به نام خدا

دانشگاه صنعتی امیرکبیر

دانشکده مهندسی کامپیوتر

پاسخ تمرین سری چهارم شناسایی آماری الگو

استاد:

دکتر رحمتی

دانشجو:

حلیمه رحیمی

شماره دانشجویی:

۹۹۱۳۱۰۴۳

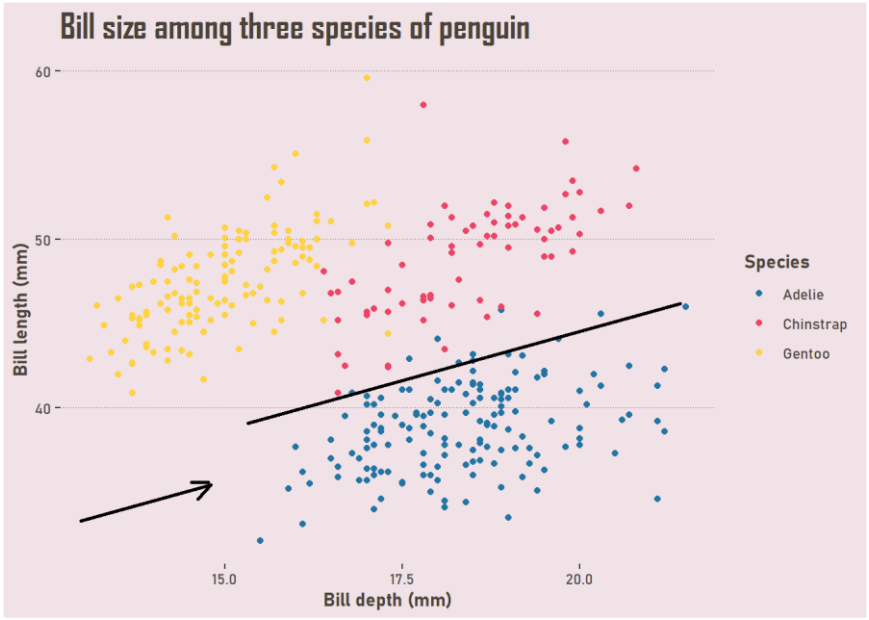
زمستان ۱۳۹۹

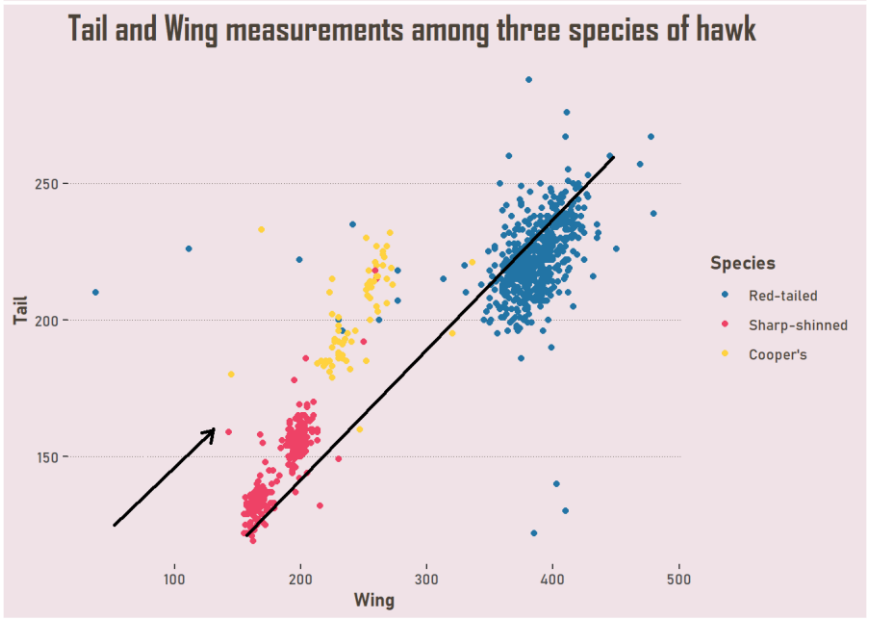
1. PCA vs. LDA: Comparison of Two Different Linear Projections.

The lines without any direction in the questions about PCA present the largest variance and in the questions concerning LDA, they represent the connecting line between the means of the classes.

a. Draw the first principal component direction assuming samples of class blue and red.

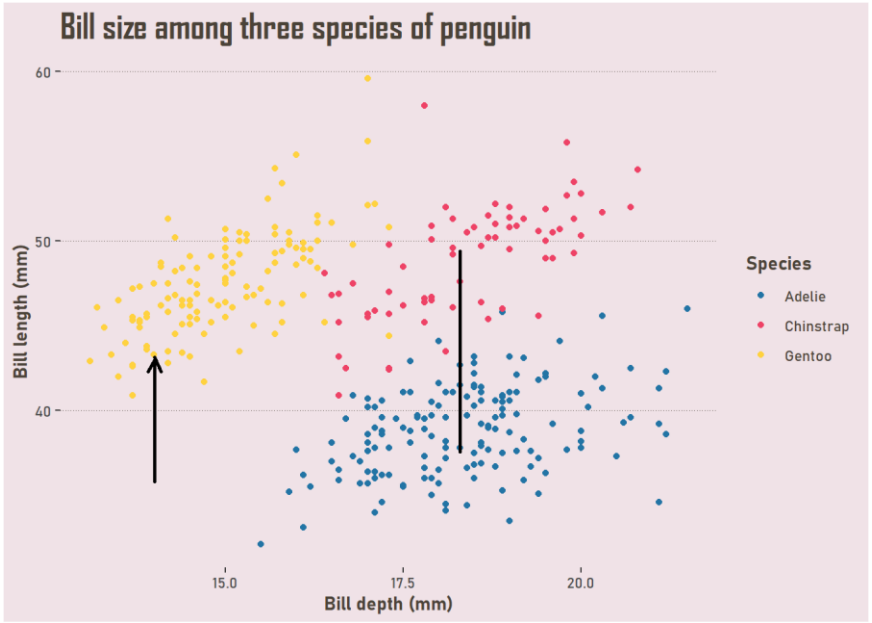
PCA is unsupervised and preserves the direction of the largest variance of the whole data.

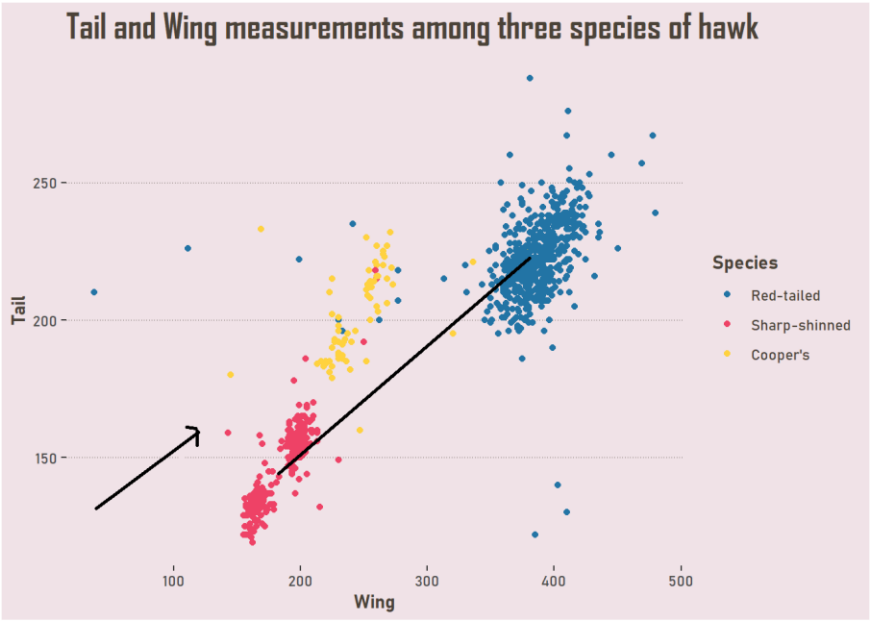




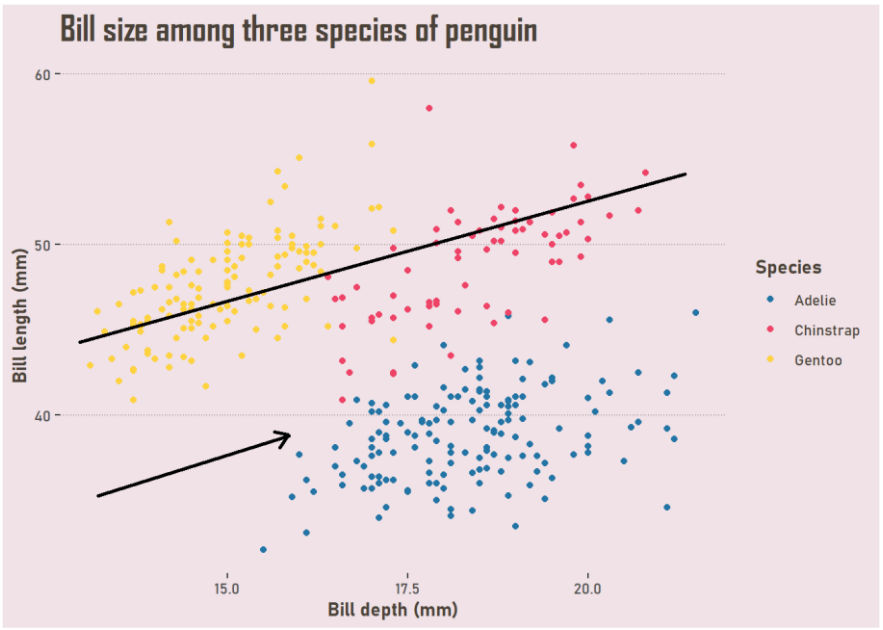
b. Draw the first fisher's linear discriminant direction assuming samples of class blue and red.

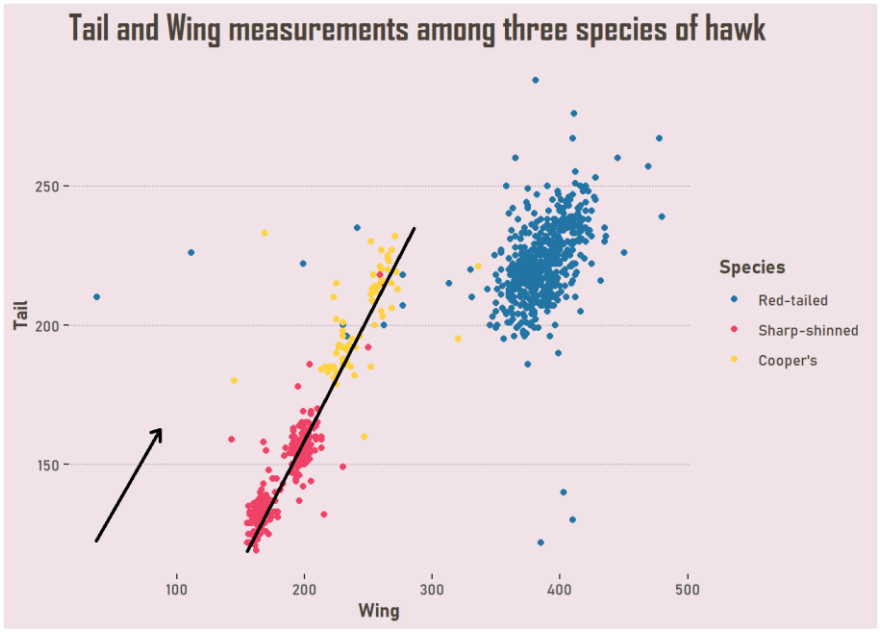
Fisher's linear discriminant is supervised and defines a vector parallel to the line connecting two classes' means.



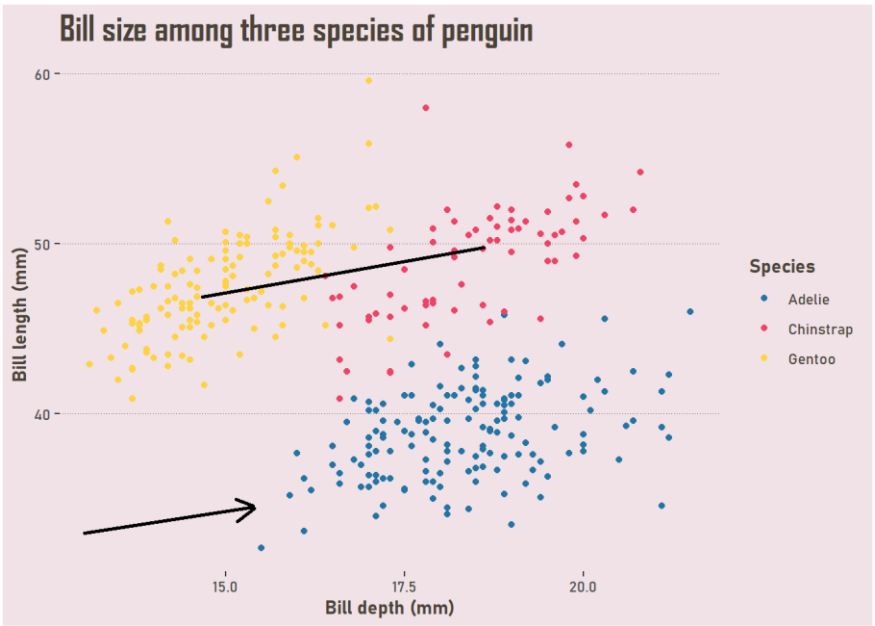


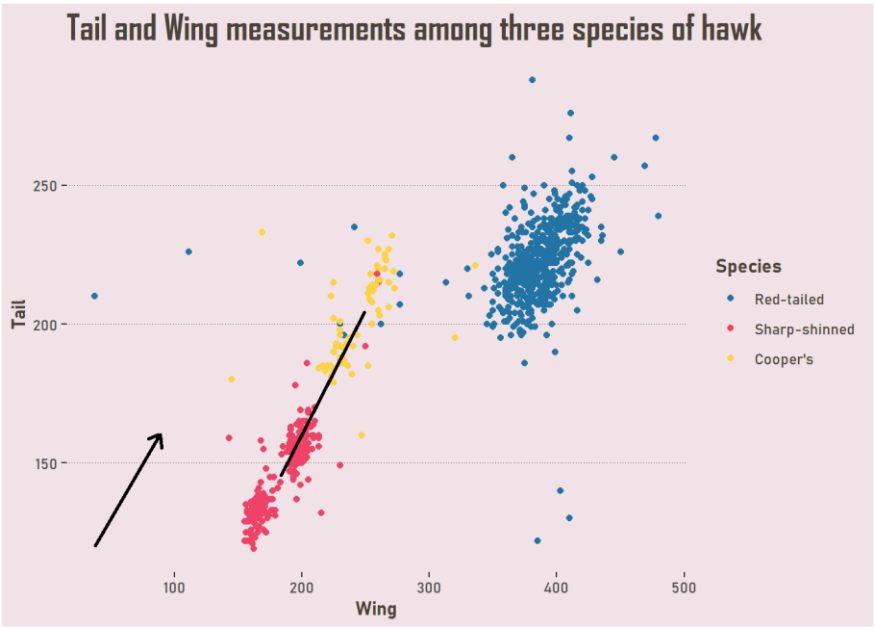
c. Repeat part a. considering samples of class red and yellow.



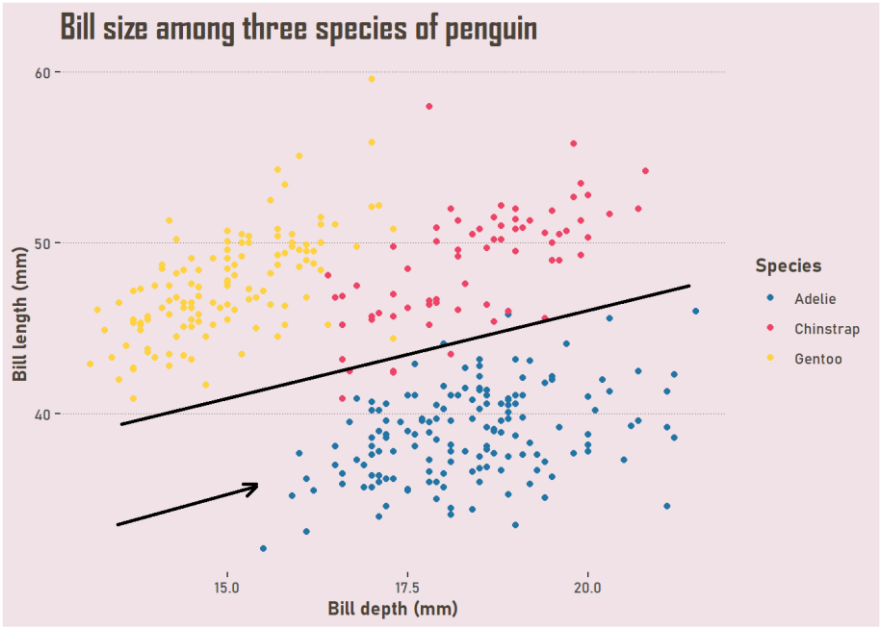


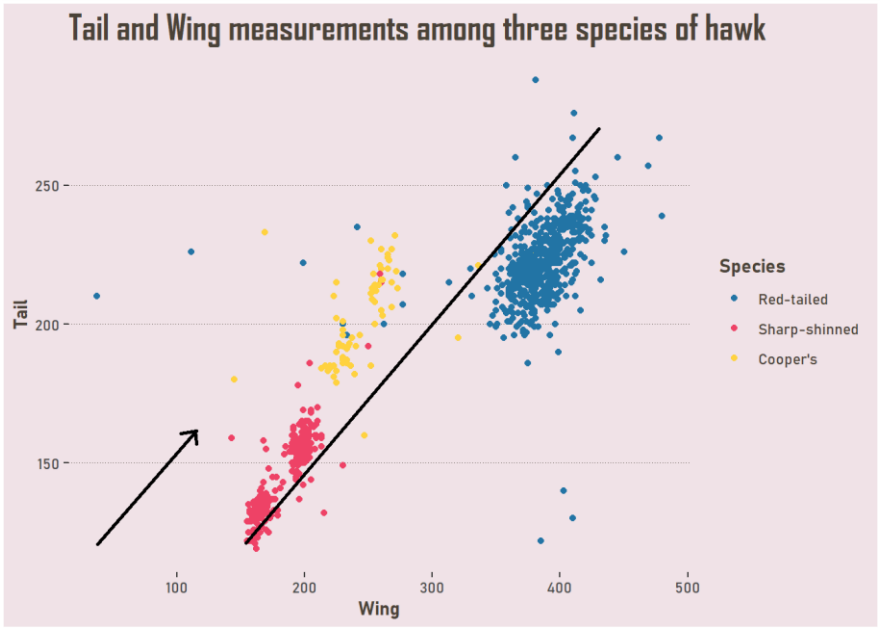
d. Repeat part b. considering samples of red and yellow.





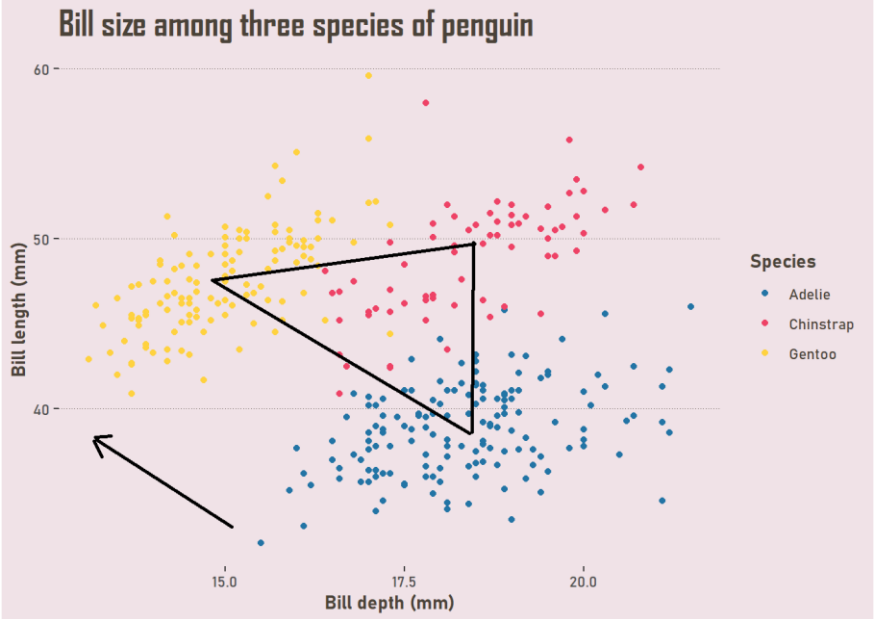
e. Repeat part a. considering all samples.

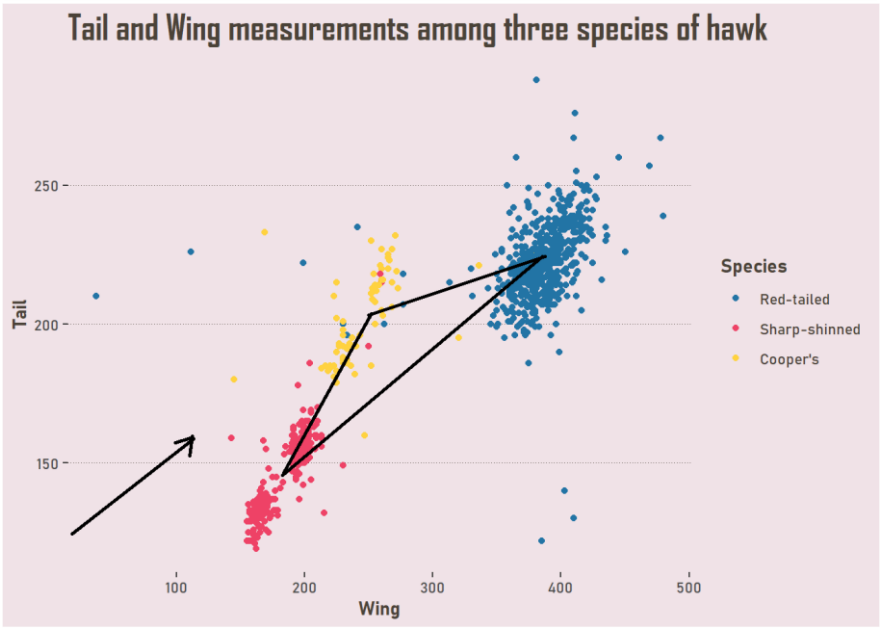




f. Repeat part b. considering all samples.

It's not only the means but also the variances that are accounted. Here as you can see, the variance of each set of data is smaller in the direction of the vector. The variance alongside two other drawn lines in the picture is more than the variance alongside the line parallel to the vector.

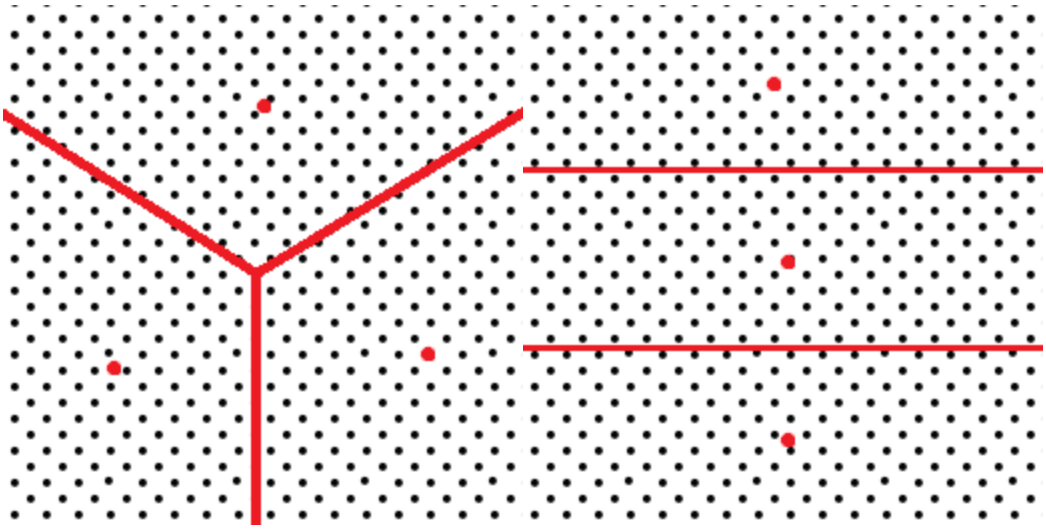




2. Understanding the Behavior of Clustering Techniques

Provide a sketch of how K-Means would split them into clusters considering the given number of clusters. You must also indicate approximately where the final centroids will be. If there is more than one possible solution, then please specify for each solution whether it is a global or local minimum.

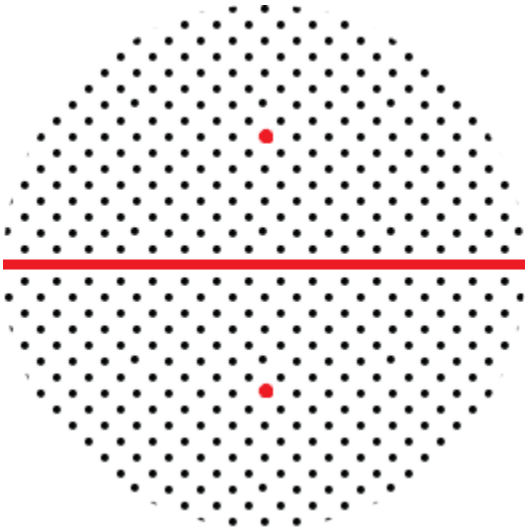
a. K = 3



Both of these are global minimum solutions. Note that by rotating the results in any degrees, you would have a solution.

The second diagram does happen when the starting centroids are near the final centroids drawn in the picture.

b. K = 2

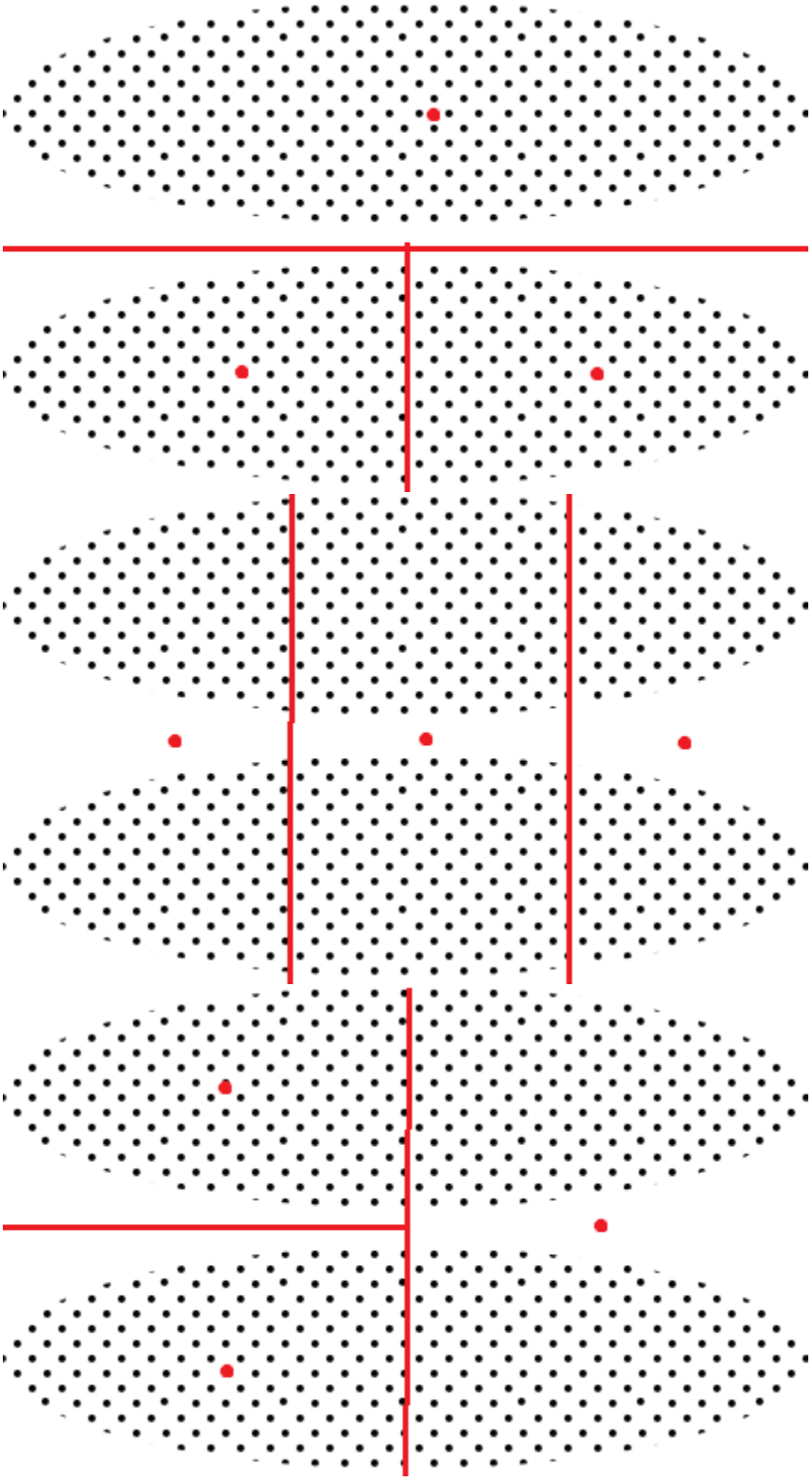


This is the global minimum solution. All other solutions would look like a rotated form of this picture. If you consider two points anywhere in this picture, it will lead to a boundary similar to this.

c. K = 3

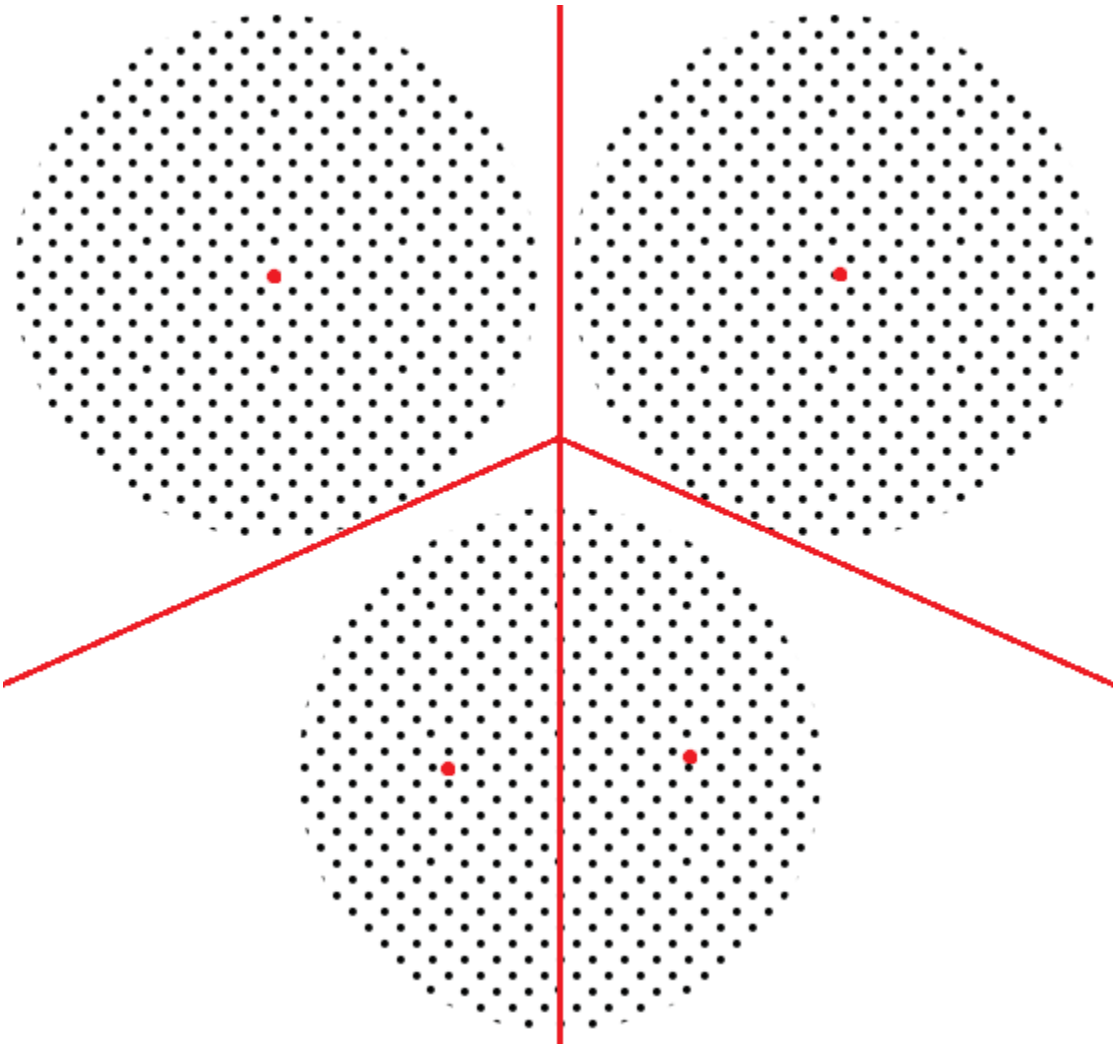
The first picture is the global minimum while the other two are local minimum.

Ending up in the centroids drawn in the pictures depend on the initial centroids.



d. K = 4

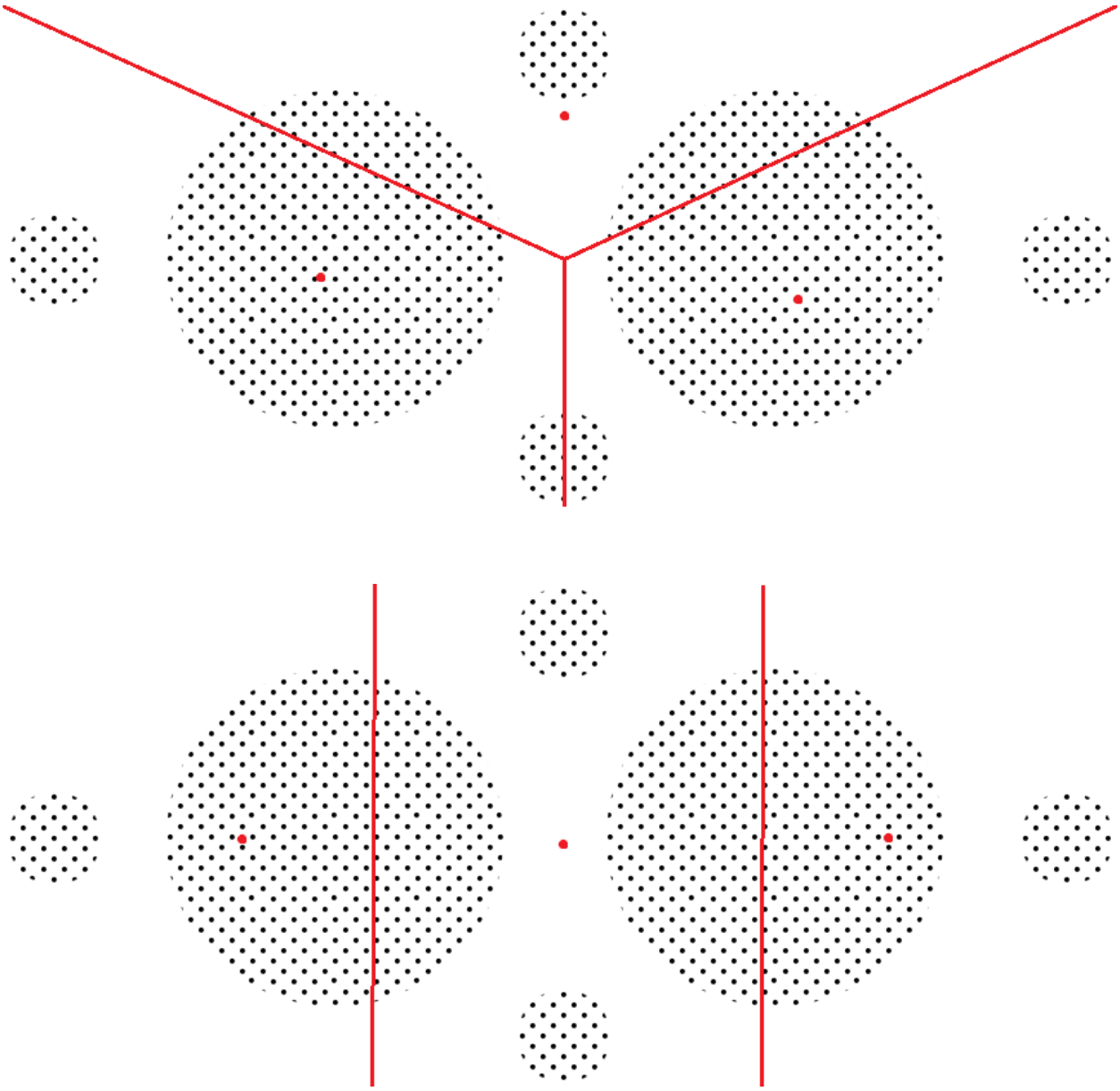
This would be the global minimum result. Even if you consider the initial centroids in one circle, this result would be obtained.



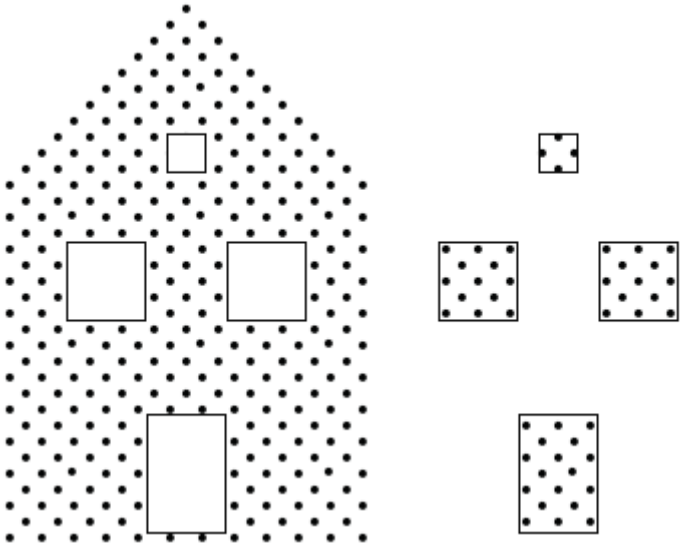
e. K = 3

The First picture shows the global minimum. If you change it upside down you can have another result.

The second picture is a local minimum.



f. **In which one of the two diagrams do the classic clustering techniques, like single linkage, find the patterns represented by door and windows?**



The second diagram (points in the windows and the door) will do better.

The big reason for this is that clustering points will connect the more dense areas faster than the less dense areas.

Consider the first picture; the windows and the door are not the patterns containing points, they're empty spaces that don't even get calculated (only the distance between points get calculated). In more detail, somewhere along the way it is possible for those empty spaces to end up unnoticed with points connecting each other. The small window is way smaller than the door, it is likely that it will vanish into clusters before we get to the empty space that is the door.

Now consider the second picture, the points that are close to one another will connect and there is about a specific distance between the points of a window or the door to another window, therefore by stopping at some level of distance, all of the windows and the door would have their own clusters.

g. Specify the limitations that clustering has in detecting the patterns formed by points.

Like I said in part f, the patterns must not be empty sets, otherwise since clustering starts with the more dense parts, the patterns must contain the more dense area.

3. Why Naples-Style Pizzas Are So Delicious?

a. Calculate the first three principal components of the given data. Are they sufficient to describe most of the variation in the dataset?

In my opinion they are. I calculated the error and how well first three principal components preserve the data information.

```
Eigen Values:
[4.22284393e+03 1.54252434e+03 2.65875883e+01 8.12218533e+00
 1.39520298e+00 3.00227145e-01]
Eigen Vectors:
[[ 8.84370251e-01  4.65810383e-01 -6.22350227e-03 -2.77065339e-02
   1.01664488e-02 -4.53617280e-04]
 [ 4.66360841e-01 -8.83012860e-01  4.41881535e-02  2.82278085e-02
  -6.22213682e-03 -2.78154961e-03]
 [ 1.54064543e-02 -4.07001426e-02 -9.98069103e-01  3.92422699e-02
   2.05966776e-02  1.63023637e-04]
 [-1.21722779e-02 -4.03776158e-02 -3.05303270e-02 -9.58336446e-01
   2.76180414e-01 -5.10393685e-02]
 [-1.57266377e-03 -1.58405327e-03  2.12996568e-02  1.66649576e-01
   7.07231888e-01  6.86725795e-01]
 [ 2.97135932e-03 -4.42859156e-03 -2.19309085e-02 -2.25199372e-01
  -6.50370003e-01  7.25117022e-01]]
```

```
sumlist = list()
s=0
seval = np.sum(eigval)
for k in eigval:
    sumlist.append((s+k)/seval)
print(sumlist)
print('Preserved: ',np.sum(sumlist[:3]))
print('Error: ',np.sum(sumlist[3:]))
```

```
[0.7278539833995054, 0.2658711760073855, 0.004582665692436094, 0.001399948716819384, 0.00024047870594636394, 5.174747
790729514e-05]
Preserved:  0.9983078250993269
Error:  0.0016921749006730432
```

It is shown here that these three principal components preserve 99.8% of the variations. As you know each eigenvalue shows which eigenvector preserves the largest variations, the greater the value of an eigenvalue the more the variation preservation of the corresponding eigenvector.

I need to add that I used numpy.linalg.eig() and the function itself does sort the eigenvalues and eigenvectors and also normalizes the vectors.

b. Use the principal components in different ways to evaluate the normality of the data.

```
from scipy.stats import kstest
```

```
print(kstest((a[:,0]-a[:,0].mean())/np.std(a[:,0]),'norm'))
```

```
KstestResult(statistic=0.11587468662828143, pvalue=0.7729756194935719)
```

```
print(kstest((a[:,1]-a[:,1].mean())/np.std(a[:,1]),'norm'))
```

```
KstestResult(statistic=0.13460958204090784, pvalue=0.6012768534568425)
```

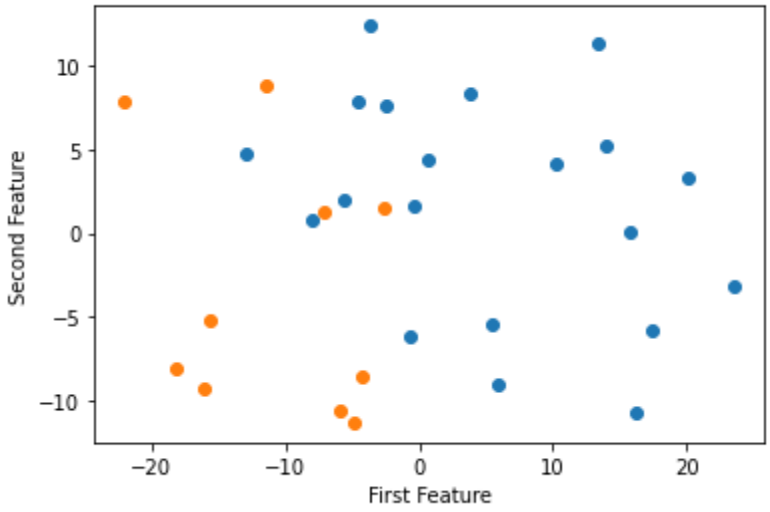
```
print(kstest((a[:,2]-a[:,2].mean())/np.std(a[:,2]),'norm'))
```

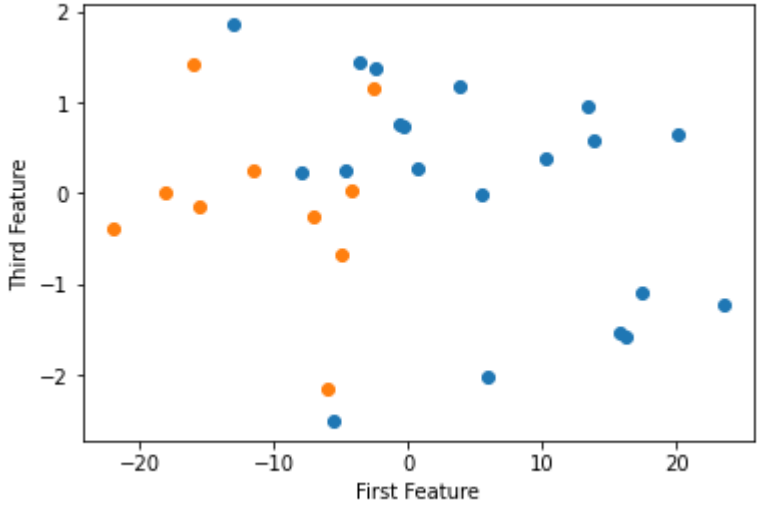
```
KstestResult(statistic=0.13073097313548177, pvalue=0.6372086835056887)
```

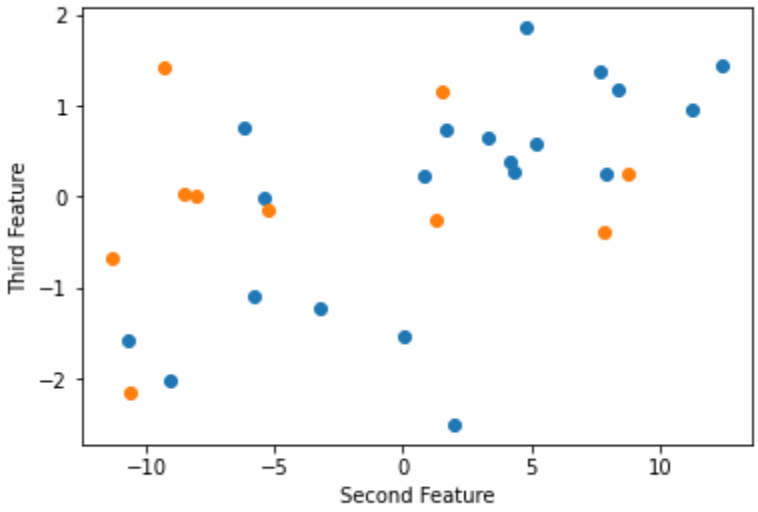
I standardized each principal component because scipy.stats.kstest considers 'norm' as a normal distribution with mean of zero and standard deviation of one. As you can see the p-value has greater value than α=0.05 meaning you can't reject the null hypothesis.

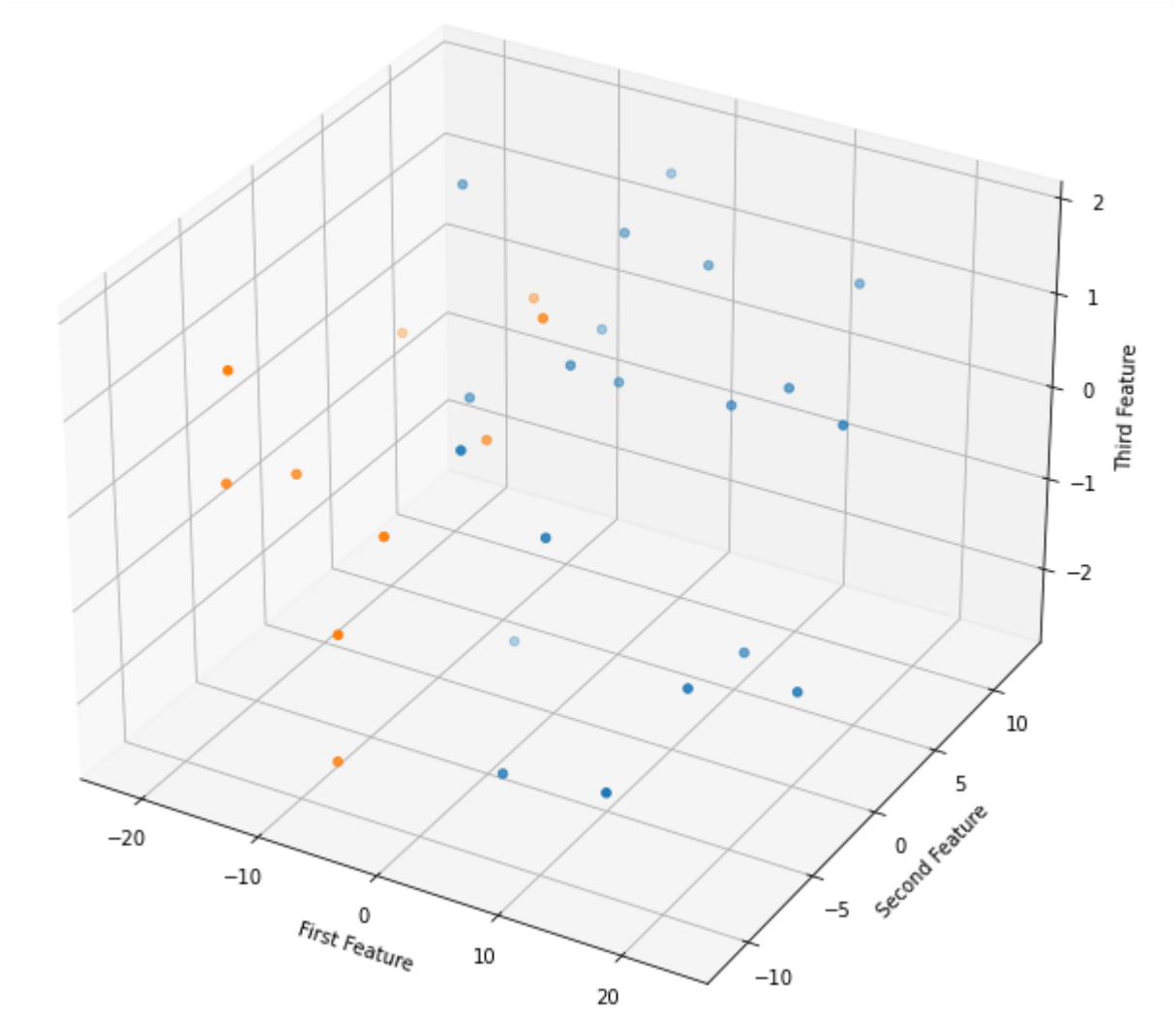
c. Plot two and three-dimensional scatter plots of the first three principal components. Based on the plots, are the first three PCs enough to discriminate between doughs from Naples and doughs from other areas? Does it comply with your conclusion in part (a)?

As you can see, it would not be possible to discriminate the two classes completely by a linear function, however you would get a good accuracy. A polynomial function could give an accuracy of 100% in the training. Also even if we considered the first two principal components or the first and third ones, there would be a good discrimination between the two classes. The first feature alone is preserving the 77% of the variations of the data and in this case the data in each class is scattered in a way that if you project them only on the first principal component you could still get an accuracy above 50% on the training set.


Second Feature
First Feature
10
5
0
−5
−10
−20
−10
0
10
20


Third Feature
First Feature
2
1
0
−1
−2
−20
−10
0
10
20





4. Interpretation of 1000 Genomes Project Results

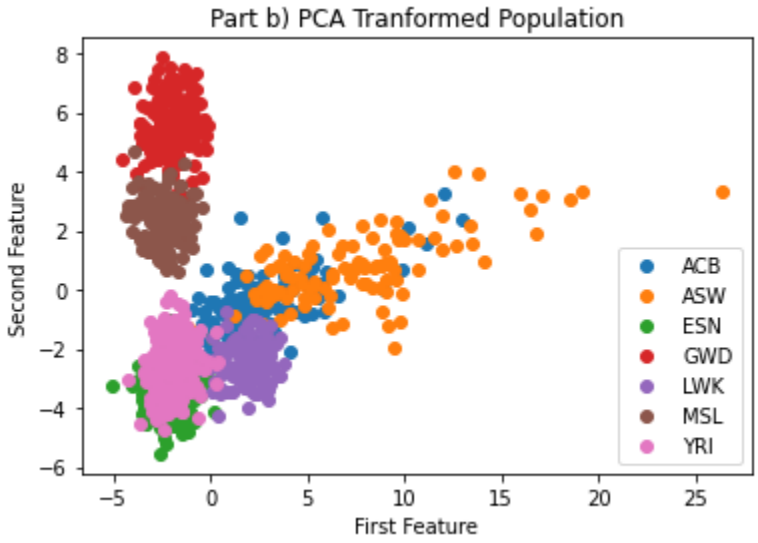
a. What would be the dimensions of the returned vectors after performing PCA on the binary matrix Y?

(995, K)

K indicating the number of principal components we will be using.

b. Project the data onto their first two principal components and display the associated scatter plot. Use different colors to specify each population.

I used sklearn.decomposition.PCA() for these parts.



c. Explain briefly one or two basic facts about the plot you obtained in the previous part. What can you interpret from the first two principal components? What aspects of the data are the first two principal components able to capture?

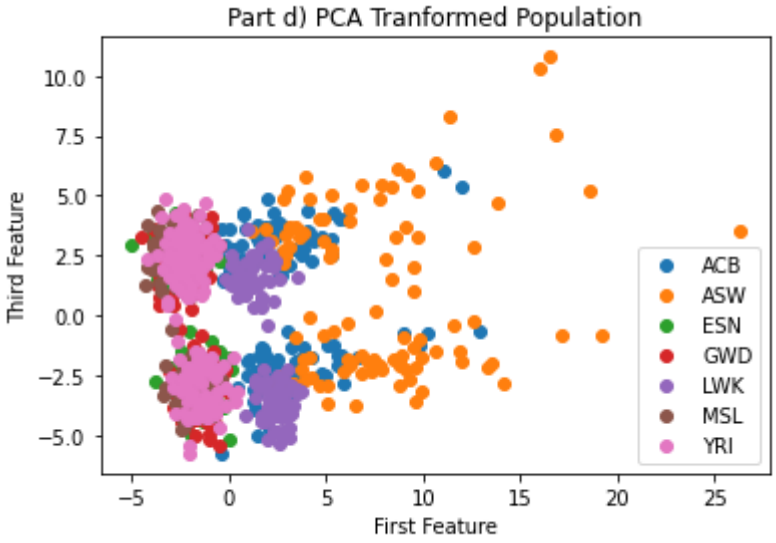
Hint: Think about history and geography.

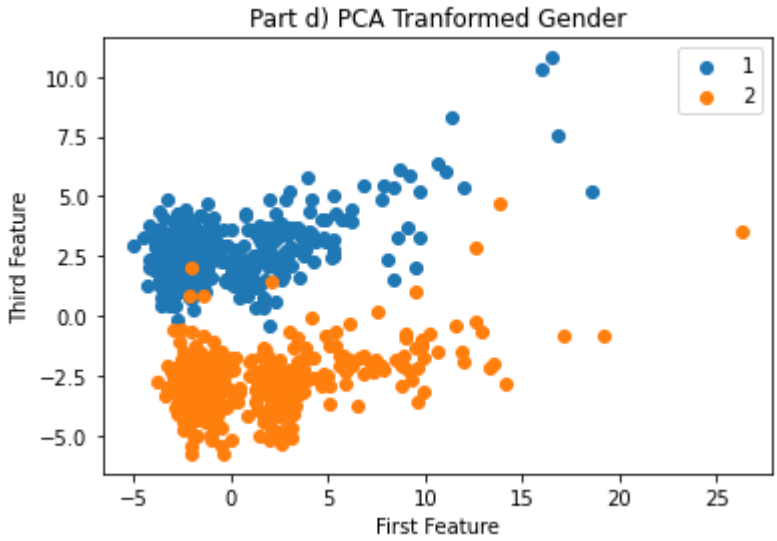
ACB and ASW both indicate the Africans or Americans of African Ancestry who live in America while the others are all in Africa. However as you can see, LWK and ACB points have some noticeable intersection and that is because a lot of slaves were brought to America from Kenya (LWK: Luhya in Webuye, Kenya).

YRI (Yoruba in Ibadan, Nigeria) and ESN (Esan in Nigeria) also have quite the intersection. MSL (Mende in Sierra Leone) and GWd (Gambian in Western Divisions in the Gambia) are close on the map, both in the east of Africa.

This plot shows that whether it was the geography or genes passed from other population types, there were some changes in African Americans while still keeping some of the similarities to their ancestors in Africa.

d. Now create another scatter plot with each individual projected onto the subspace spanned by the first and third principal components. Then play with different labeling schemes (with labels derived from the meta-data) to explain the clusters that you see.





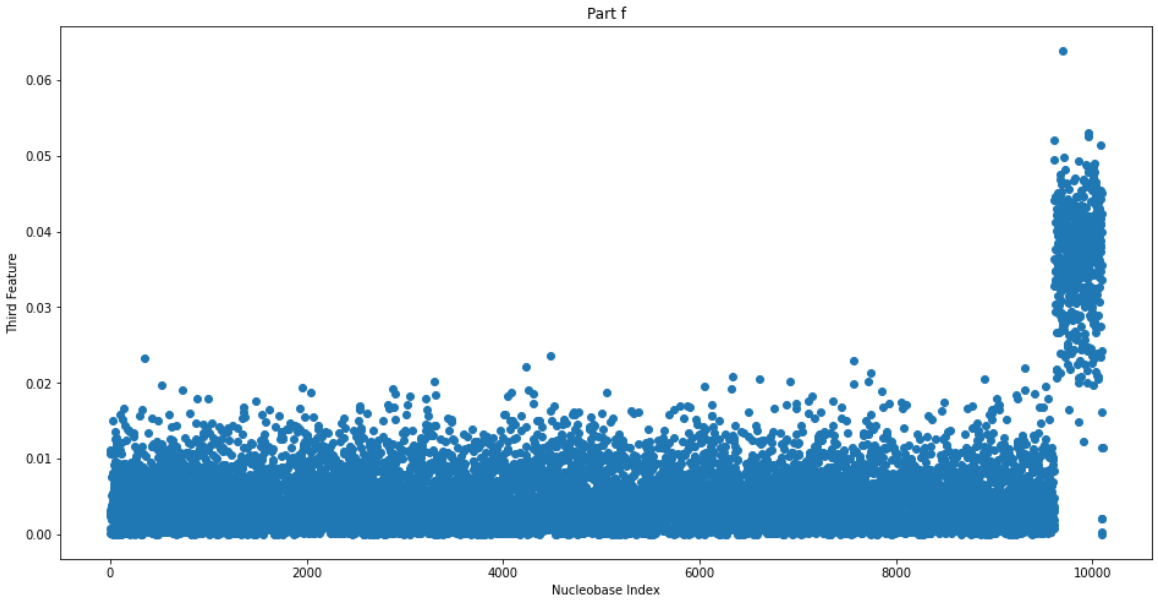
As you can see the subspace spanned by the first and third principal components can discriminate people's gender really well while not presenting well the population they belong. Therefore the third principal component captures gender.

e. Briefly explain what information does the third principal component capture?

The third principal component captures gender. If you project all the data on the third feature, you can see that the gender is discriminable with high accuracy at one point around zero.

f. Plot the nucleobase index versus the absolute value of the third principal component. What do you observe? Give a possible explanation.

Hint: Think about chromosomes.



It is possible that nucleobase index 10000 to 10101 define the characteristics of a female and male body. Probably it is about the chromosome difference between women and men. Each person normally has one pair of sex chromosomes in each cell. Females have two X chromosomes, while males have one X and one Y chromosome.

5. How Twitter Reacts to a Crisis?

a. Use the provided initial centroids and cluster the tweets in K = 25 clusters. The output must be a file which contains the clustering results such that each line represents a cluster in the form of: cluster_id: a list of tweet IDs which belong to this cluster. Include this file in your report.

You can find the file in folder 'P5'.

Here are the centroids computed at each iteration. It took two iterations to converge.

I have six functions: two of them I wrote to help with the Jaccard Distance Calculation. 'Init_centroids' is for the initialization. This function gets a parameter 'rand' that if we want to give the initial centroids ourselves we have to put it equal to False or else it will choose the initial centroids similar to kmeans++ algorithm, which I used for question b. 'find_closest_centroids' will calculate the closest centroid to each data point and include a column by the name of 'cluster' to the dataset. 'calc_centroids' will find the point in each cluster that has the least mean distance to other points in its own cluster. 'kmeans' is the function that connects these last three functions to each other.

```
K-means with given Initial Centroids:
Initial Centroids:
 ['323906397735641088', '323906483584655360', '323906657333682176', '323907258301939713', '323909308188344320', '3239
13403460636673', '324067437886713856', '324117950774775809', '324138055772561408', '324219503401644033', '32432024701
8573824', '324346553835868161', '324372750330363904', '324408472441585664', '324422817565257728', '32444801399930470
4', '324785120085176320', '325059351209443329', '325060324992643072', '325162944931438592', '325253327048822784', '32
5337623910559745', '325409910642835456', '325701934273134594', '325946633986641920']
----------
New Centroids:
[323906397618196483, 323906485249789952, 323906651209994241, 323906398176030720, 323909308188344320, 3239134034606366
73, 324038827578626048, 324117950774775809, 324070589214117888, 324375472681148416, 324427314060484608, 3243482527638
73283, 324372750330363904, 324408861563944960, 324422817565257728, 324448013999304704, 324785129241341952, 3250593512
09443329, 325060154087309312, 325171606253080577, 325253327048822784, 325335253487063041, 325409910588317696, 3257019
32083716096, 325946283368013824]
----------
New Centroids:
[323906397618196483, 323906485249789952, 323906651209994241, 323906398176030720, 323909308188344320, 3239134034606366
73, 324038827578626048, 324117950774775809, 324070589214117888, 324375472681148416, 324427314060484608, 3243482527638
73283, 324372750330363904, 324408861563944960, 324422817565257728, 324448013999304704, 324785129241341952, 3250593512
09443329, 325060154087309312, 325171606253080577, 325253327048822784, 325335253487063041, 325409910588317696, 3257019
32083716096, 325946283368013824]
```

b. Design and implement an efficient method to find the K initial centers so that K-Means can generate good clustering results similar to the results you obtained in the previous part.

I did also save a text file containing the list of data points in each cluster obtained from this part.

The only difference here occurs in 'init_centroids'. The first point is chosen randomly and then the distance of each point to the first centroid is calculated, the point with the most distance will be chosen as the next centroid. Every time I drop the chosen centroid in order to avoid choosing the same point again. The same procedure happens on a loop, calculating the point that has the largest distance from all of the previously chosen centroids (by considering the minimum distance the point has to a centroid), until 'k' centroids are found.

You can see the initial and final centroids in the picture below. Centroids converged after two iterations. This shows that this procedure does present a good initialization for this dataset.

```
K-means++:
Initial Centroids:
 [323906397735641088, 324117950774775809, 325253327048822784, 323906398176030720, 324320247018573824, 324419757661097
984, 323946504752603136, 324372750330363904, 324188481918230528, 325172960413503489, 324432383115935744, 323909308188
344320, 325917940564250624, 325060145082163200, 325702031912349696, 324448013999304704, 325059351209443329, 325342218
942554112, 323906651209994241, 324038827578626048, 324785120085176320, 324346553835868161, 325410327296614400, 323906
483584655360, 324422817565257728]
----------
New Centroids:
[323906397618196483, 324117950774775809, 325253327048822784, 323906398176030720, 324427314060484608, 3243754726811484
16, 323913403460636673, 324372750330363904, 324070589214117888, 325171606253080577, 324408861563944960, 3239093081883
44320, 325946283368013824, 325060154087309312, 325701932083716096, 324448013999304704, 325059351209443329, 3253352534
87063041, 323906651209994241, 324038827578626048, 324785129241341952, 324348252763873283, 325409910588317696, 3239064
85249789952, 324422817565257728]
----------
New Centroids:
[323906397618196483, 324117950774775809, 325253327048822784, 323906398176030720, 324427314060484608, 3243754726811484
16, 323913403460636673, 324372750330363904, 324070589214117888, 325171606253080577, 324408861563944960, 3239093081883
44320, 325946283368013824, 325060154087309312, 325701932083716096, 324448013999304704, 325059351209443329, 3253352534
87063041, 323906651209994241, 324038827578626048, 324785129241341952, 324348252763873283, 325409910588317696, 3239064
85249789952, 324422817565257728]
```

In the end, I used Jaccard Similarity to show that the outputs from part a. and part b. are the same.

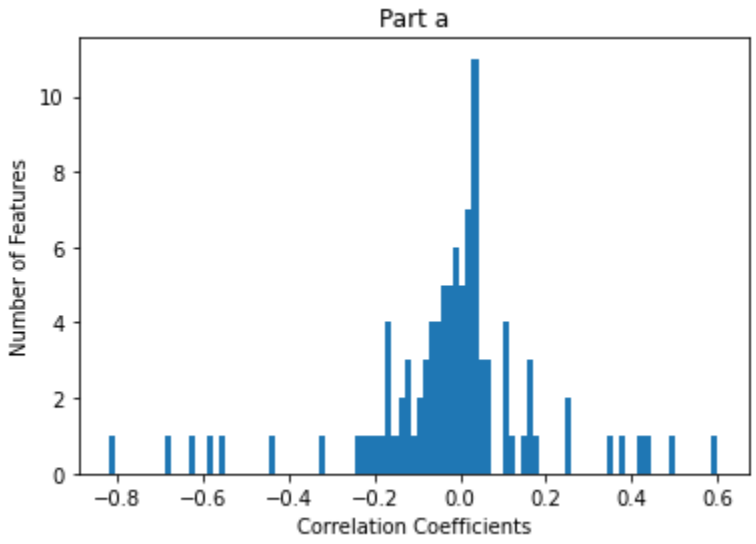
```
list1 = centroids1
list2 = centroids2
print('Similarity between the chosen centroids in part a and b:\n', jaccard_similarity(list1, list2))
```

```
Similarity between the chosen centroids in part a and b:
 1.0
```

6. A Tribute to Stephen Hawking

a. Select 10 features with highest covariance with ALSFRS slope. You must support your choice by appropriate visualization.

Greater absolute correlation coefficients between each feature with ALSFRS slope shows higher covariance. So I computed the coefficients and plotted a histogram indicating the number of features that have the corresponding coefficients.



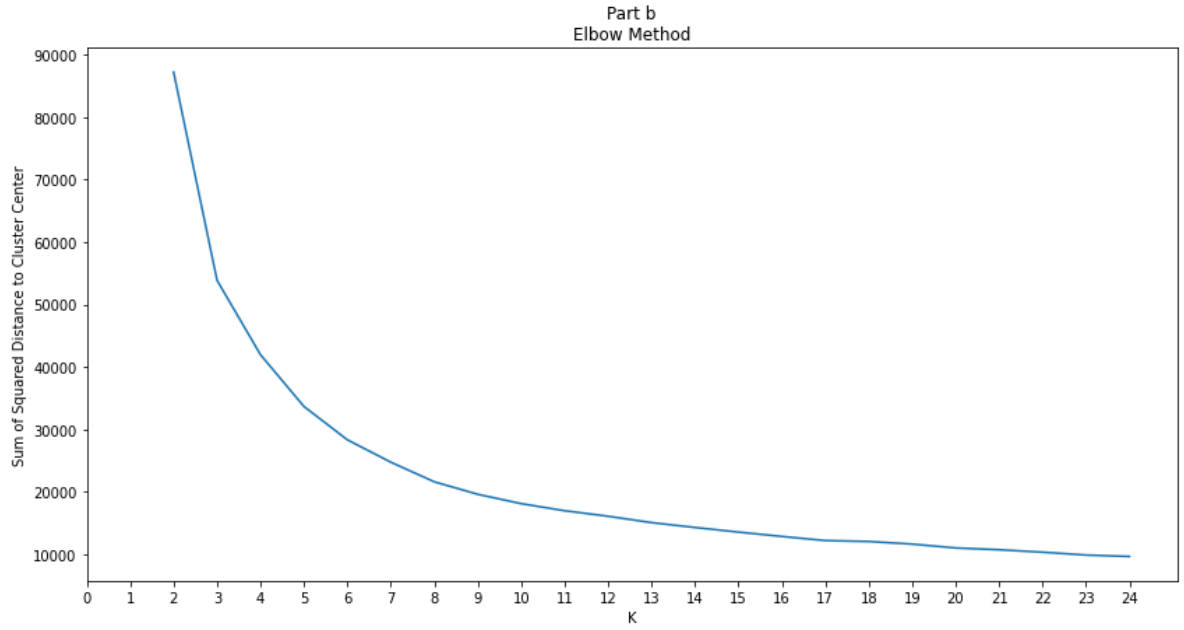
As you can see the ten best features in this question have absolute value of correlation coefficient more than 0.4.

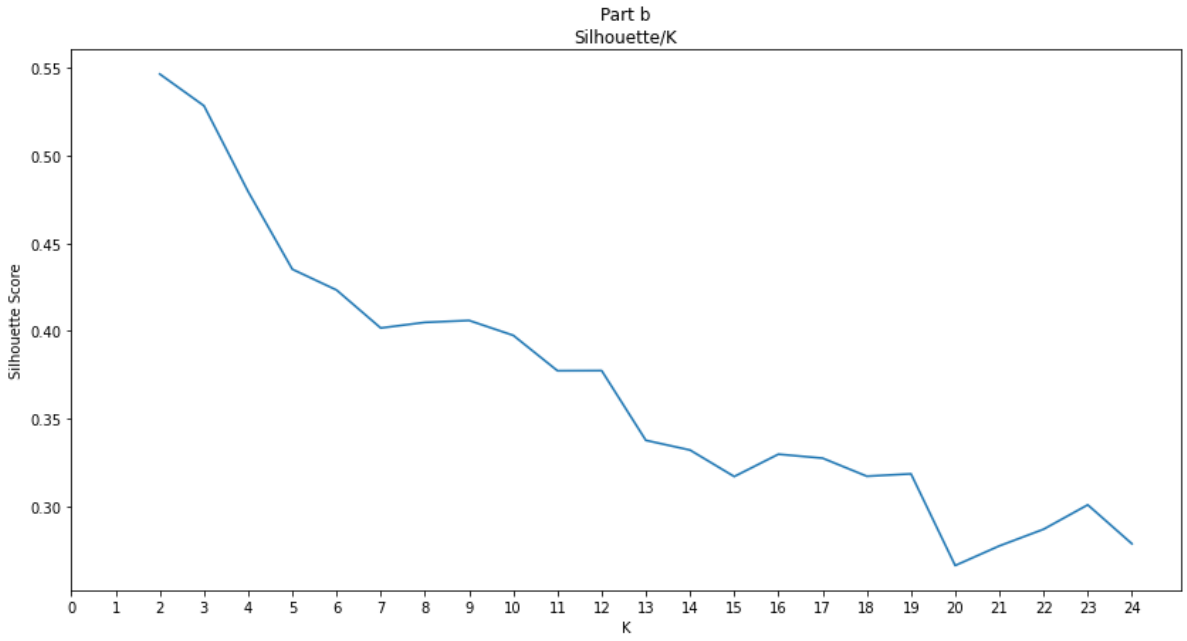
This can also be seen when you print the absolute values of the coefficients of those ten best features:

```
ALSFRS_Total_range    0.819305
trunk_range           0.686407
hands_range           0.632350
ALSFRS_Total_min      0.601810
leg_range             0.584507
mouth_range           0.553194
trunk_min             0.488530
mouth_min             0.435538
respiratory_range     0.432878
hands_min             0.424107
dtype: float64
Best Features:
 Index(['ALSFRS_Total_range', 'trunk_range', 'hands_range', 'ALSFRS_Total_min',
       'leg_range', 'mouth_range', 'trunk_min', 'mouth_min',
       'respiratory_range', 'hands_min'],
      dtype='object')
```

b. Train a K-Means model on the reduced dataset. Using a clear strategy, find the best possible value for the parameter K and use it afterwards.

I plotted both The Elbow Method and Silhouette/K for k=2 to 24.

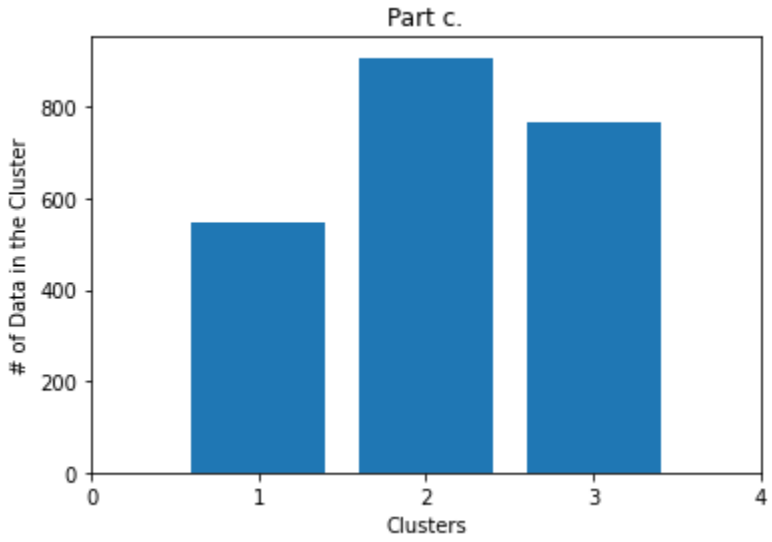




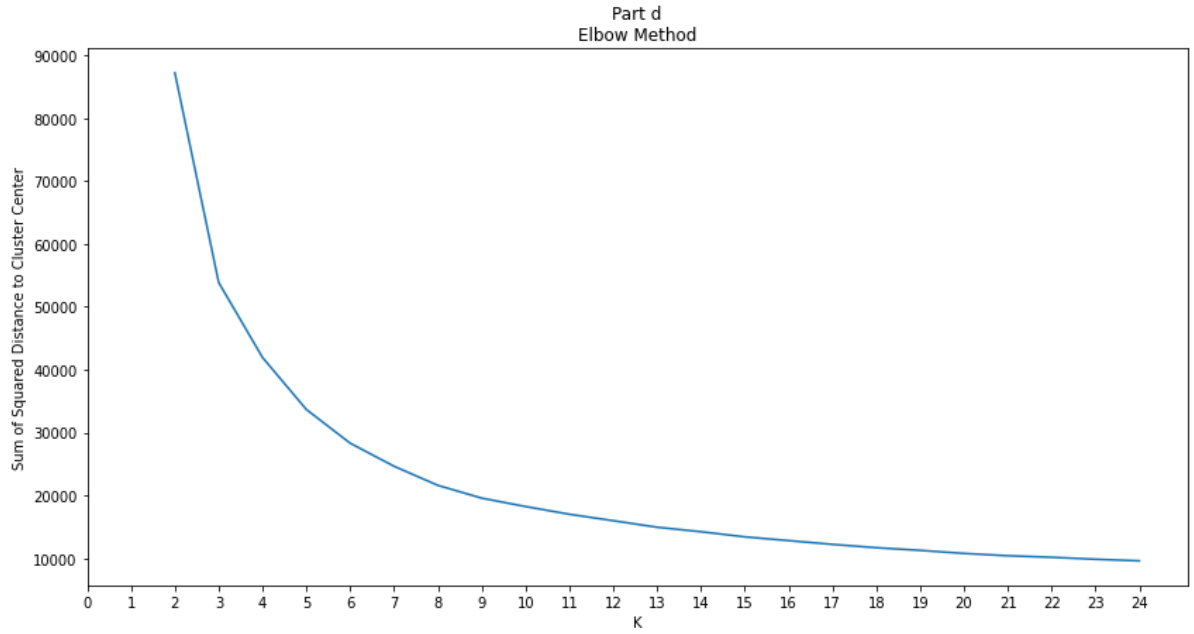
In the elbow method there are some distortions at 3, 4 and 5. I thought in order to get better silhouette, I consider k=3 needed to tell that the silhouette above 0.5 is a better choice.

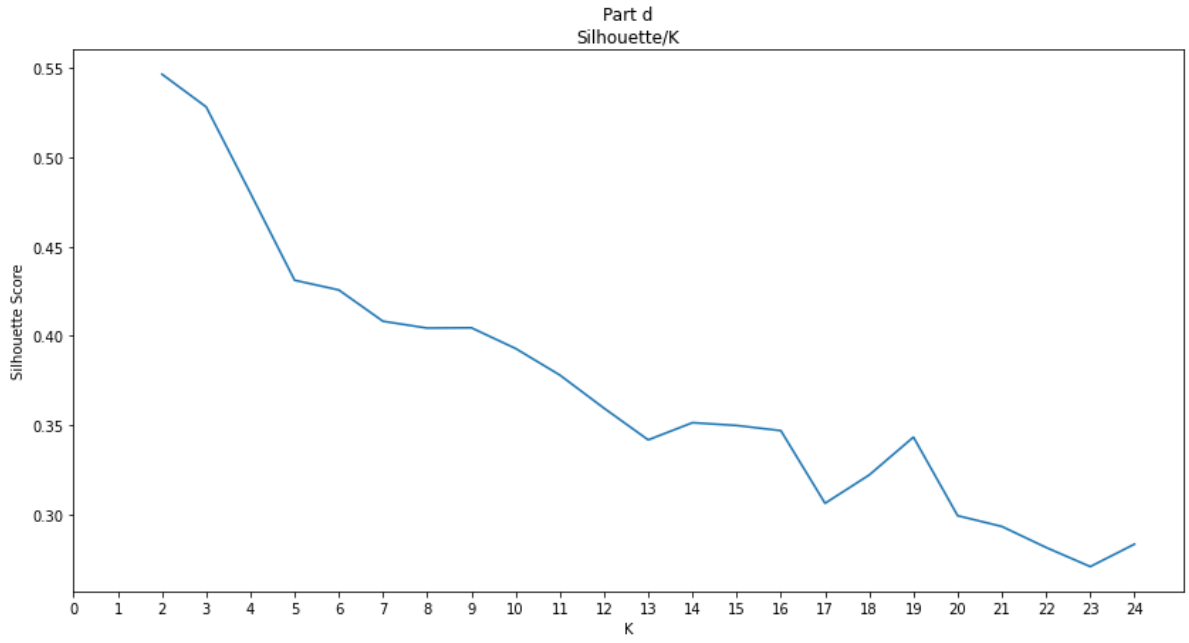
c. Evaluate the model performance by reporting the centre of clusters and silhouette, and explain the details. Also use bar plot to show the centres.

```
Part c, Centroids:
 [[4.23871766e-02 1.00976663e-02 9.64347110e-03 8.69160584e+00
  8.34827940e-03 1.32495969e-02 6.66058394e-01 3.71532847e+00
  4.14316504e-03 7.57299270e-01]
 [2.72241502e-02 7.90387219e-03 7.93094910e-03 1.86112335e+01
  6.50006705e-03 5.85648443e-03 2.31387665e+00 8.22797357e+00
  2.57026244e-03 2.32599119e+00]
 [1.29451082e-02 4.11012977e-03 3.66931079e-03 2.93676662e+01
  4.20179500e-03 2.71340720e-03 5.35202086e+00 1.01486310e+01
  1.28188620e-03 5.53715776e+00]]
Part c, Silhouette:
 0.5282006367866574
```



d. Tune parameters and plot with K-Means++.





Even though k=4 has a better silhouette in k-means++, I still choose k=3 because its silhouette is above 0.5.

e. Return the model with optimal parameters and interpret the clustering results.

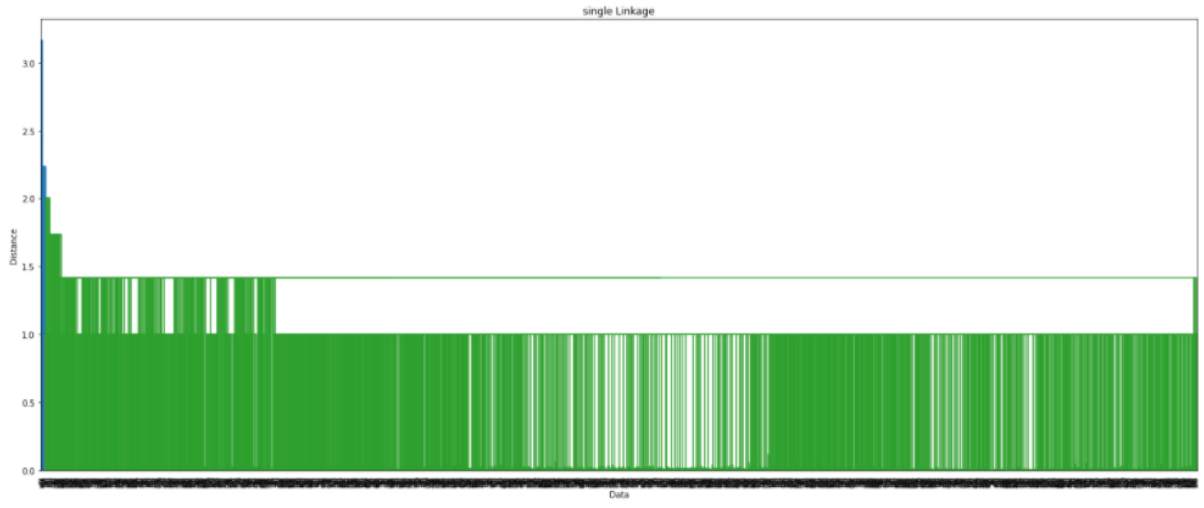
```
Part e, Centroids:
 [[2.72241502e-02 7.90387219e-03 7.93094910e-03 1.86112335e+01
  6.50006705e-03 5.85648443e-03 2.31387665e+00 8.22797357e+00
  2.57026244e-03 2.32599119e+00]
 [4.23871766e-02 1.00976663e-02 9.64347110e-03 8.69160584e+00
  8.34827940e-03 1.32495969e-02 6.66058394e-01 3.71532847e+00
  4.14316504e-03 7.57299270e-01]
 [1.29451082e-02 4.11012977e-03 3.66931079e-03 2.93676662e+01
  4.20179500e-03 2.71340720e-03 5.35202086e+00 1.01486310e+01
  1.28188620e-03 5.53715776e+00]]
Part e, Silhouette:
 0.5282006367867924
```

I need to say that may be choosing 2 clusters would have a better result. It does result a better silhouette and also in the final centroids from three clusters, it's easy to see that the first and third centroids are close to each other.

Average silhouette score more than 0.5 means more than half of the data is most likely to be in the right cluster. This is based on the samples having less distance to their own cluster data points rather than the ones in other clusters.

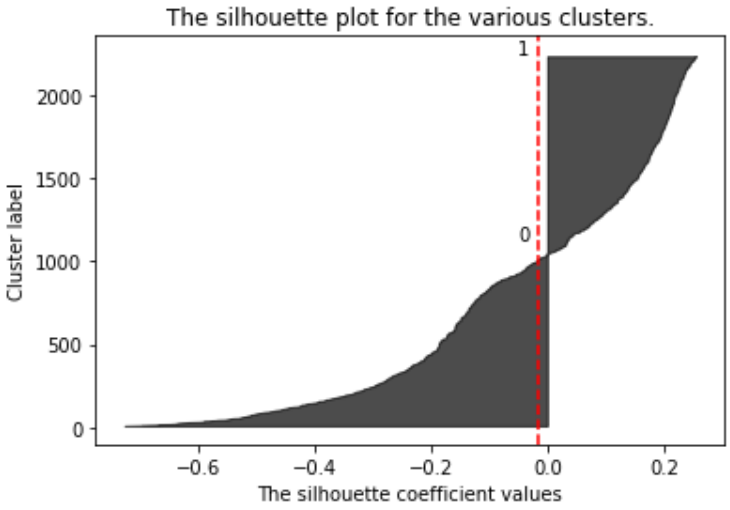
f. Apply hierarchical clustering on three different linkages and compare the corresponding silhouette plots along with dendrograms.

Single Linkage:



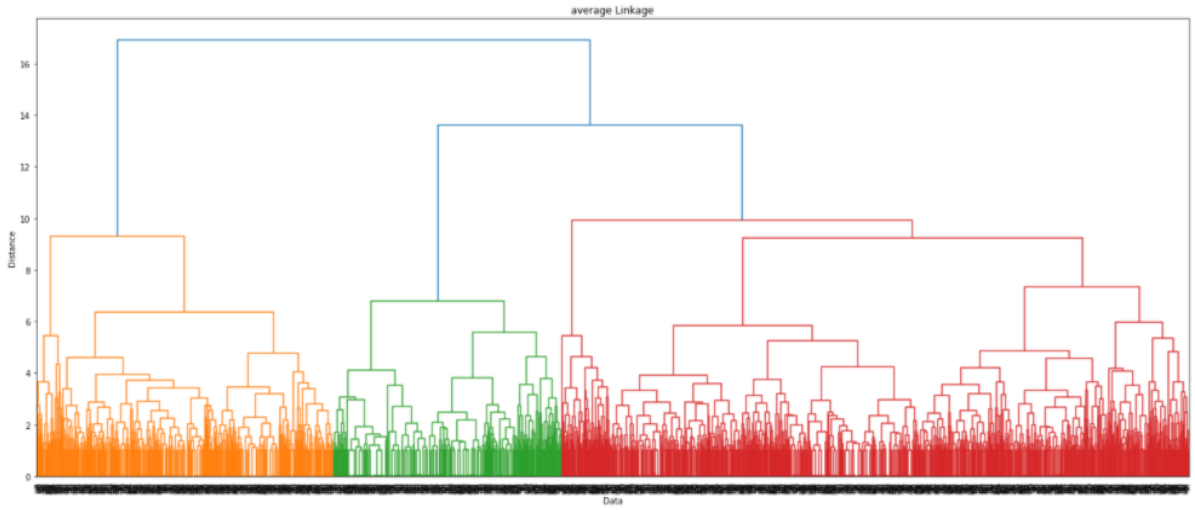
It seems the data points are quite close and that is why when using single linkage, almost all the data connect one another like chain and create one giant cluster and a small cluster.

```
For n_clusters = 2 The average silhouette_score is : -0.015897276670669428
```



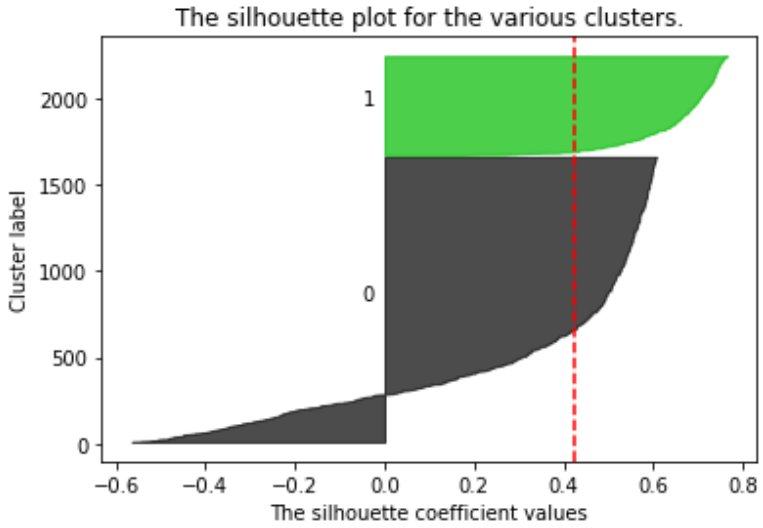
This is showing that almost all the data is assigned to cluster zero.

Average Linkage:



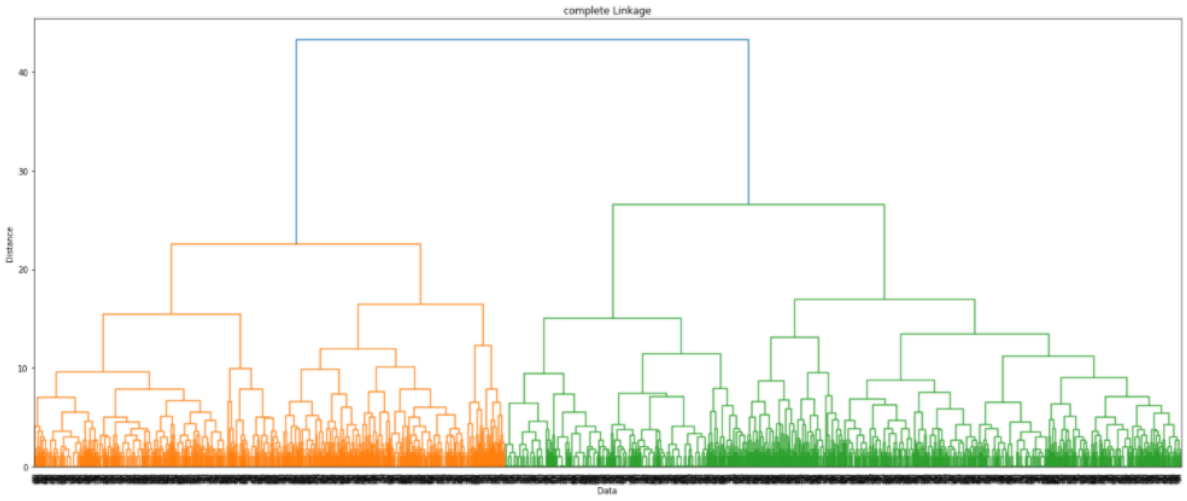
Average Linkage does better than single linkage. It seems there are nice blobs containing each cluster.

```
For n_clusters = 2 The average silhouette_score is : 0.42206597911651156
```



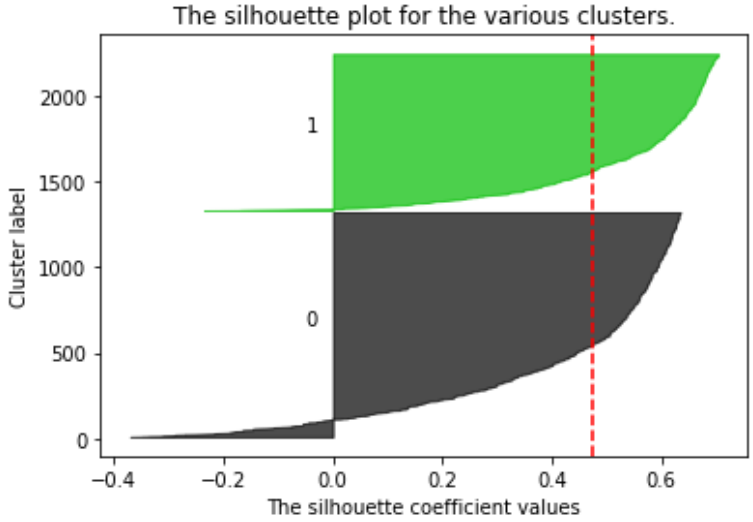
As you see, the average silhouette score is 0.42 therefore although it is not more than 0.5, it is better than single linkage silhouette score. Also the number of data assigned to cluster zero is more than the ones assigned to one. It might be better if some of the data in cluster zero get assigned to cluster one.

Complete Linkage:



In this dendogram one can see a clean separation between data points assigned to cluster zero and one.

```
For n_clusters = 2 The average silhouette_score is : 0.4736813820725446
```



The Average Silhouette score for complete linkage is better than the previous methods, meaning it seems the data points are better assigned.

g. Compare the results of the above methods.

Based on the things said in the last part since single linkage usually forms thin and long clusters, the data based on the ten chosen features are not in thin and long clusters and they probably are in blob-like clusters. This gets obvious with complete linkage resulting a better score since this method does form compact and blob-like clusters.

7. Are You Into Fashion?

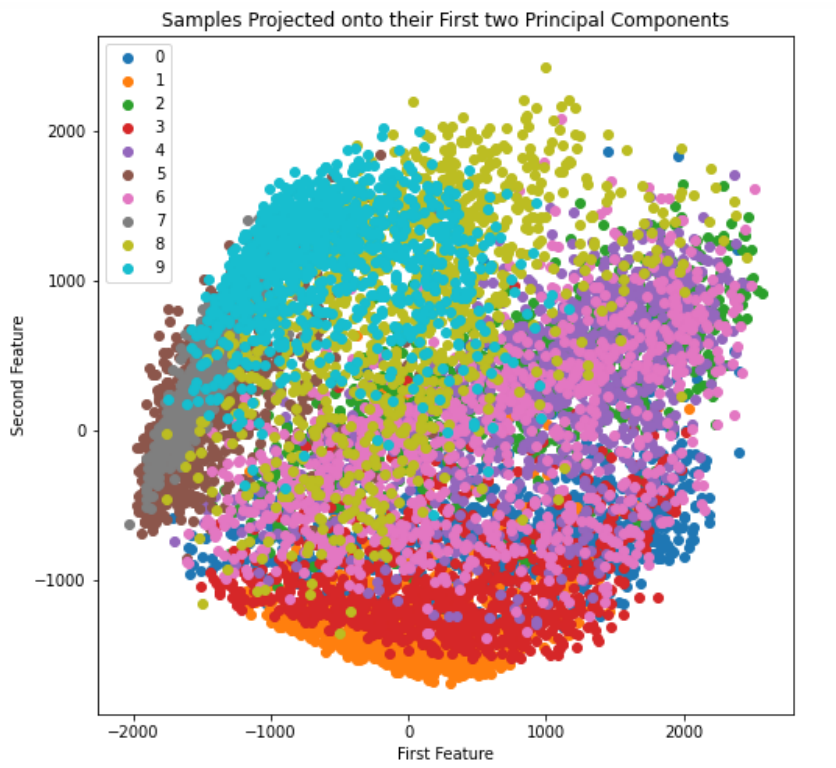
a. Perform PCA on the dataset to reduce its dimensions. Display the top 20 eigenvalues as well as the representation of the samples projected onto their first two and three principal components.

I used sklearn.decomposition.PCA() to compute the principal components.

Eigenvalues and eigenvectors can be accessed as attributes of this object, respectively 'components_' and 'explained_variance_'.

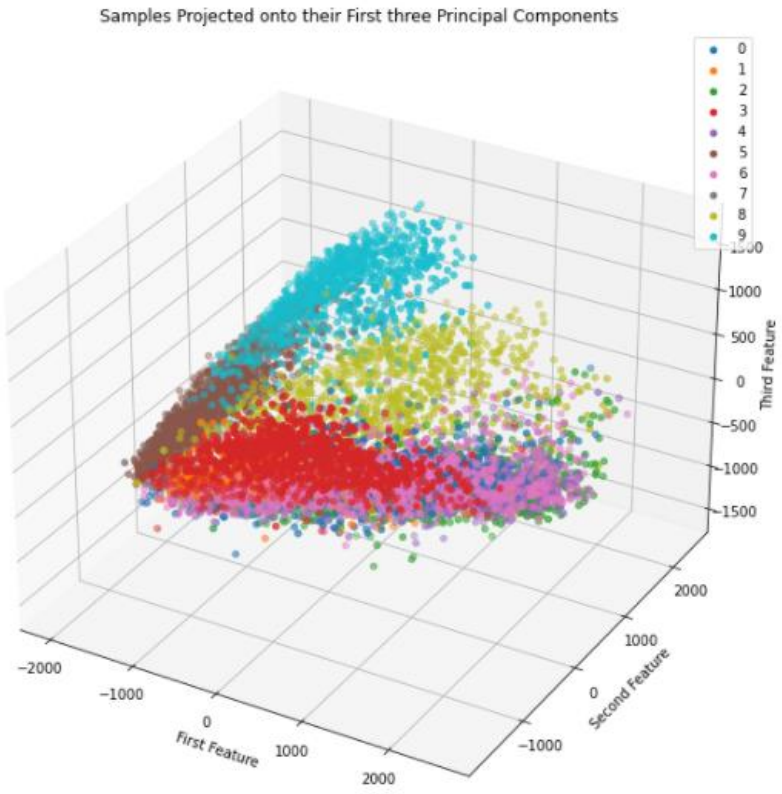
```
Top 20 Eigenvalues:
 [1288319.52477778  779197.62253773  265730.43854769  218669.76933454
  169257.23458071  152452.76424582  104674.41864859   83982.28146176
   58343.40694262   57195.68413797   43687.97445268   40723.99121484
   33555.72854707   28600.97342107   27417.8420902    25851.22397945
   24625.32484158   23565.45043936   21254.55872949   19657.63948351]
```

Fit_transform(x) will fit the data (x) and return the principal components.



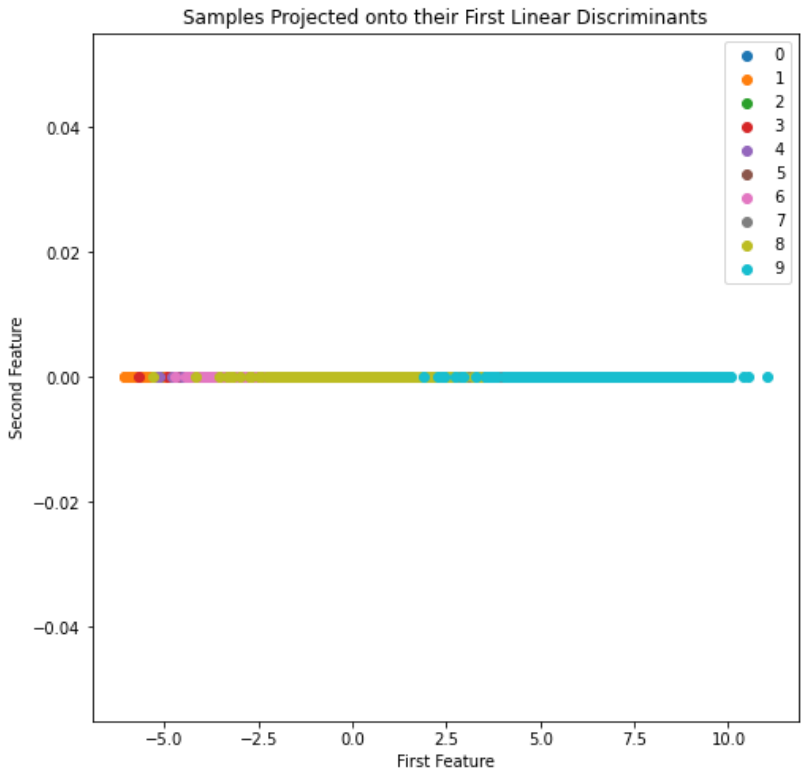
Class 5 and 7 (Sandal and Sneaker) have a lot of intersection and they are both a type of shoe, also class 9 (Ankle Boot) has some intersection with them. Class 8 (Bag) has some intersection with class 9 (Ankle Boot), considering they both have similar width and height this makes sense. Class 1 and 3 (Trouser and Dress) have a lot of intersection and this makes sense since they both have greater height considering other classes. Class 2, 4 and 6 (Pullover, Coat and Shirt) have quite the intersection and these classes do look like each other. Class 0 (T-Shirt) also has an intersection with class 4, 6 and 3. It's possible that it is because dresses usually have short or no sleeves similar to t-shirts.

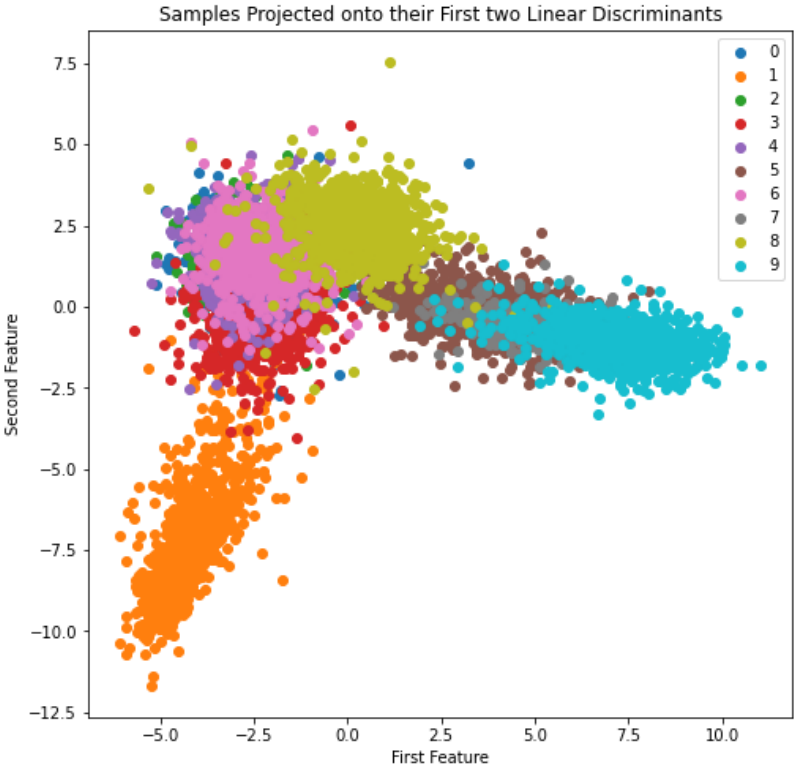
Adding the third principal component you can see that bags get discriminable from shoes to some degree and also t-shirts discriminate a bit from dresses as well.



b. Apply LDA on the dataset and project the samples onto their first and second linear discriminants and display the results.

I used sklearn.discriminant_analysis.LinearDiscriminantAnalysis() to compute the linear discriminants.





Fit_transform(x, y) will fit the data (x) based on their labels (y) and return the linear discriminants.

Only first linear discriminant is not a good idea and there is a lot of intersection.

With two linear discriminants, we can see that the shoe types discriminate from other classes by a great degree. Also it gets easy to discriminate Trousers from other classes. Bags have some intersection with the classes T-shirt, Shirt, Coat, Pullover that have taken the same area. Dresses have more intersection with these classes.

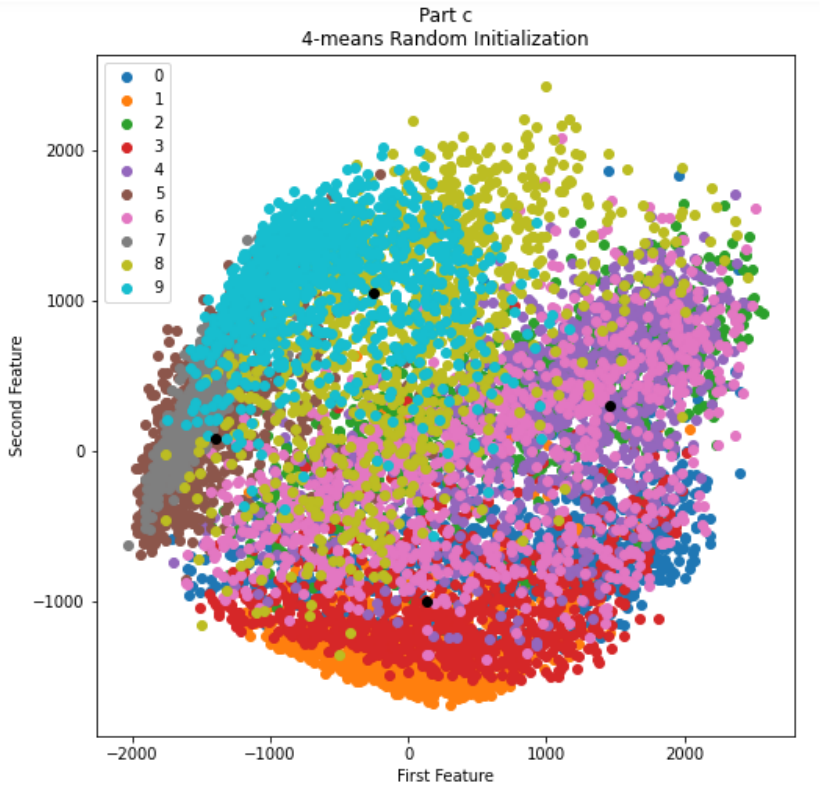
c. Now perform K-Means on the data using their first two principal components. Select the initial centroids randomly and set K = 4, 7, 10. Display the result of clustering with final centers highlighted. Also compare the results together and with the plot you obtained in part (a).

I did as it's told in the question using sklearn.cluster.KMeans().

You can see the results with black dots indicating the centroids in each plot, also the coordinates for each final centroids are given.

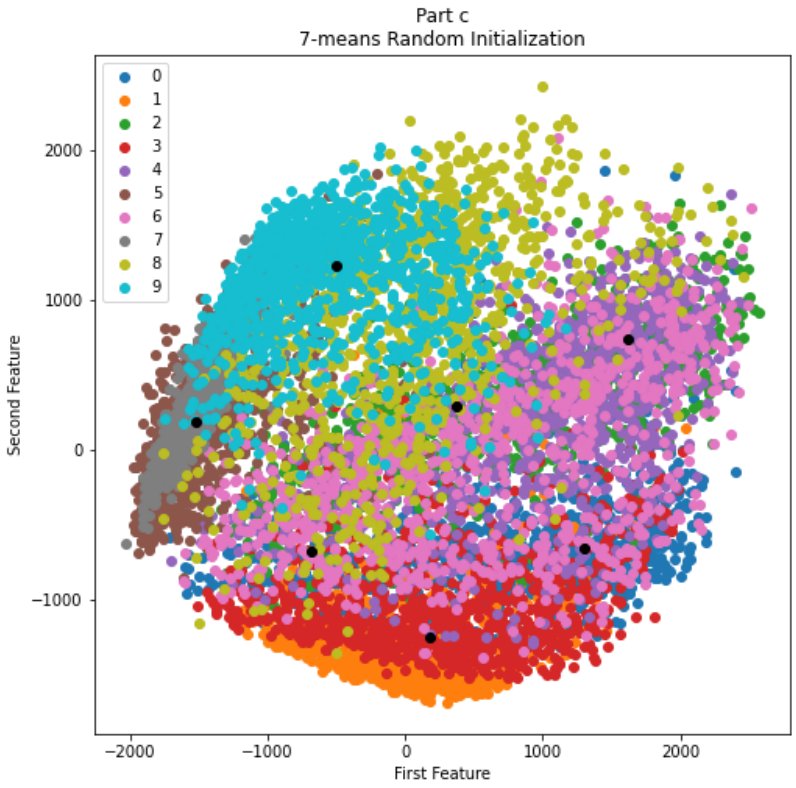
Four centroids is dividing the space to four equal parts and each part is containing variety of data points from different classes. From this it is obvious that four clusters is not enough to separate the classes properly. However based on the results in part a, one could say that the first cluster (containing the first centroid given in the picture below) mostly contains dresses, trousers and some t-shirts and shirts. The second cluster mostly containing coats, shirts and pullovers, the third one containing bags and ankle boots and the fourth one sandals and sneakers.

```
Final Centroids for random 4-means:
 [[  134.89089158 -1002.45416367]
 [ -250.02453903  1046.49700216]
 [ 1458.05372717   301.87191197]
 [-1395.45395185    83.85928286]]
```



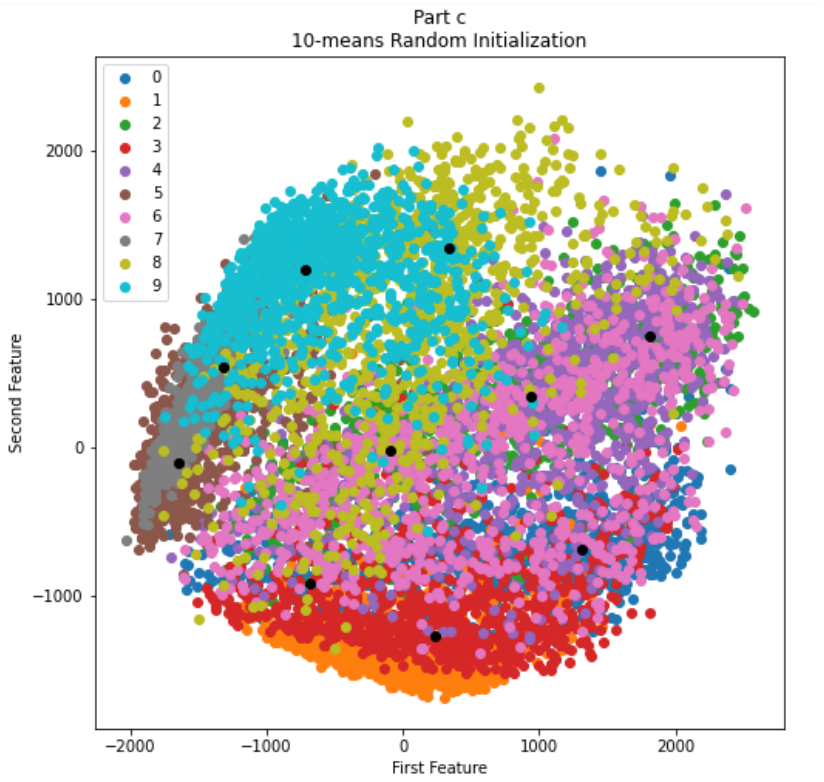
Seven clusters is not doing well in two dimensions. You can see that the second cluster contains data points from seven classes.

```
Final Centroids for random 7-means:
 [[ 1621.51993733   742.15909046]
 [ -678.57723797  -682.41016529]
 [ 1301.05671997  -656.94940971]
 [-1520.25380962   187.83302841]
 [  372.98100037   293.58203121]
 [  177.0006429  -1249.01886763]
 [ -499.07295698  1226.06927694]]
```



Ten clusters also don't do well in two dimensions; Some clusters contain a vast variety of data points from different classes.

```
Final Centroids for random 10-means:
 [[  938.34449154   343.27477143]
 [ 1808.71167986   752.74138922]
 [  335.01177095  1346.02951357]
 [ -679.13876521  -921.161046  ]
 [  232.1597652  -1270.6529627 ]
 [ 1317.06885859  -686.88202874]
 [-1643.22375736  -101.48286465]
 [  -94.96962262   -18.40305736]
 [ -718.7097315   1201.73696491]
 [-1319.33032579   536.75983914]]
```



d. Repeat part (c) considering the initial centroids as the mean of the samples such that for K=4, initial centroids are the mean of samples of classes {1,3,5,7}, {2,4}, {6,8,10} and {9}, for K=7, initial centroids are the mean of samples of classes {1,3,5}, {2}, {4}, {6}, {7}, {9} and {8,10}, and for K=10, initial centroids are the mean of sample of each of the 10 classes separately. Display and compare the results with those of the previous parts. Also comment on your observations.

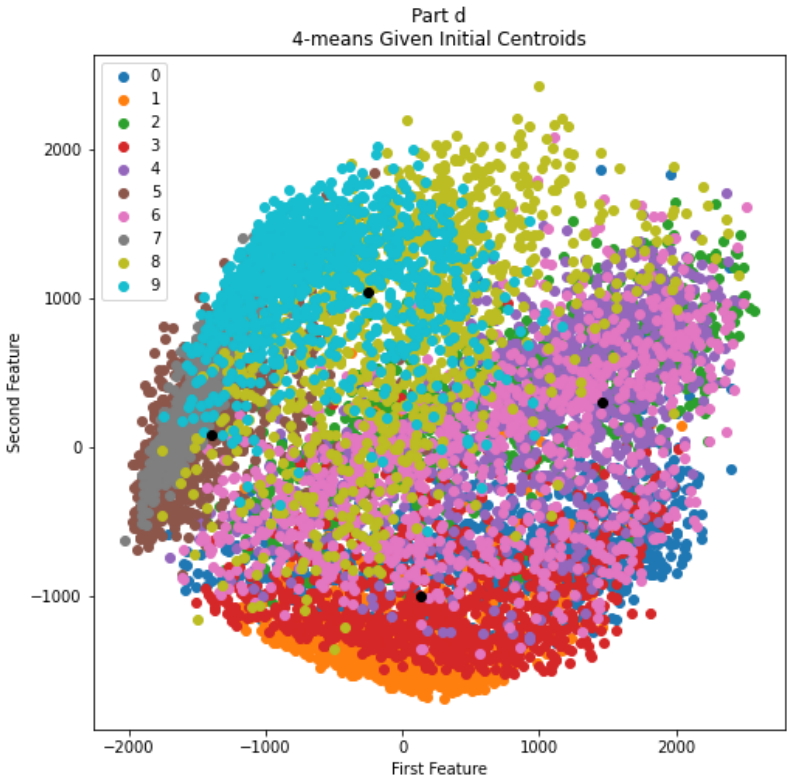
The Final Centroids are near the ones we found in the previous parts, only their order has changes due to the initial points not being random.

From the explanations I gave in part c, you can see how they were closely related to the given classes in this question.

The same thing happens with k=7 and k=10.

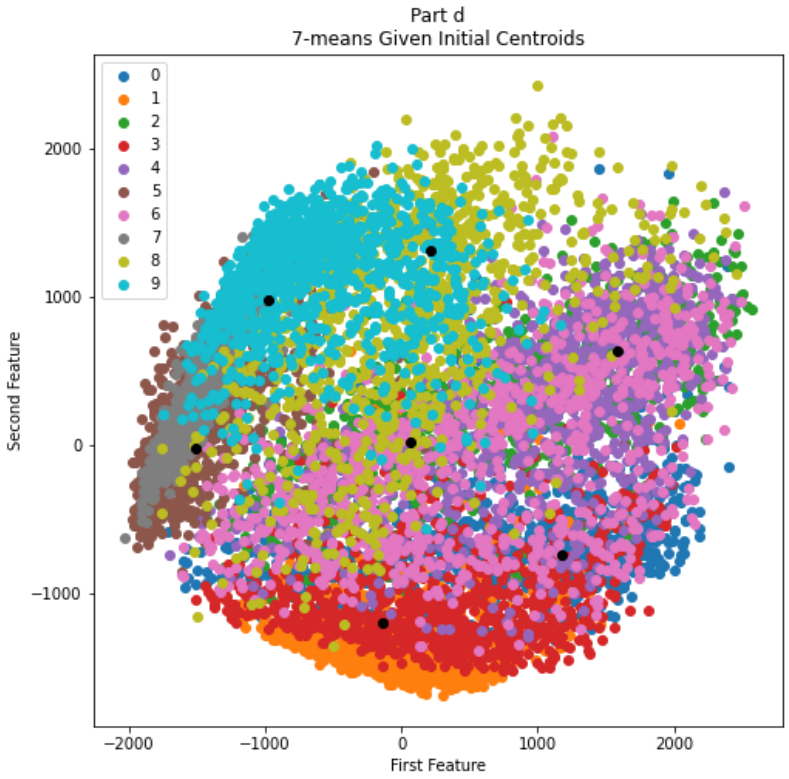
```
Initial Centroids for 4-means:
 [[  801.49215612   -16.48456837]
 [  157.38855841 -1186.8653773 ]
 [-1187.65526315   528.71048437]
 [   42.22004815   853.53757495]]

Final Centroids for 4-means:
 [[ 1458.85697254   300.44107859]
 [  133.46288895 -1003.05685467]
 [-1395.04249642    84.96844085]
 [ -247.62319327  1045.61305538]]
```



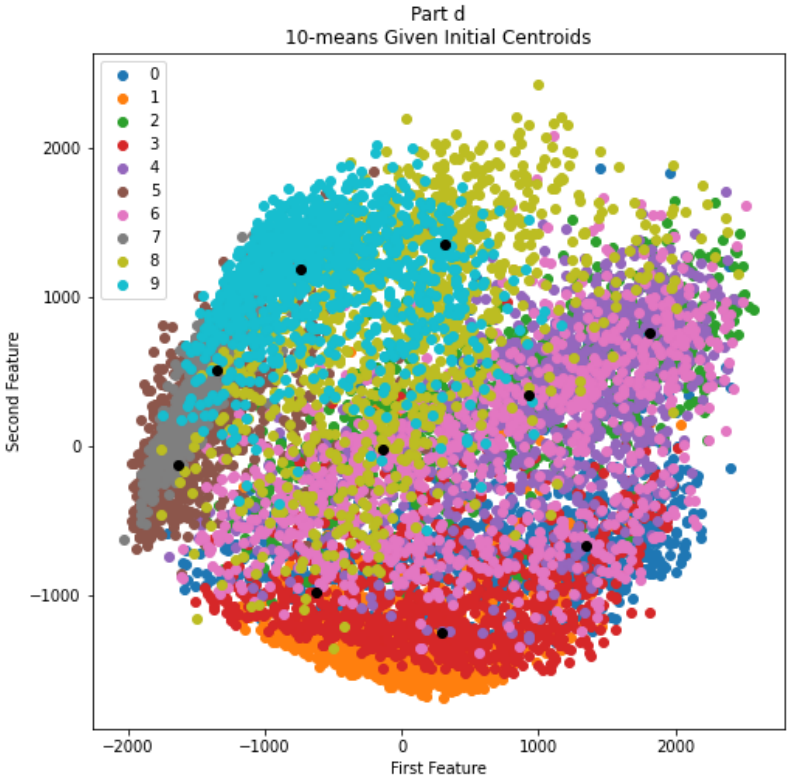
```
Initial Centroids for 7-means:
 [[  872.64428803    -5.07693251]
 [   21.99915562 -1351.07068695]
 [  292.7779612  -1022.66006765]
 [-1499.39952553    92.99348207]
 [  588.0357604    -50.70747594]
 [   42.22004815   853.53757495]
 [-1031.78313195   746.56898552]]

Final Centroids for 7-means:
 [[ 1590.3713748    636.48164431]
 [ -136.48494165 -1198.06429935]
 [ 1181.55604888  -741.67682985]
 [-1512.98409314   -26.15749697]
 [   61.57222292    20.27759411]
 [  214.09866664  1311.11353128]
 [ -977.47104932   976.24313391]]
```



```
Initial Centroids for 10-means:
 [[  710.05168291  -603.05272142]
 [   21.99915562 -1351.07068695]
 [  830.20959182   316.12315115]
 [  292.7779612  -1022.66006765]
 [ 1077.67158935   271.69877274]
 [-1499.39952553    92.99348207]
 [  588.0357604    -50.70747594]
 [-1481.01691638   395.74123708]
 [   42.22004815   853.53757495]
 [ -582.54934753  1097.39673396]]
```

```
Final Centroids for 10-means:
 [[ 1342.50479943  -667.437965  ]
 [ -629.30319819  -987.43054719]
 [  922.44496213   342.97558409]
 [  288.64363534 -1253.69550458]
 [ 1807.02429505   758.31989015]
 [-1641.6457468    -129.96649807]
 [ -139.87026762   -26.68341956]
 [-1349.14072533   505.28789419]
 [  309.2771595   1349.82987375]
 [ -745.97326124  1181.93559655]]
```



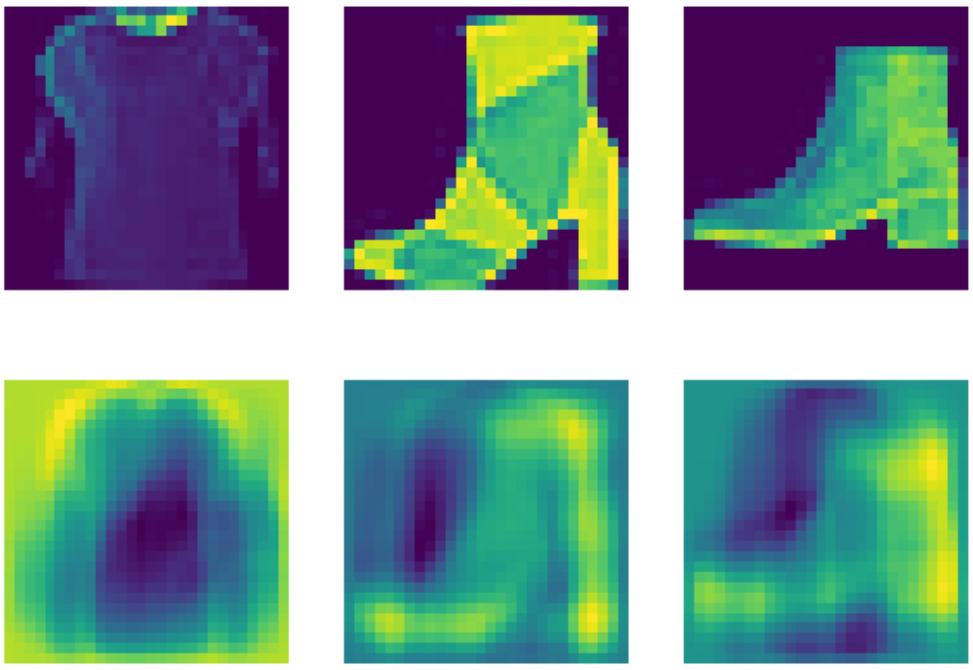
e. As can be seen, clustering the data using their first two principal components doesn't produce satisfactory results. We are going to increase the clustering accuracy by considering more principal components. By trial and error find the number of principal components which is enough to capture 0.95 of the data variance. Reconstruct three arbitrary samples using these principal components and compare the results with their corresponding original images.

```
s=0
k=0
while s<=0.95:
    s+=pca.explained_variance_ratio_[k]
    k+=1
print('for k= ',k ,'we have: ', s)
```

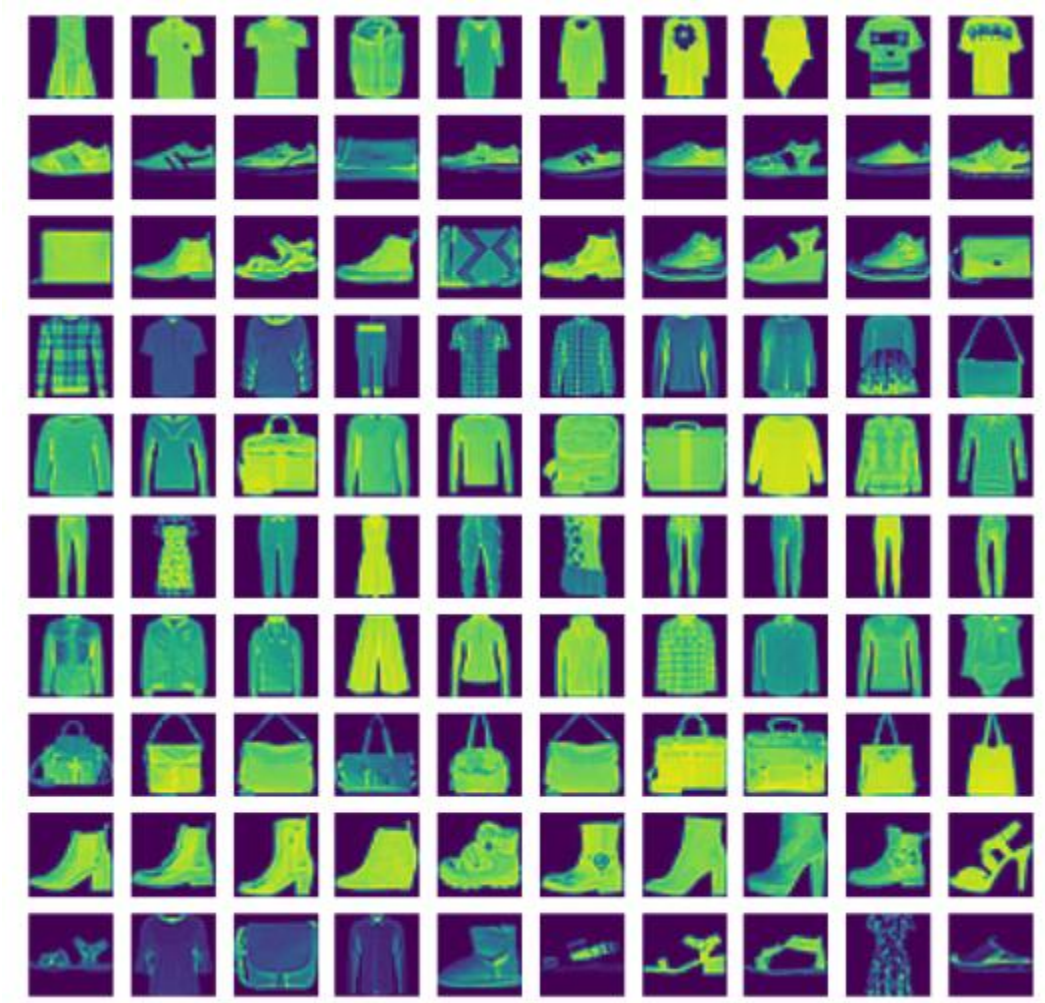
```
for k=  183 we have:  0.9501609243274309
```

With k=183 PCs can preserve slightly more than 0.95 of the data variance.

In order to reconstruct the samples I calculated the dot product of PCs and the corresponding eigenvectors. Then I reshaped the matrix into a 28*28 picture. The samples on the bottom show the reconstructions of the pictures on the top.



f. Use K-Means to divide data into 10 clusters. Set the required parameters appropriately. Display 10 samples of each cluster randomly (100 in total) and comment on the results.



It seems cluster 0 contains data mostly from classes 1 and 4 (T-Shirt/Top and Dress),

Cluster 1 from class 8 (Sneaker),

Cluster 2 from classes 9 and 10 (Bag and Ankle Boot),

Cluster 3 from classes 1, 3 and 7 (T-Shirt/Top, Pullover and Shirt),

Cluster 4 from classes 3, 5, 7 and 9 (Pullover, Coat, Shirt and Bag),

Cluster 5 from classes 2 and 4 (Trouser and Dress),

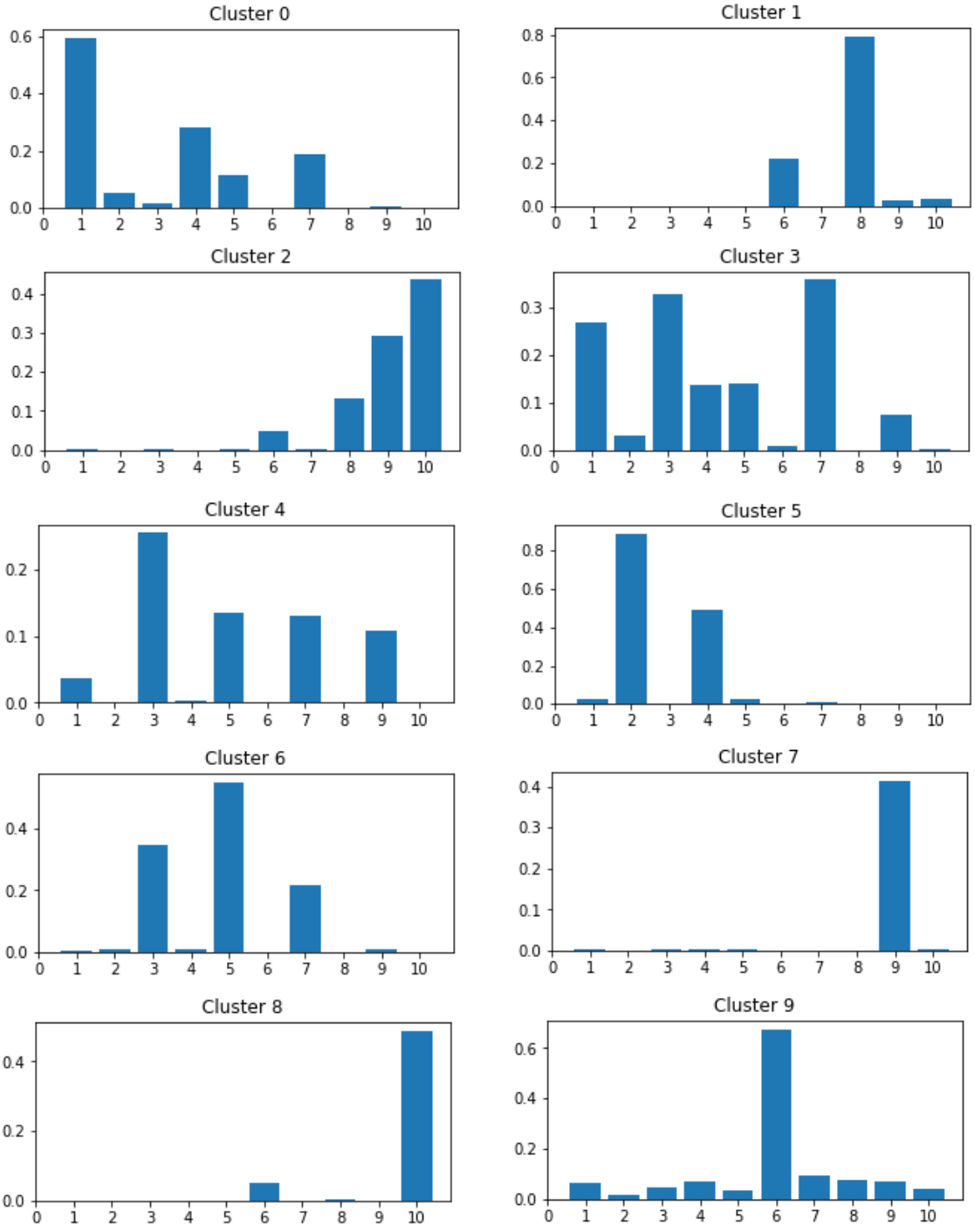
Cluster 6 from classes 3, 5 and 7 (Pullover, Coat and Shirt),

Cluster 7 from class 9 (Bag),

Cluster 8 from class 10 (Ankle Boot),

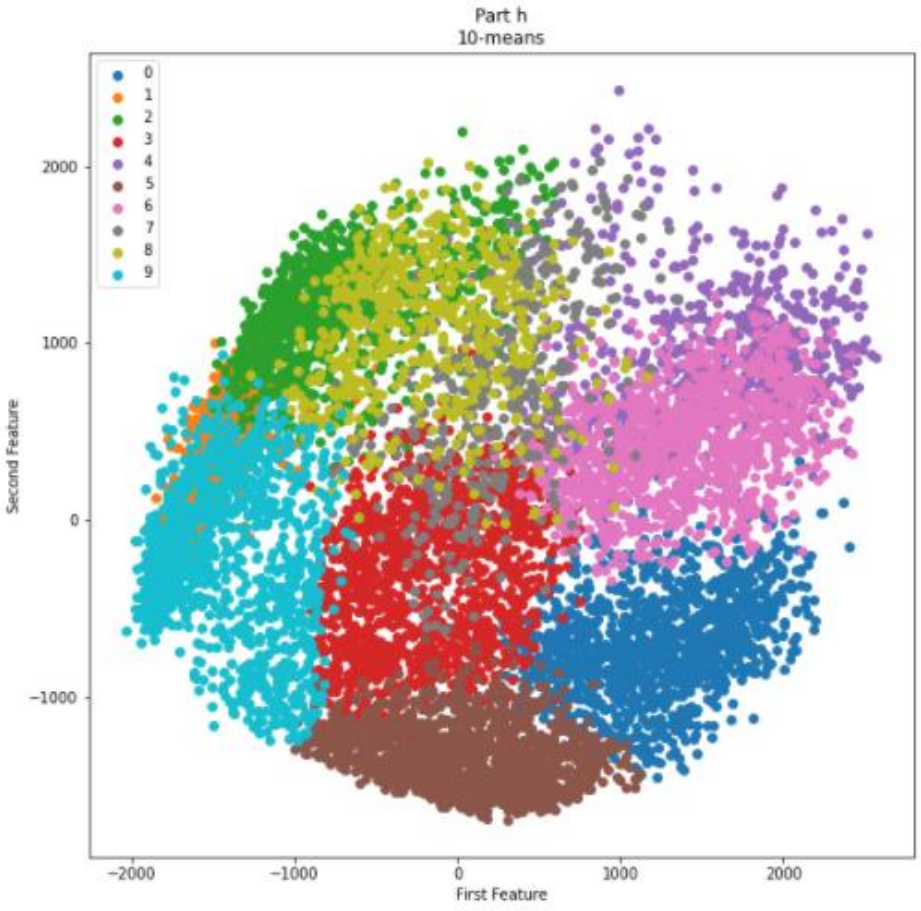
Cluster 9 from classes 6 (Sandal).

g. Draw 10 bar graphs for each cluster, each representing the distribution of the samples of different classes (in percentage). Compare different clusters using these bar graphs.



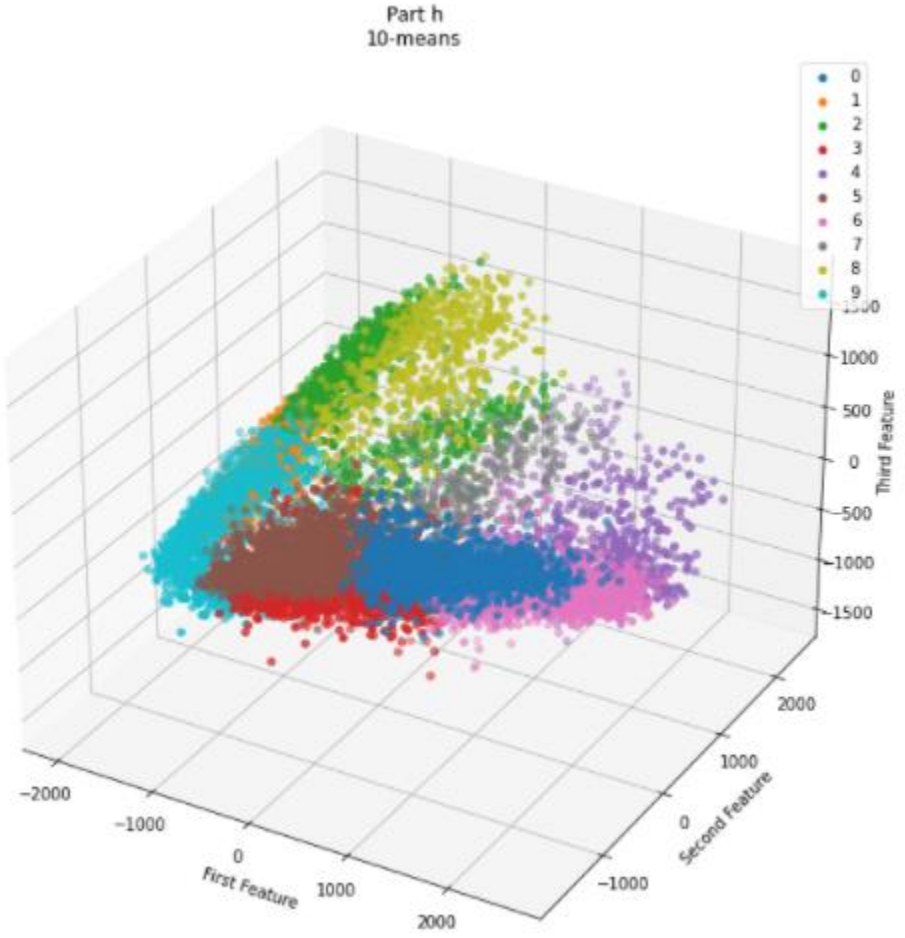
The bar graphs comply greatly with the results from the previous part. Also they show the purity of each cluster. Cluster 7 and 8 are purer than the others while clusters 3 and 4 are a group of several classes.

h. Finally, visualize the clustering by keeping only 2 and 3 features and displaying the corresponding scatter plots.



Of course in higher dimensions it would be much more obvious that these clusters are not sharing intersections but even in this 2-d plot one can see that 10-means has created blob-like and compact clusters and considering the plot from part a, makes sense that how some clusters are containing data points from different classes; for example cluster 6 in pink would contain data from classes Pullover, Coat and Shirt. Also it's understandable that class ankle boot and bag have discriminated.

These explanations are more obvious in the 3-d plot.



8. Some Explanatory Questions

Please answer the following questions as clear as possible:

a. Why doesn't it sound reasonable if the number of PCs is greater than dimensions?

Since we are using eigenvalues and eigenvectors and the number of eigenvectors is equal to the matrix rank.

Also if the number of PCs was greater than dimensions, we would be able to take the data to higher dimensions while we want dimension reduction.

b. Explain how PCA can be applied to the problem of image compression.

PCA identifies the k vectors that maximize the variance of the original image matrix. Each of these vectors is just a linear combination of the original pixels. So when we use PCA to identify a basis of k vectors that best represent the images, each image will simply be a linear combination of these basis vectors. So we can represent the image in just k dimensions, rather than the original d.

c. Is it possible to make a kernel version of K-Means clustering? If yes, how? And what would be an advantage of kernel-based K-Means? If no, why?

Yes. Same as we had in SVM, before clustering, points are mapped to a higher-dimensional feature space using a nonlinear function, and then kernel k-means partitions the points by linear separators in the new space. K-means cannot separate clusters that are non-linearly separable in input space, however by kernel k-means it is possible to separate this type of clusters.

d. In K-Means clustering algorithm, the goal is to minimize the variance of the solution. In general, how does the variance of a partition change as the number of clusters (K) is increased? Justify your answer.

The variance of a partition decreases, since the number of points in each partition would become smaller and also they would become closer to their mean. When k=n (number of points) variance within a partition becomes zero, since the only point in a cluster would be the centroid.

e. What value of the parameter K in K-Means can always lead to a variance of zero, and why?

When k=n (number of points) variance within a partition becomes zero, since the only point in a cluster would be the centroid. The subtraction in $\frac{1}{n}\sum_{1}^{K}\sum_{x_i \in C_K} \| \mathrm{xi} - \bar{\mathrm{x}}\mathrm{k} \|^2$ would be zero.