CIS 419/519: Applied Machine Learning — Fall 2024

Homework 3

Handed Out: — *Due: 7:59 pm Nov 6*

Name: Richard Zhang

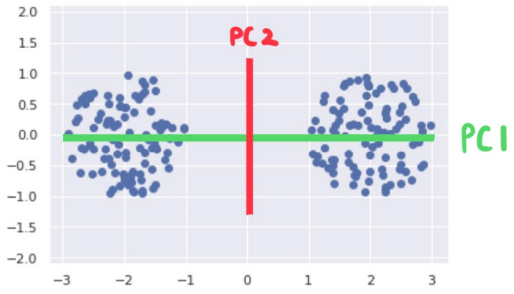
PennKey: zhank24

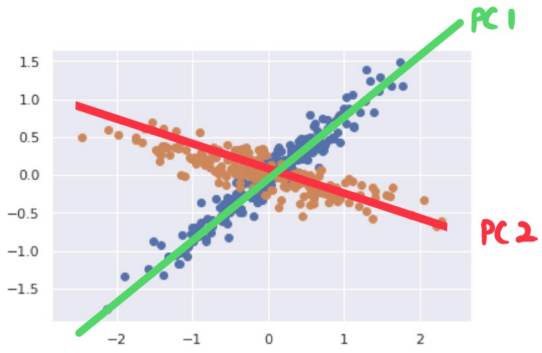
PennID: 19331985

1 Declaration

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

2 Question 1

1. 



2. The total variance is the sum of all eigenvalues:

$$\text{Total Variance} = 2.1 + 1.8 + 1.3 + 0.9 + 0.4 + 0.2 + 0.15 + 0.02 + 0.001 = 6.871$$

$$0.75 \cdot 6.871 = 5.15325$$

We need to add the highest eigenvalues until the sum is at least 5.15325:

1st eigenvalue: 2.1
plus 2nd eigenvalue: $1.8 + 2.1 = 3.9$
plus 3rd eigenvalue: $1.3 + 3.9 = 5.2 > 5.15325$

Thus, the first three PCs can achieve a variance of 5.2, which is more than 75% of the total variance. So, K is 3.

3. (a) a point with large feature values in Z_1 and small feature values in Z_2

3 Question 2

1. First Iteration

   We first calculate the distance from every point to both centroids:

$$d(A,1) = \sqrt{(2-6)^2 + (3-9)^2} = \sqrt{52} \approx 7.211$$

$$d(A,2) = \sqrt{(2-8)^2 + (3-4)^2} = \sqrt{37} \approx 6.082$$

$$d(B,1) = \sqrt{(4-6)^2 + (6-9)^2} = \sqrt{13} \approx 3.61$$

$$d(B,2) = \sqrt{(4-8)^2 + (6-4)^2} = \sqrt{20} \approx 4.47$$

$$d(C,1) = \sqrt{(5-6)^2 + (1-9)^2} = \sqrt{65} \approx 8.06$$

$$d(C,2) = \sqrt{(5-8)^2 + (1-4)^2} = \sqrt{18} \approx 4.24$$

$$d(D,1) = \sqrt{(10-6)^2 + (12-9)^2} = \sqrt{25} = 5$$

$$d(D,2) = \sqrt{(10-8)^2 + (12-4)^2} = \sqrt{68} \approx 8.25$$

   Now, we assign points to their nearest centroids:

   Cluster 1 members: B, D

   Cluster 2 members: A, C

   Then, we updater centroids:

   New centroid 1:

$$(\frac{4+10}{2}, \frac{6+12}{2}) = (7, 9)$$

   New centroid 2:

$$(\frac{2+5}{2}, \frac{3+1}{2}) = (3.5, 2)$$

2. Second Iteration

   We first calculate the distance from every point to both centroids:

$$d(A,1) = \sqrt{(2-7)^2 + (3-9)^2} = \sqrt{61} \approx 7.81$$

$$d(A,2) = \sqrt{(2-3.5)^2 + (3-2)^2} = \sqrt{3.25} \approx 1.80$$

$$d(B,1) = \sqrt{(4-7)^2 + (6-9)^2} = \sqrt{18} \approx 4.24$$
$$d(B,2) = \sqrt{(4-3.5)^2 + (6-2)^2} = \sqrt{16.25} \approx 4.03$$

$$d(C,1) = \sqrt{(5-7)^2 + (1-9)^2} = \sqrt{68} \approx 8.25$$
$$d(C,2) = \sqrt{(5-3.5)^2 + (1-2)^2} = \sqrt{3.25} \approx 1.80$$

$$d(D,1) = \sqrt{(10-7)^2 + (12-9)^2} = \sqrt{18} \approx 4.24$$
$$d(D,2) = \sqrt{(10-3.5)^2 + (12-2)^2} = \sqrt{142.25} \approx 11.93$$

Now, we assign points to their nearest centroids:

Cluster 1 members: B

Cluster 2 members: A, B, C

Then, we updater centroids:

New centroid 1:

$$((10, 12)$$

New centroid 2:

$$(\frac{2+4+5}{3}, \frac{3+6+1}{3}) = (3.67, 3.33)$$

Report the results:

First Iteration:

- $A : d(A,1) = 7.211, \quad d(A,2) = 6.082$
- $B : d(B,1) = 3.61, \quad d(B,2) = 4.47$
- $C : d(C,1) = 8.06, \quad d(C,2) = 4.24$
- $D : d(D,1) = 5, \quad d(D,2) = 8.25$
- cluster 1 members: B, D
- cluster 1 updated centroid: $(7, 9)$
- cluster 2 members: A, C
- cluster 2 updated centroid: $(3.5, 2)$

Second Iteration

- $A : d(A,1) = 7.81, \quad d(A,2) = 1.80$
- $B : d(B,1) = 4.24, \quad d(B,2) = 4.03$

- $C : d(C, 1) = 8.25, \quad d(C, 2) = 1.80$
- $D : d(D, 1) = 4.24, \quad d(D, 2) = 11.93$
- cluster 1 members: D
- cluster 1 updated centroid: $(10, 12)$
- cluster 2 members: A, B, C
- cluster 2 updated centroid: $(3.67, 3.33)$

4 Question 3

1. **B**. This is an averaging filter, which can blur an image when applied. To be more specific, it would average all the pixels in this $3 * 3$ matrix and reduce details in the image. B is the image blurred compared to the original. It also loses many details and become smoother.

2. **A**. This is just a scaling filter that make the center pixel's value 2 times bigger than its original value. As a result, the contrast of an image would increase, making it brighter (having higher intensity). A is the only image meeting this description.

3. **C**. This filter is a filter used to detect vertical edges because we can observe the different values of adjacent columns. The left and right columns are positive and negative, respectively, creating a huge difference on the image if there is a vertical intensity change. C is the image meeting the above description. A is more like a horizontal edge filter.

5 Question 4

1. We know:
$$\text{output dimension} = \frac{\text{input dimension} - k + 2p}{s} + 1$$
where p is the padding, k is the kernel size, and s is the stride.

2. Step (a) — Conv2d:

   We know there are 5 out channels, and:
$$\frac{232 - 5}{1} + 1 = 228$$
   Thus, the output dimension is $(228, 228, 5)$.

3. Step (b) — ReLu:

   No change.

4. Step (c) — MaxPool2d:

   We know stride is 2 and the kernel size is 2, so:

   $$\frac{228-2}{2}+1=114$$

   Thus, the output dimension is $(114, 114, 5)$.

5. Step (d) — Conv2d:

   We know there are 10 out channels, and:

   $$\frac{114-3}{1}+1=112$$

   Thus, the output dimension is $(112, 112, 10)$.

6. Step (e) — ReLu:

   No change.

7. Step (f) — MaxPool2d:

   We know stride is 2 and the kernel size is 2, so:

   $$\frac{112-2}{2}+1=56$$

   Thus, the output dimension is $(56, 56, 10)$.

8. Step (g) — Conv2d:

   We know there are 20 out channels, and:

   $$\frac{56-3}{1}+1=54$$

   Thus, the output dimension is $(54, 54, 20)$.

9. Step (h) — ReLu:

   No change.

10. Step (f) — MaxPool2d:

    We know stride is 2 and the kernel size is 2, so:

    $$\frac{54-2}{2}+1=27$$

    Thus, the output dimension is $(27, 27, 20)$.

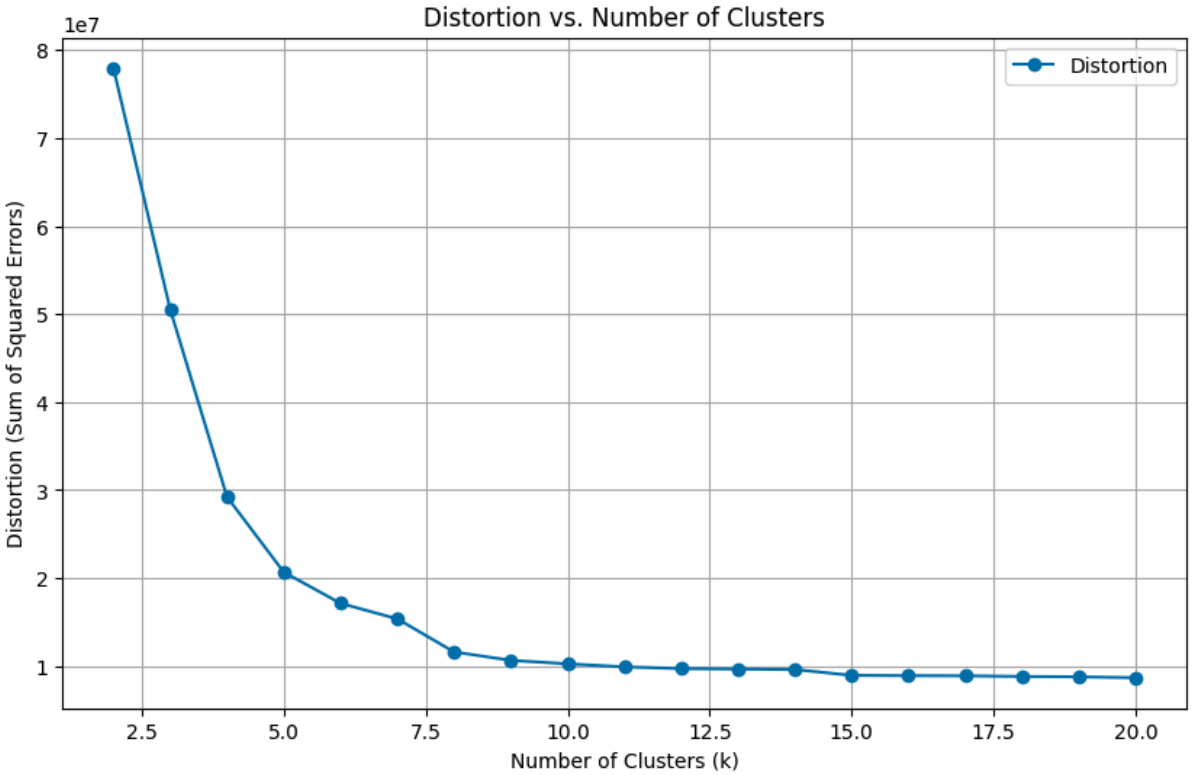
11. The number of learnable parameters is the total number of parameters in every Conv2d layer, and the equation is:

    $$\text{number of parameters} = (\text{kernel height} \times \text{kernel width} \times \text{in channels} + 1) \times \text{out channels}$$

    $$(5\times5\times3+1)\times5+(3\times3\times5+1)\times10+(3\times3\times10+1)\times20=380+460+1820=2660$$

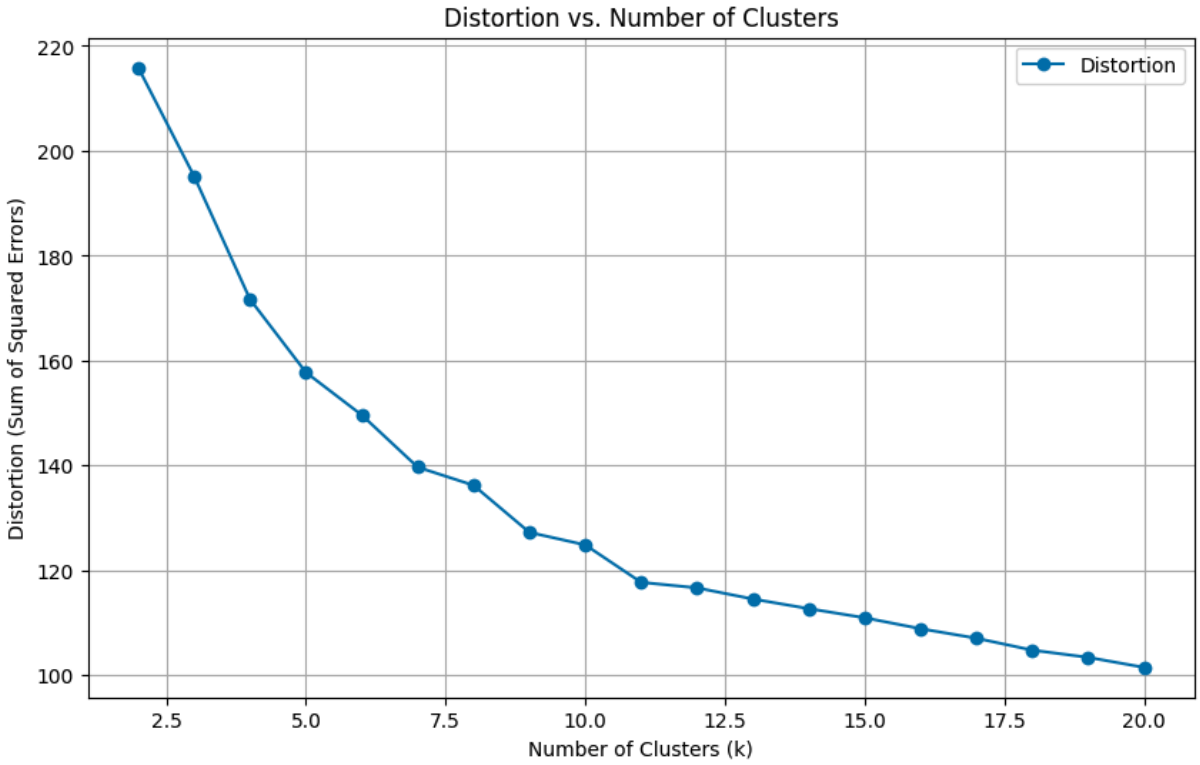
    Therefore, the final output size is $(27, 27, 20)$**, and the number of learnable parameters is** 2660**.**

6 Python Programming Question 1.4

1. 

2. We need to determine the elbow point, which is the point where the decreasing rate of distortion begins to slow down, to find the best value for k. $\mathbf{k = 5}$ is the elbow point in this plot, so the best value of k is 5.

7 Python Programming Question 1.5

1. 

2. We can observe that when $\mathbf{k = 7}$, the decreasing rate of distortion begins to slow down, so it should be the best value that can balance the decreases of distortion and the number of clusters.

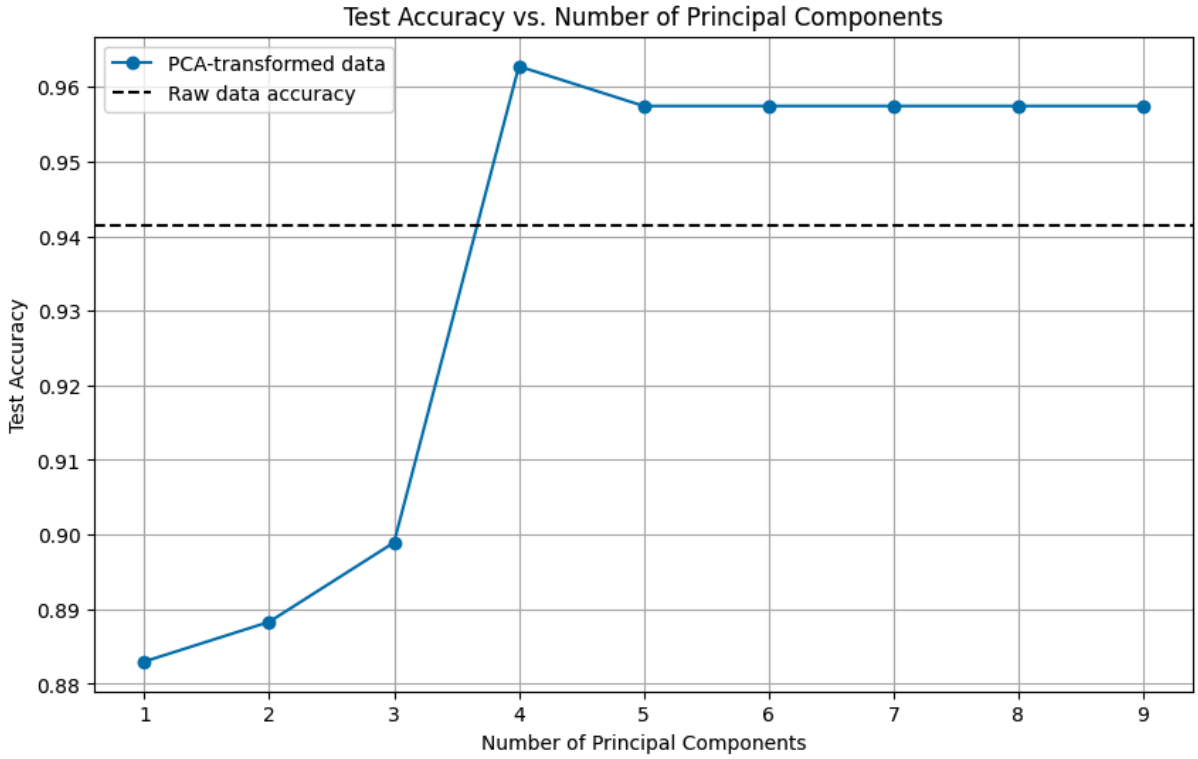
8 Python Programming Question 1.6

1. The difference is because we use Euclidean distance to form clusters, and this distance is sensitive to the scale of the data. When data has different scales, larger-scale data would heavily impact the distance calculations, which lead to biased clusters. Without scaling, the clusters we form could be skewed along the axes of features with larger ranges and harm the model's accuracy. If we standardize all the features, we make sure every variable would contribute equally. This is the reason why we get different results with and without feature scaling.

2. Yes, I think it's very helpful to scale features before fitting K-means, especially if we have features with very different ranges or scales. Scaling each feature to a standard range make our results (aka resulting clusters) more interpretable and let each variable contributes equally to the calculation of Euclidean distance. This decreases bias from variables with larger scales, making sure that K-means can discover clusters based on the overall structure of our data. Finally, it also helps the k-means converge more quickly and effectively.

9 Python Programming Question 2.1

1. {0.9414893617021277}



2. From what I see, adding more principal components initially boosts test accuracy, with a sharp improvement up to around 3 components, where it even exceeds the accuracy achieved with the raw data ($n = 4$). After this point, adding extra components doesn't help much, and the accuracy begins to go down (from $n = 4$ to $n = 5$). After $n = 5$, the accuracy remains stable at a high level. **Based on this plot, the optimal number of principal components is 4**. At this point, we can make our computation fast and efficient, save resources and time, and avoid the over-fitting problem.

   This pattern can be explained by the fact that in PCA, the first few components capture most of the variance in the data, letting the model perform well with fewer dimensions. This explains the rise in accuracy from 1 to 4 components. After that, the additional components begins to capture noises, which doesn't contribute to the model's accuracy. This can lead to a decrease in accuracy, as seen with 5 components. In conclusion, PCA is great tool that can balance accuracy and the number of dimensionality, and by using it, we can have similar, or even better, performance than with all of the feaatures.

10 Python Programming Question 2.2.2

1. We know $n = 4$ is the best value from the previous part. We run the code and get the following output:

2. top 3 features that contribute most to PC1: [(0.8520633917981462, 'worst area'), (0.516826468722458, 'mean area'), (0.055727166911070415, 'area error')]

   top 3 features that contribute most to PC2: [(0.8518237204834198, 'mean area'), (0.06274808274892897, 'mean perimeter'), (0.009287056497235222, 'mean radius')]

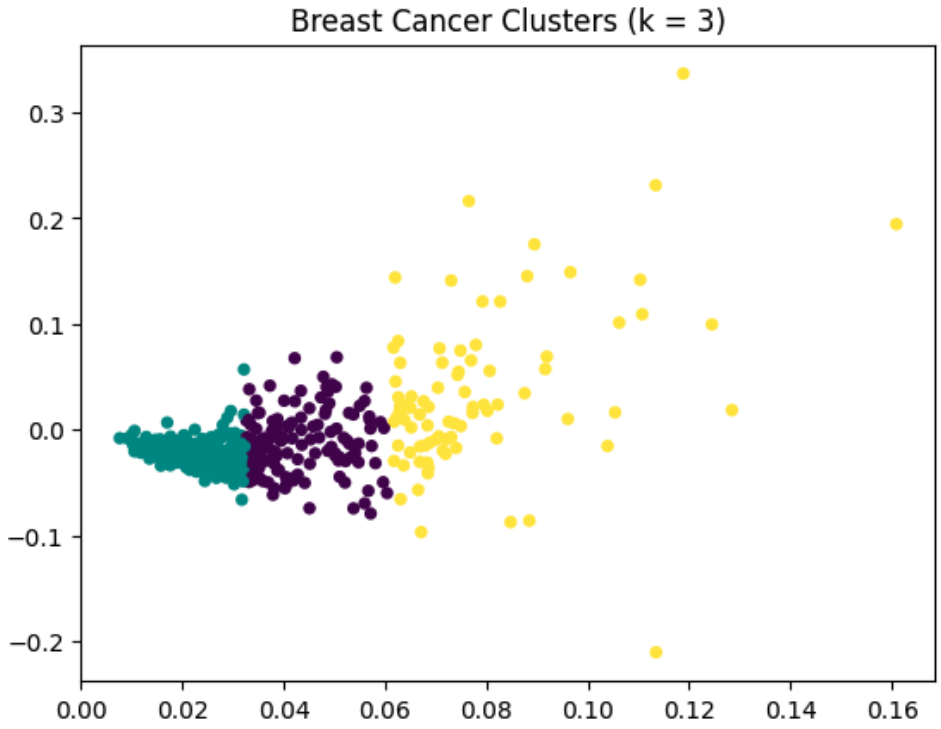
   top 3 features that contribute most to PC3: [(0.9902458782833071, 'area error'), (0.04385603691150584, 'perimeter error'), (0.006233776347976491, 'texture error')]

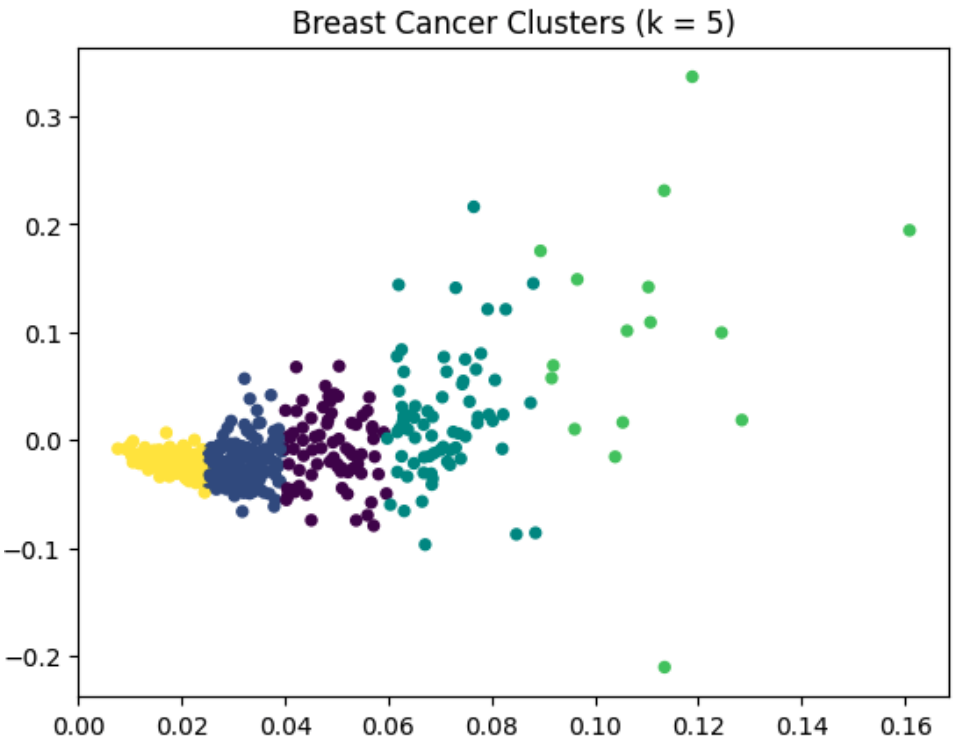
   top 3 features that contribute most to PC4: [(0.6668164509720133, 'worst perimeter'), (0.5420574121748627, 'worst texture'), (0.36241511065885623, 'mean texture')]

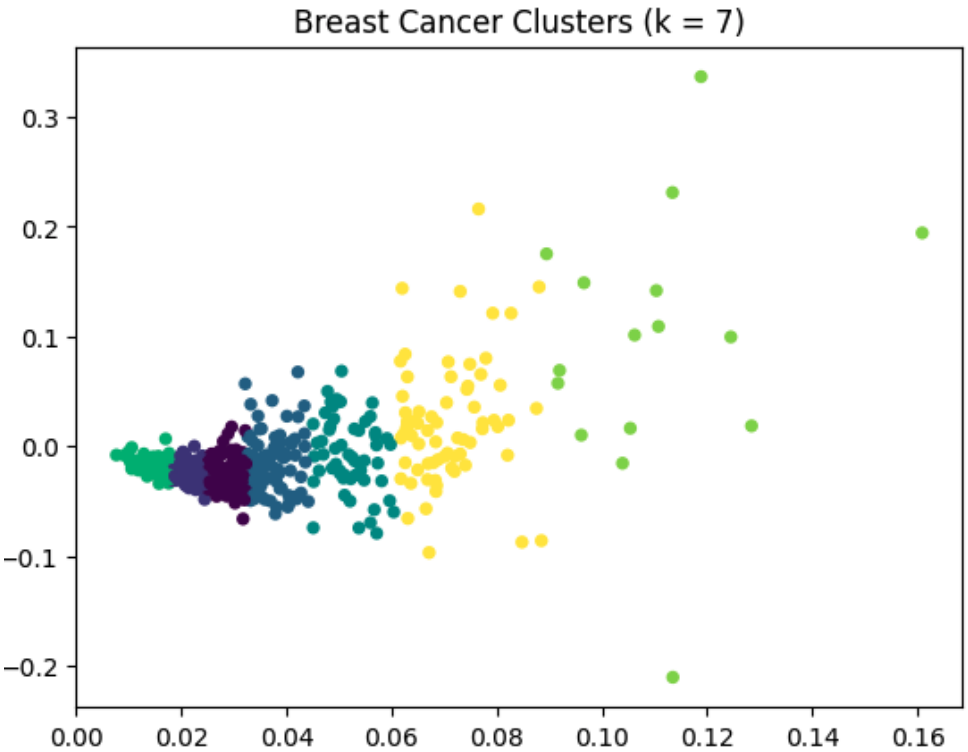
   top 3 features explained by these PCs together: [(0.9902458782833071, 'area error'), (0.8520633917981462, 'worst area'), (0.8518237204834198, 'mean area')]

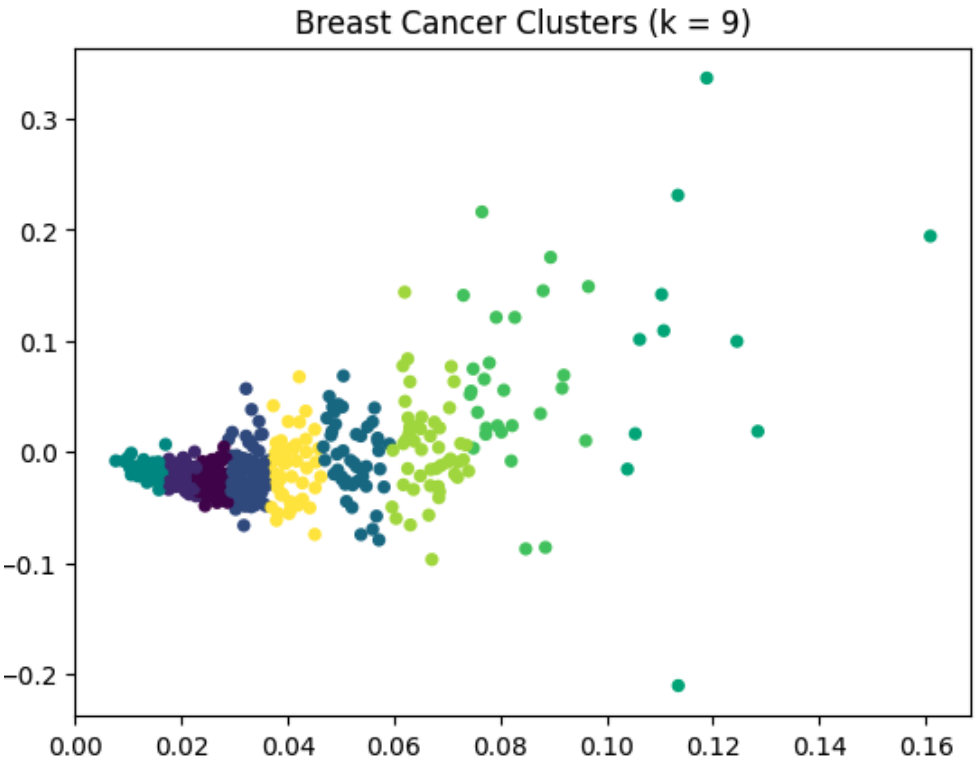
3. **Thus, the top 3 features explained by these PCs together are area error, worst area, and mean area.**

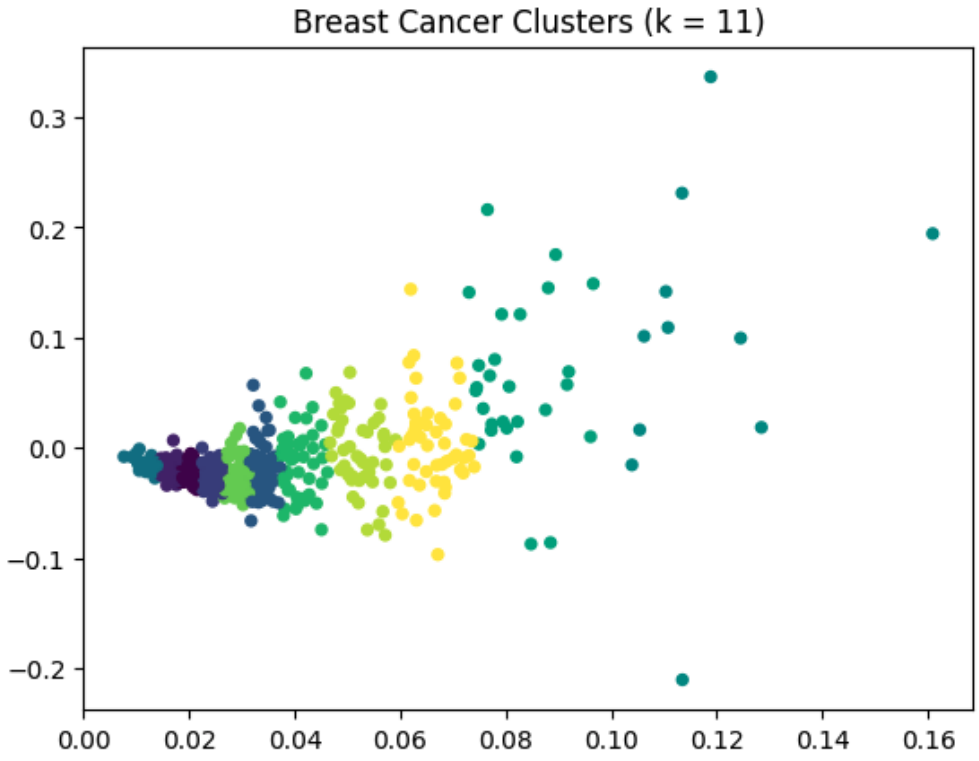
11 Python Programming Question 2.3

1. 

2. 
Breast Cancer Clusters (k = 5)
0.3
0.2
0.1
0.0
−0.1
−0.2
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16

3. 
Breast Cancer Clusters (k = 7)
0.3
0.2
0.1
0.0
−0.1
−0.2
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16

4. 
Breast Cancer Clusters (k = 9)
0.3
0.2
0.1
0.0
−0.1
−0.2
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16

5. 
Breast Cancer Clusters (k = 11)
0.3
0.2
0.1
0.0
−0.1
−0.2
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16