinal layer with kernel size kxk, stride s, and padding p, the output size O of the output feature map for a given input dimension D is going to be $O = \frac{D-k+2\cdot p}{s} + 1$.
   Furthermore, given a max pooling layer with kernel size kxk, stride s, and padding p, the output size O of the output feature map for a given input dimension D is going to be $O = \frac{D-k+2\cdot p}{s} + 1$.

For a third feature, the number of out channels will be the dimension output of the previous layer.

With these, we can compute the output size of the convolutional layers and the max pooling layers.
Our network has 4 2d convolution layers and 3 max pool layers.
We can progress through the network and compute the output size of each layer. We are given input size (232, 232, 3), dimensions 1, 2, 3 respectively

- Convolution layer 1: (In channels 3, Out channels 5, kernel: 5x5 Stride 1, Padding 0)
  - $O_{\text{dimension}_1} = \frac{232-5+2\cdot 0}{1} + 1 = 228$
  - $O_{\text{dimension}_2} = \frac{232-5+2\cdot 0}{1} + 1 = 228$
  - $O_{\text{dimension}_3} = 5$
- Max Pooling layer 1: (kernel: 2x2, Stride 2, Padding 0)
  - $O_{\text{dimension}_1} = \frac{228-2+2\cdot 0}{2} + 1 = 114$
  - $O_{\text{dimension}_2} = \frac{228-2+2\cdot 0}{2} + 1 = 114$
  - $O_{\text{dimension}_3} = 5$
- Convolution layer 2: (In channels 5, Out channels 10, kernel: 3x3, Stride 1, Padding 0)
  - $O_{\text{dimension}_1} = \frac{114-3+2\cdot 0}{1} + 1 = 112$
  - $O_{\text{dimension}_2} = \frac{114-3+2\cdot 0}{1} + 1 = 112$
  - $O_{\text{dimension}_3} = 10$
- Max Pooling layer 2: (kernel: 2x2, Stride 2, Padding 0)
  - $O_{\text{dimension}_1} = \frac{112-2+2\cdot 0}{2} + 1 = 56$
  - $O_{\text{dimension}_2} = \frac{112-2+2\cdot 0}{2} + 1 = 56$
  - $O_{\text{dimension}_3} = 10$
- Convolution layer 3: (In channels 10, Out channels 20, kernel: 3x3, Stride 1, Padding 0)
  - $O_{\text{dimension}_1} = \frac{56-3+2\cdot 0}{1} + 1 = 54$
  - $O_{\text{dimension}_2} = \frac{56-3+2\cdot 0}{1} + 1 = 54$
  - $O_{\text{dimension}_3} = 20$
- Max Pooling layer 3: (kernel: 2x2, Stride 2, Padding 0)
  - $O_{\text{dimension}_1} = \frac{54-2+2\cdot 0}{2} + 1 = 27$
  - $O_{\text{dimension}_2} = \frac{54-2+2\cdot 0}{2} + 1 = 27$
  - $O_{\text{dimension}_3} = 20$

We come to the conclusion of the output size of the network: (27, 27, 20).

(b) **27** output size dimension 2

(c) **20** output size dimension 3

(d) **2660** total number of parameters

Given the C_{out}, or number of output channels, number of input channels C_{in}, the kernel size kxk, we know the number of parameters per convolutional layer is going to be $(C_{\text{out}} \cdot C_{\text{in}} \cdot k^2) + C_{\text{out}}$.

We can compute the number of parameters for each layer and then sum them up

- Convolutional layer 1: $(5 \cdot 3 \cdot 5^2) + 5 = 380$
- Convolutional layer 2: $(10 \cdot 5 \cdot 3^2) + 10 = 460$
- Convolutional layer 3: $(20 \cdot 10 \cdot 3^2) + 20 = 1820$

Total parameters = 380 + 460 + 1820 = 2660

3 Python Programming Questions

1. **Question 1**

(a) **1.4**

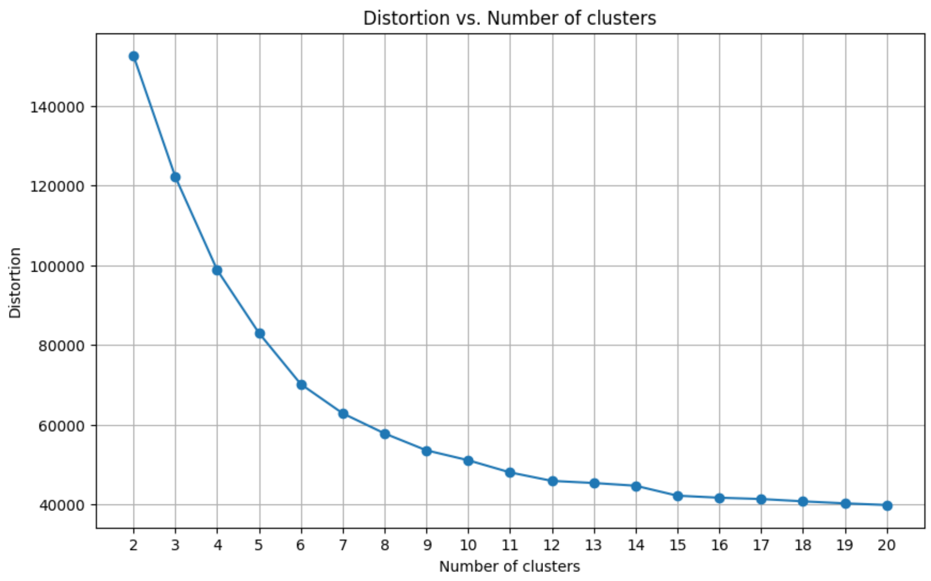

Figure 2: K Means Clustering Elbow Method No Scaling

We can see the optimal k is somewhere between 7 and 8. Let us say k = 7.

(b) **1.5**

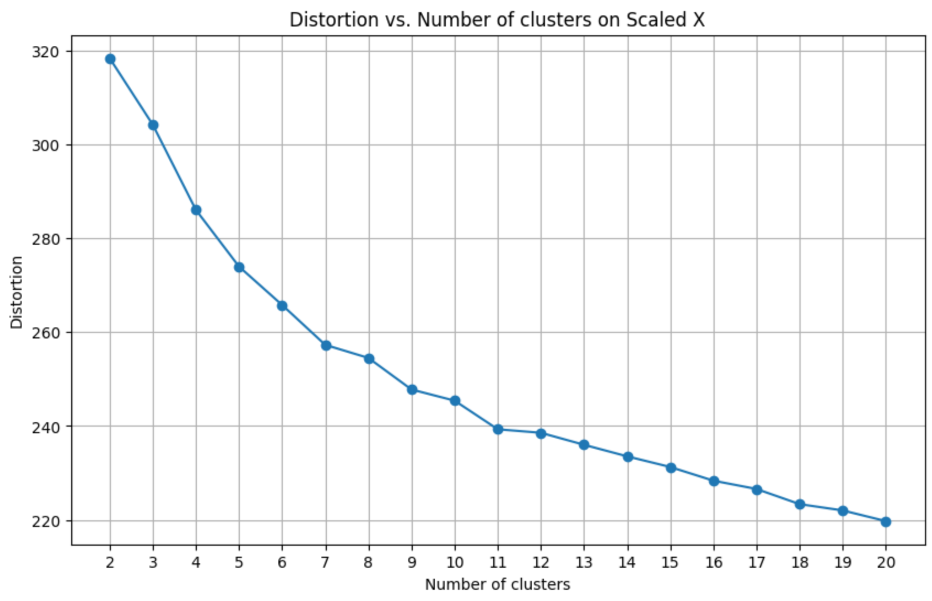

Figure 3: K Means Clustering Elbow Method With Scaling

We can see the optimal k is somewhere between 10 and 11. Let us say k = 10.

(c) **1.6**

1. Why do you get different results with and without feature scaling?

With feature scaling, the data is normalized to have a mean of 0 and a standard deviation of 1. This means that all the features are on the same numerical scale. This is important because now we are able to capture the distances between points on a uniform scale. Since we are more easily able to determine the clusters with the scaled data, we can see that the elbow method yields k = 10 or k = 11 as the optimal k. However, with the unscaled features, we notice that the elbow method yields an optimal k of k = 7 or k = 8. With the scaled features, we can see that we obtain a greater estimate for optimal k. The optimal k shifts because the focus of the features is distributed equally across all features.

2. Should you scale the features before fitting k-means? Why or why not?

Scaling the features is beneficial before fitting k-means clustering because we can capture the contributions of all features equally on the same scale. Thus, larger scale features will not overtake the algorithm, as we will have more accurate attribution and distance metric calculations.

2. **Question 2**

(a) **2.1**

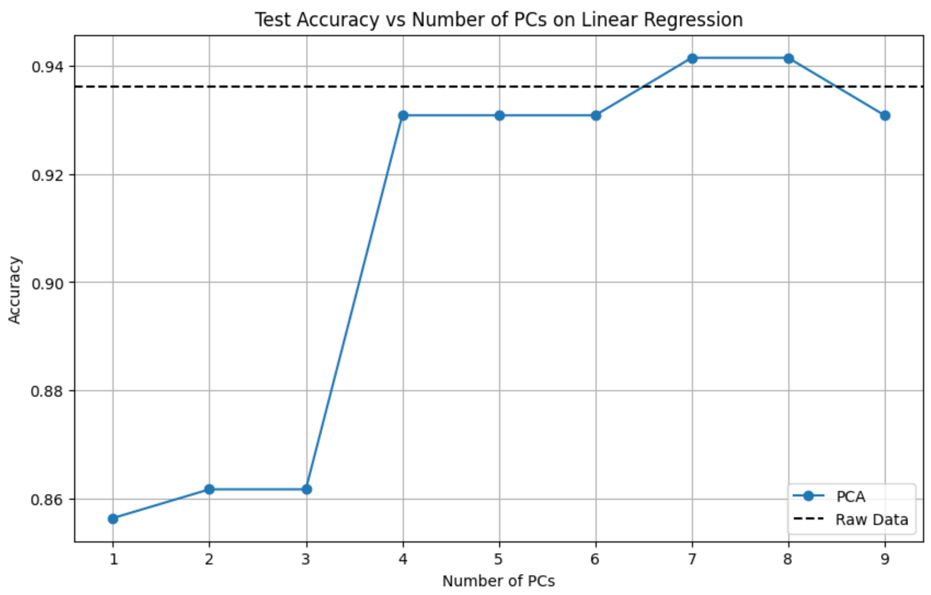

Figure 4: PCA Accuracy

As expected, as we begin to add principal components to the model, the test accuracy of the model increases. However, after peaking at the 7th principal component, the test accuracy begins to level out and decreases slightly. This trend that we are noticing can be attributed to the fact that the model is overfitting to the training data. The model is capturing the noise in the data after the 7th principal component and thus the test accuracy begins to decrease. We also notice that adding principal components is not necessarily outperforming the original model without PCA. This could be due to the fact that the model itself already is attributing large weights to important features.

(b) **2.2.2** The features that are explained best by the PCs together are 'Worst Area', 'Mean Area', and 'Area error'

(c) **2.3**

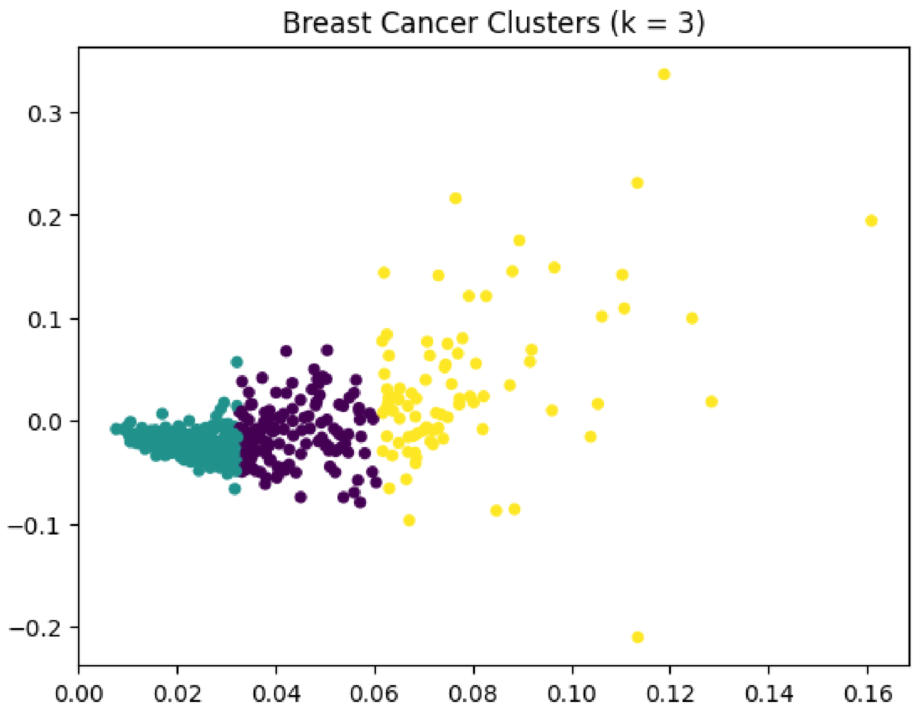

Figure 5: Breast Cancer Clusters K = 3

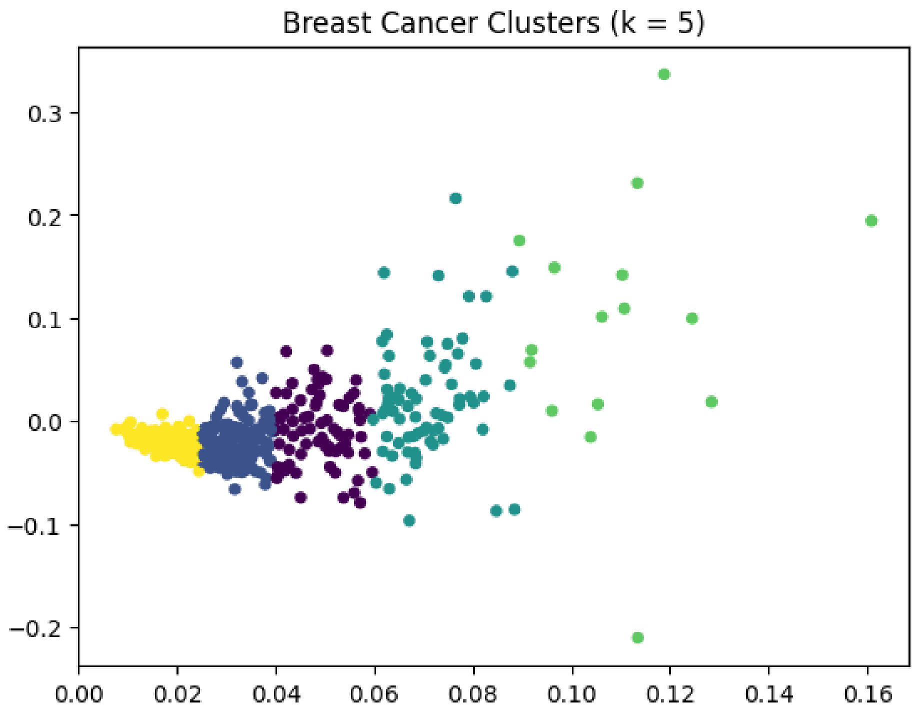

Figure 6: Breast Cancer Clusters K = 5

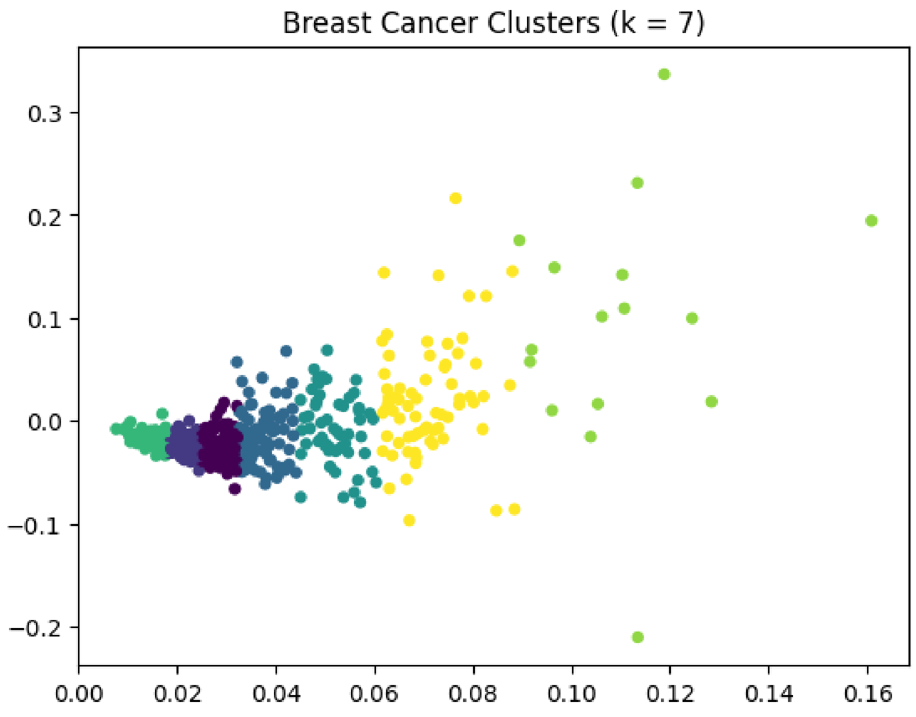

Figure 7: Breast Cancer Clusters K = 7

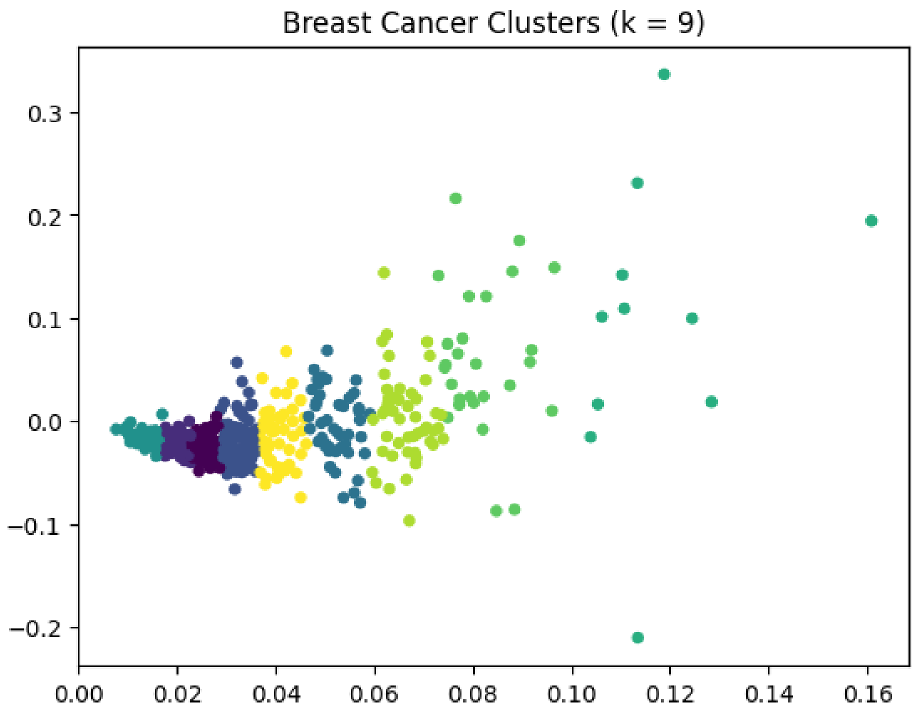

Figure 8: Breast Cancer Clusters K = 9

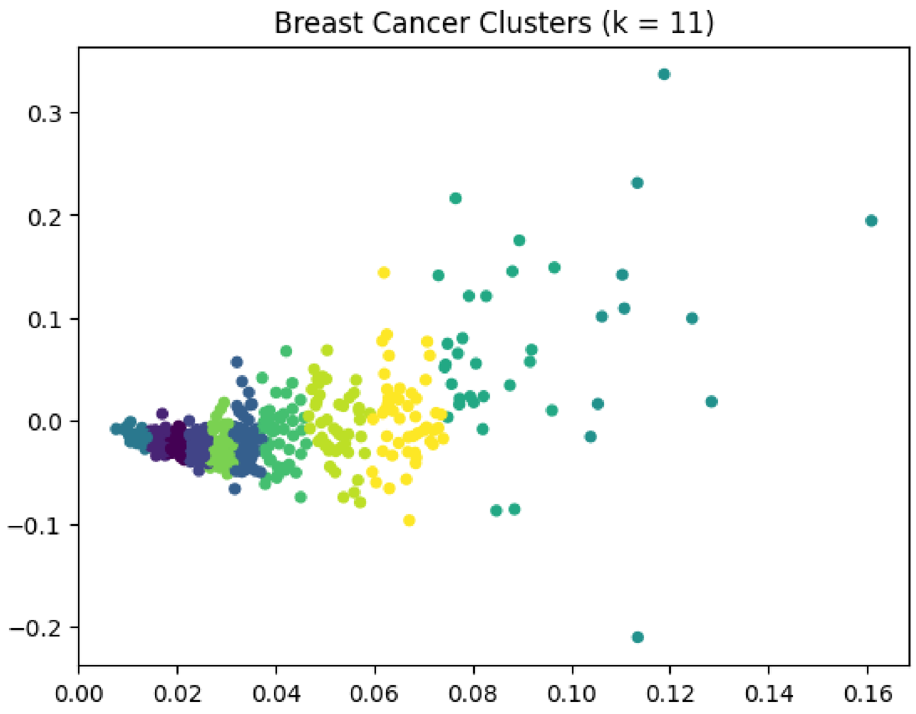

Figure 9: Breast Cancer Clusters K = 11