CS6140 Project 2

Yihan Xu
NUID: 001566238
Jake Van Meter
NUID: 002965845

Overview:

In this project, we implemented the K-Nearest-Neighbors **algorithm** and tested it with a **USA_Housing.csv** dataset which contains 5000 rows. We converted the numeric dependent variables into several categories and experimented with different approaches. And we also experimented several different **cluster** algorithms with sklearn, analyzed the cluster quality by computing representation error and several cluster quality metrics, applied kmeans algorithm on a **wine-clustering.csv** dataset. Additionally, we used **K-Nearest Neighbor** and **PCA** to classify activity from phone measurements.

For the **extensions**, we have covered the following:

- Created the confusion matrix to evaluate the KNN classifier.
- Tried 2 additional clustering algorithms other than k-means: hierarchical clustering and mean-shift clustering.
- Implemented 2 cluster quality metrics: Rissannon Minimum Description Length, Krzanowski and Lai.
- Implemented the nearest neighbor function such that it returns the error terms for each point.
- Applied k-means to projected data with weighted eigenvectors
- Did more exploration in final task by trying different number of dimensions: 28 and 11 which explained 80% and 70% of variance respectively

Task 1

We used **USA_Housing.csv** as our dataset, and it is in the dataset folder. It has 5000 rows and contains 5 different independent variables and 1 numeric dependent variable. Since the dependent variable is numeric, we plotted the distribution of the values in the training set and partitioned the value range and divided it into 4 categories. We have tried different ranges and picked the one that is most reasonable.

1.A

Include an equation or algorithm in your report that describes your distance function.

The equation we used to calculate our distance metric was a Normalized Euclidian Distance equation of the form:

$$D = \sqrt{\sum \frac{(x_i - y_i)^2}{\sigma_i}}$$

Where,

- D is the distance,
- x, y are two samples from the data set,
- i is the column,
- σ_i is the standard deviation of column i.

This algorithm is implemented in code by the `dist_between_data_sets` function in `nn.py`. This function takes in the two data sets in the form of pandas DataFrames and returns an ExN distance matrix where each of the N columns is the distance from a row in the first matrix to each of the E rows in the second matrix.

1.B

Using the function you wrote for task A, implement a Nearest Neighbor classification function.

The function `k_nearest_neighbors` function in `nn.py` utilizes the `dist_between_data_sets` to classify samples based on the dependent variable labels of their k nearest neighbors, and uses majority voting to find the predicted category. This function works for any value of k, and therefore is functionally a standard nearest neighbor classifier when the given k is equal to 1.

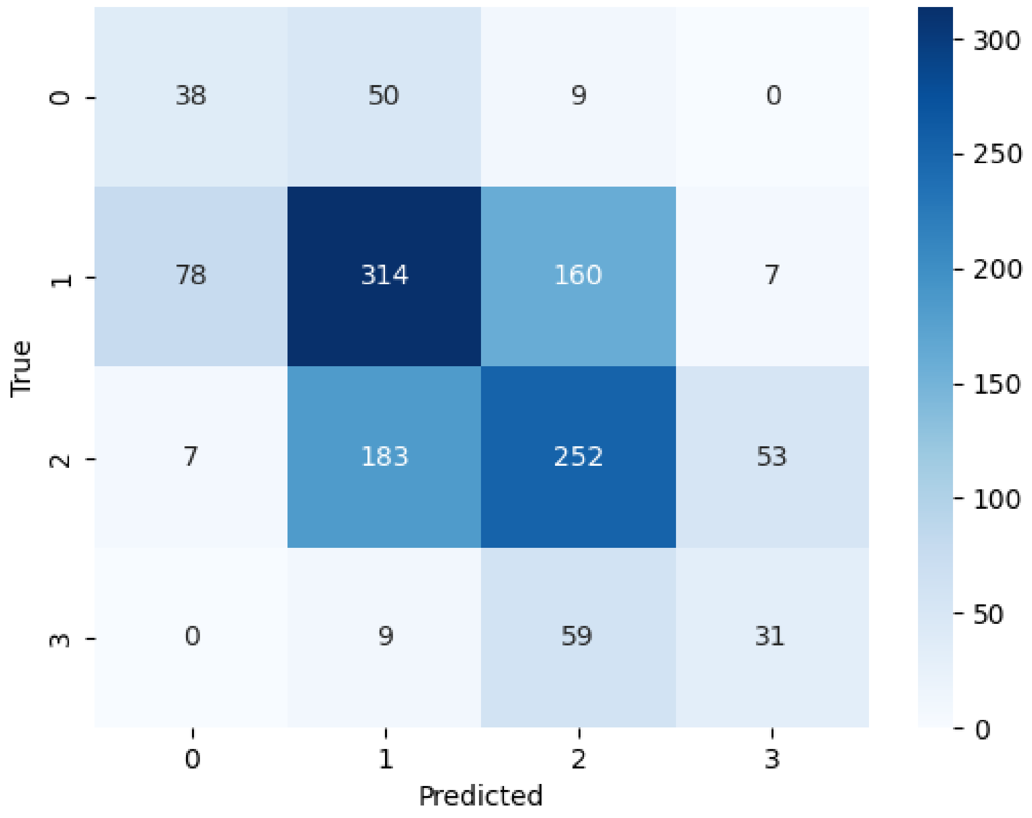
1.C

Compute the simple accuracy of your classifier: what percentage of the test data was correctly classified?

Precision of the classifier with one nearest neighbor (unwhitened data):

```
precision using un_whitened data is: 0.508
```

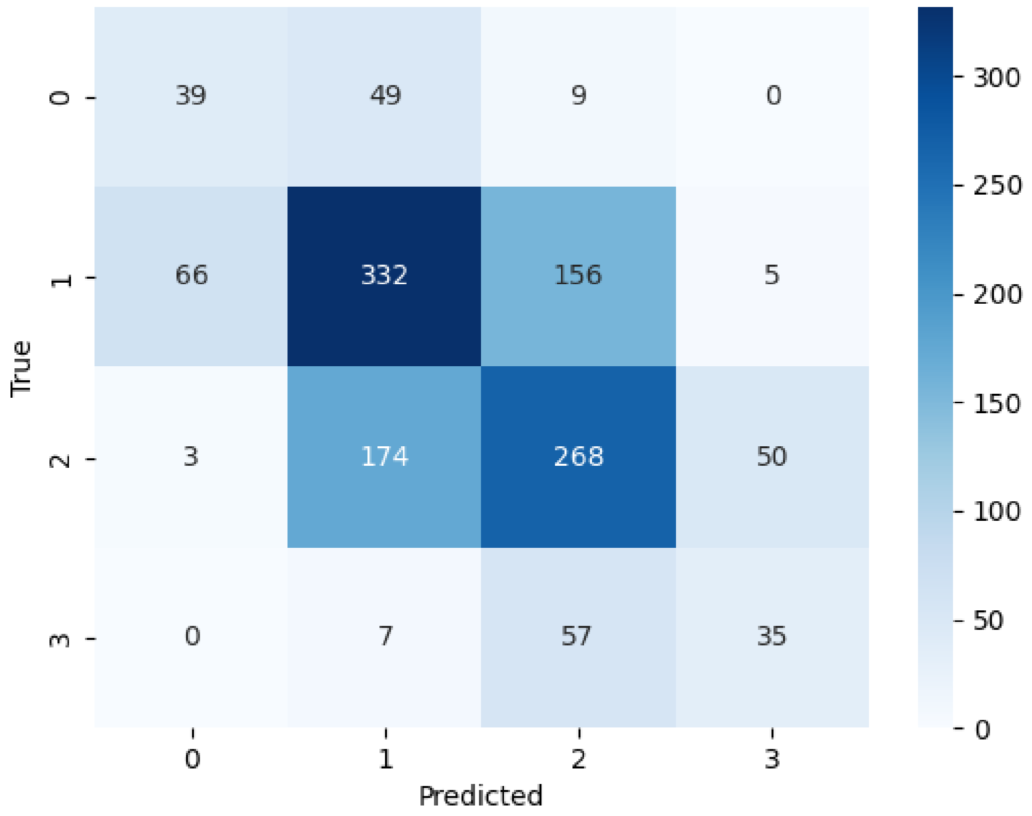
Confusion matrix of the classifier with one nearest neighbor (unwhitened data):



Precision of the classifier with one nearest neighbor (whitened data):

```
precision using whitened data is: 0.5392
```

Confusion matrix of the classifier with one nearest neighbor (whitened data):



$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 39 & 49 & 9 & 0 \\ 1 & 66 & 332 & 156 & 5 \\ 2 & 3 & 174 & 268 & 50 \\ 3 & 0 & 7 & 57 & 35 \end{array}$$

According to the result, the precision increases 3% by normalizing the data, which means the normalization helps with the precision.

The overall precision is around 50%, with 4 different categories, 50% is not good enough, this might be due to the dataset is not categorical and is more of a regression dataset. We will try different modifications in 1.D to see if there’s any improvements.

According to the confusion matrix, most of the predicted values are 1 and 2, and 30% of them are wrong. The offset of the result is not large, the larger difference of the true and predicted, the less is the number. The true value and predicted value are pretty close.

1.D

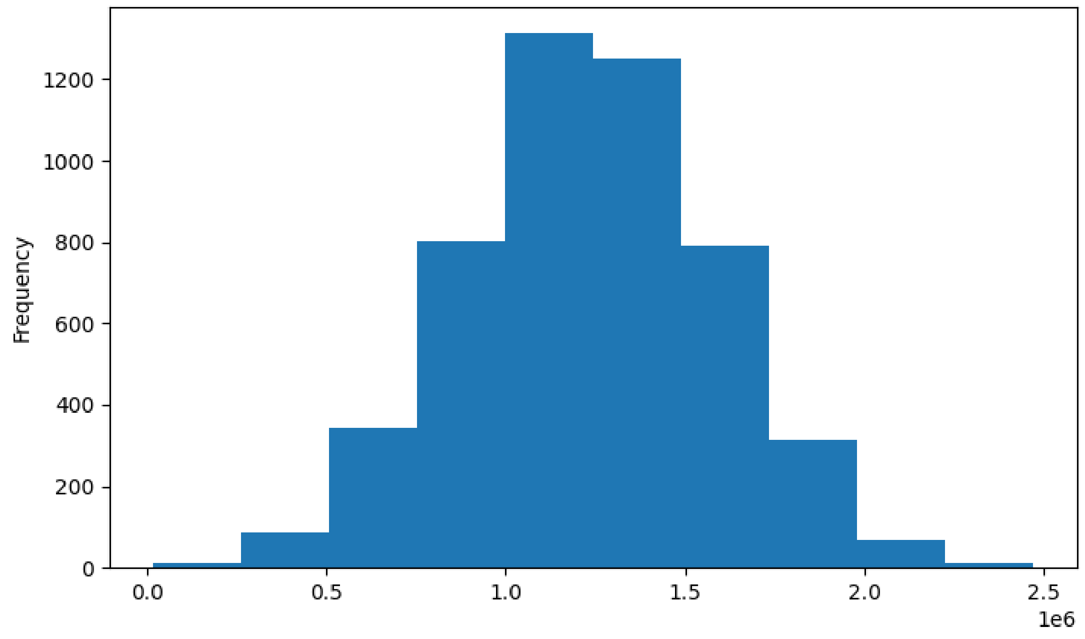
Include in your report a description of your experimentation and the results for at least one alternate version of your classifier. Discuss how your modification affected the accuracy or speed of your classifier.

To improve the precision and optimize the classifier, we have tried the following:

- Tried different number of nearest neighbors
- Tried different methods of classifying the dependent variable
- Tried different methods of distance metrics (whitened or unwhitened)

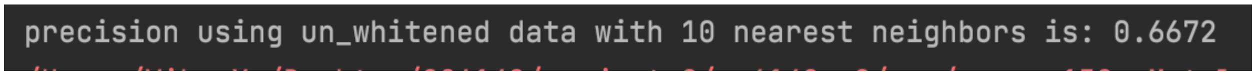
We have tried different partitions of the dependent variable (price), and the larger the range is, the larger the precision is. But with a larger range, the less information is provided to us. For example, if the price range is from 1 to 10 million, most of the houses will fall into this range, but this does not provide us with any useful information. After looking at the price histogram and tried different partitions, we have divided the price into 4 ranges based on a histogram distribution:

- < 750,000
- 750,000 ~ 1250,000
- 1250,000 ~ 1750,000
- > 1750,000



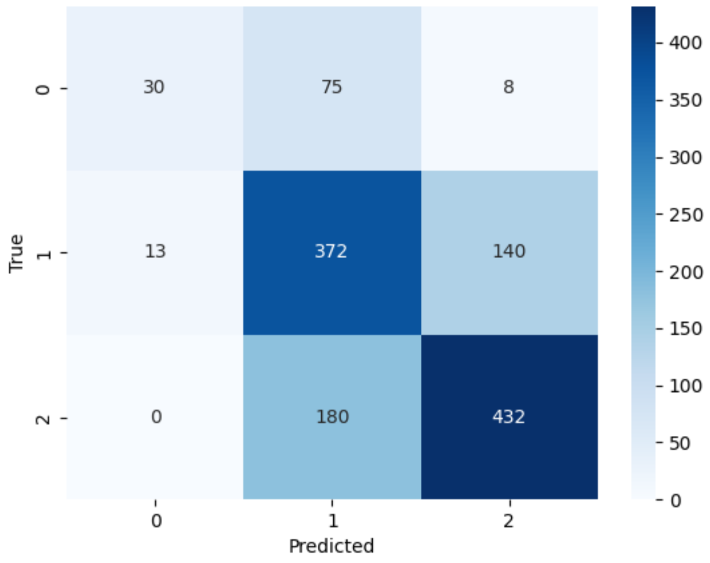
We have also tried different k in KNN = 10 and KNN = 20 as shown by the following images.

Precision of the classifier with 10 nearest neighbors (unwhitened data):

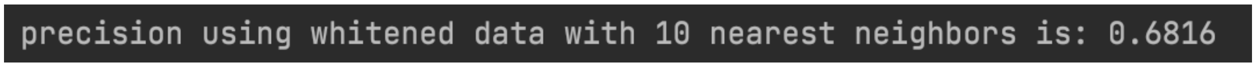


```
precision using un_whitened data with 10 nearest neighbors is: 0.6672
```

Confusion matrix of the classifier with 10 nearest neighbors (unwhitened data):

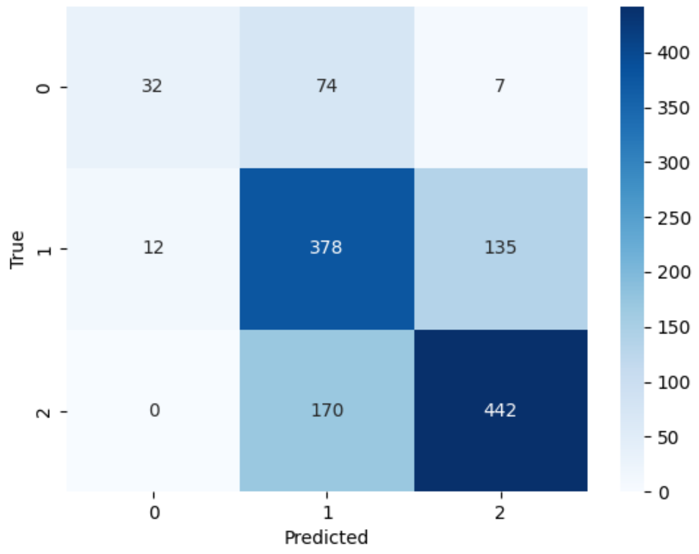


Precision of the classifier with 10 nearest neighbors (whitened data):

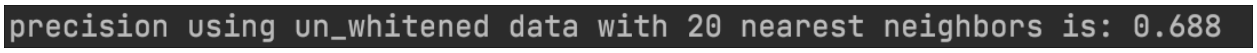


```
precision using whitened data with 10 nearest neighbors is: 0.6816
```

Confusion matrix of the classifier with 10 nearest neighbors (whitened data):

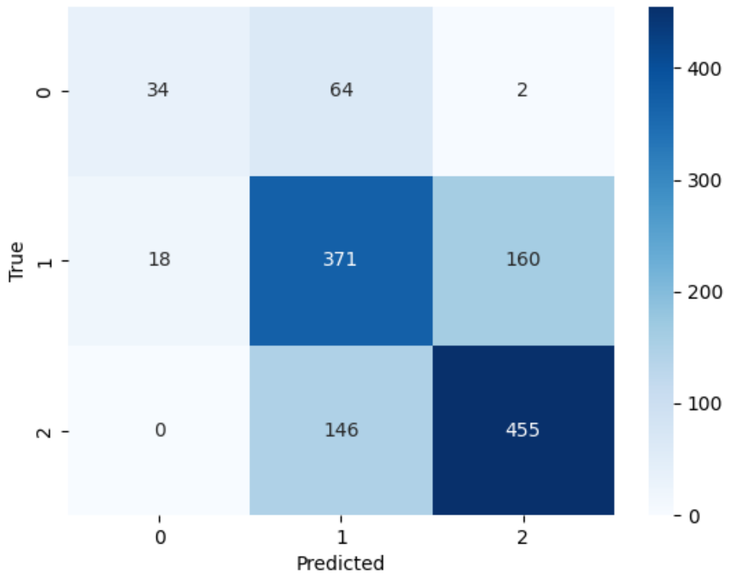


Precision of the classifier with 20 nearest neighbors (unwhitened data):

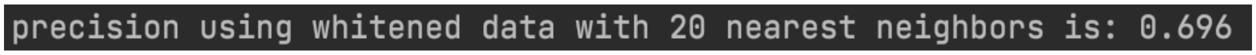


```
precision using un_whitened data with 20 nearest neighbors is: 0.688
```

Confusion matrix of the classifier with 20 nearest neighbors (unwhitened data):

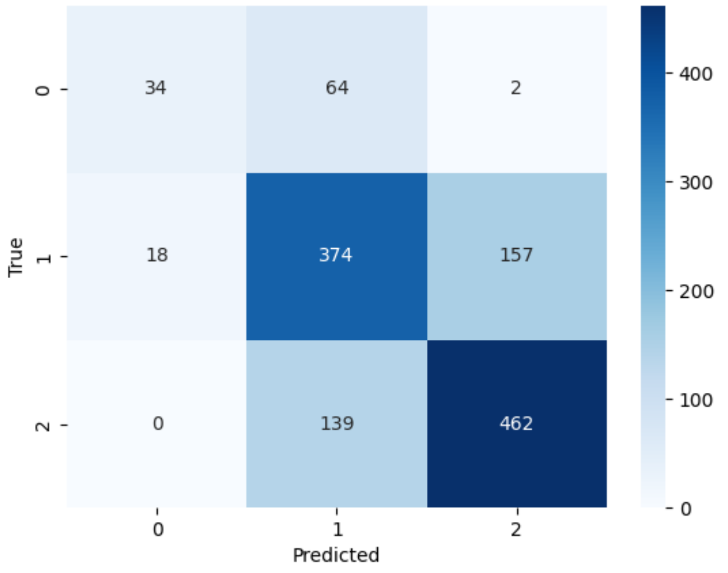


Precision of the classifier with 20 nearest neighbors (whitened data):



```
precision using whitened data with 20 nearest neighbors is: 0.696
```

Confusion matrix of the classifier with 20 nearest neighbors (whitened data):



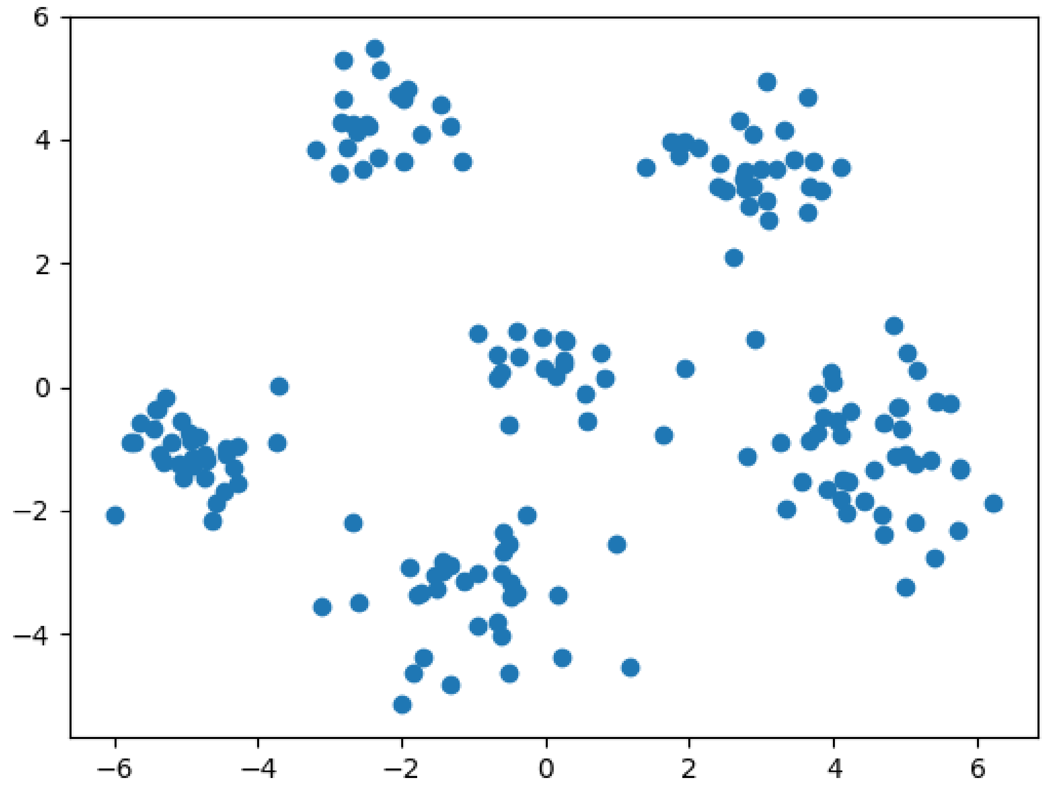
According to the result, the precision gets 15% higher when k is improved from 1 to 10, and when k is improved from 10 to 20, the improvement on precision is trivial (only 2%), which means 10 is a good enough number of nearest neighbors for this dataset.
And the precision gets 1% to 2% higher when the data is normalized, this means by normalizing the data, we can get slightly more precise results on this dataset, but it does not help much.

Task 2

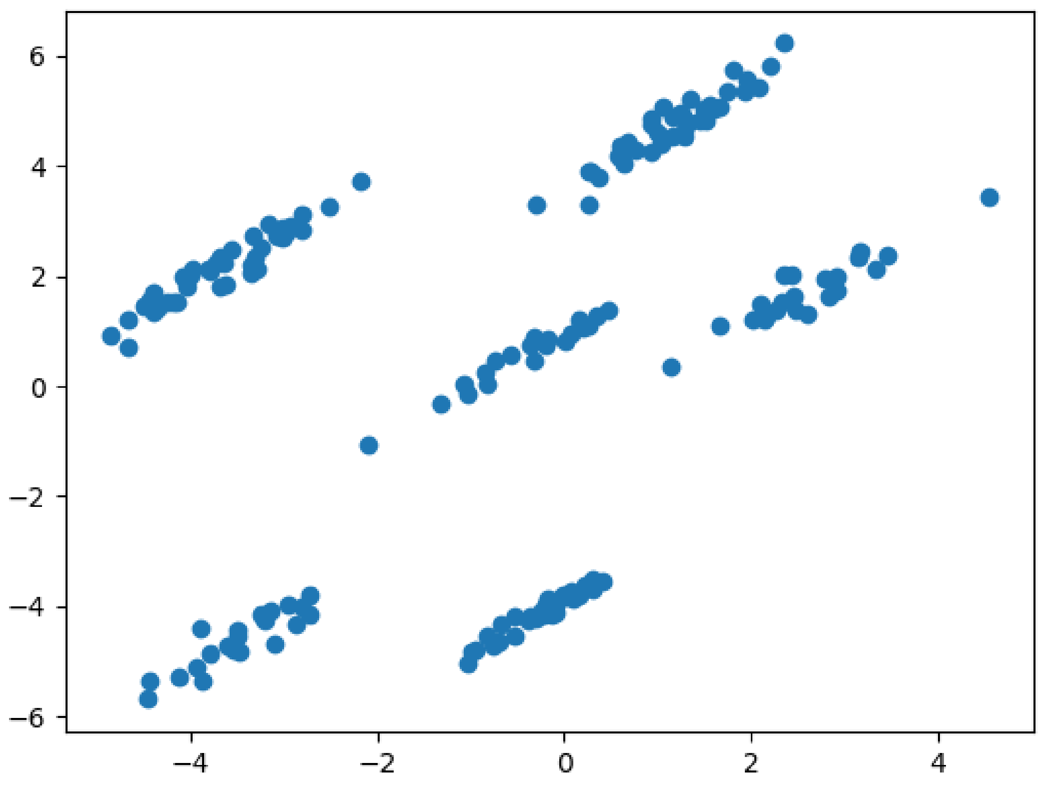
2.A

Plot the data and note the structure of the data. How many natural clusters are in each data set? What is the shape of the clusters?

Plot of dataset A:



Plot of dataset B:

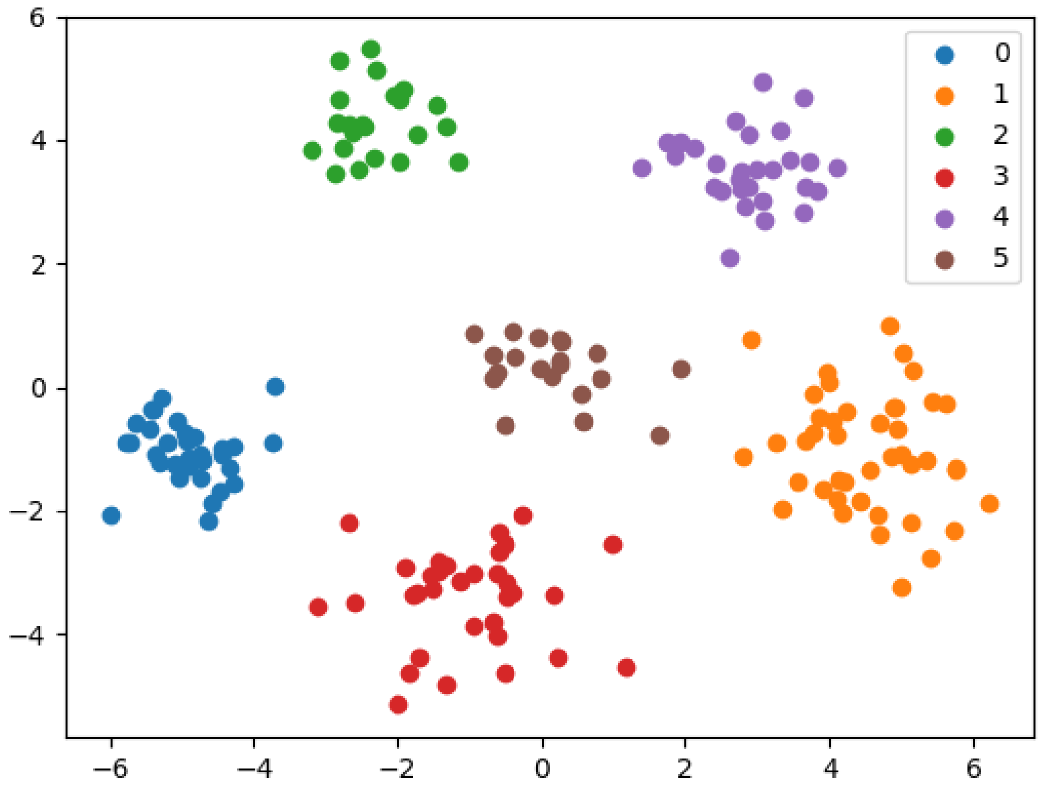


By looking at the plot, there are 6 natural clusters in both datasets, the shape of the clusters in dataset A is more spherical, and the shape of the clusters in dataset B is more linear.

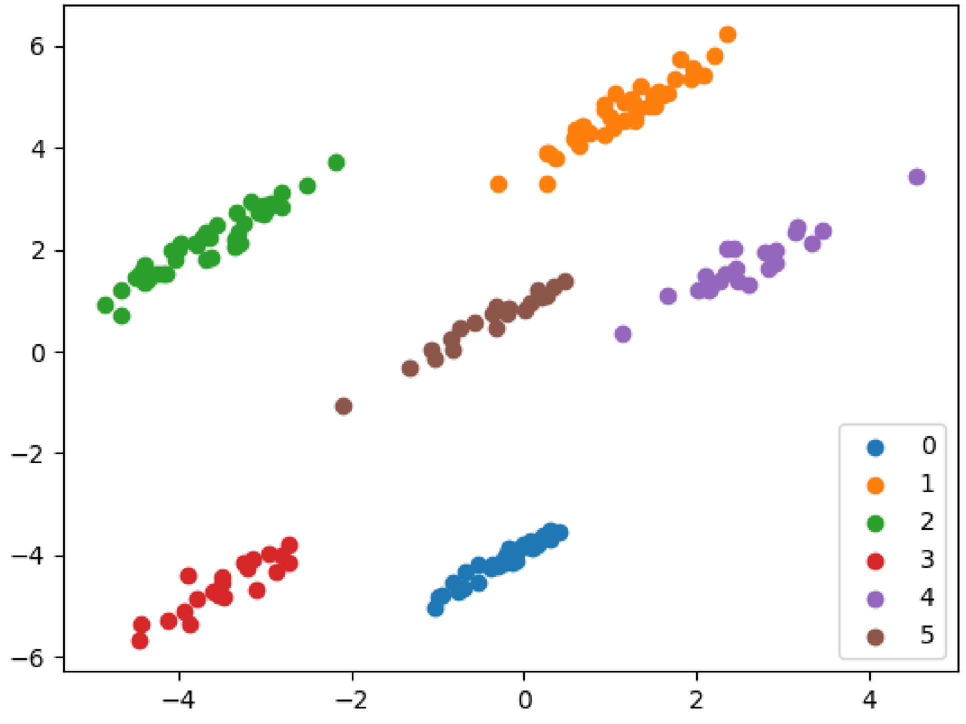
2.B

Generate a plot for each data set using K=6, using color to indicate the cluster ID for each point. Pick a second clustering method implemented by sklearn and show the comparative results. Are they different? Do the differences make sense given the differences in the clustering algorithms?

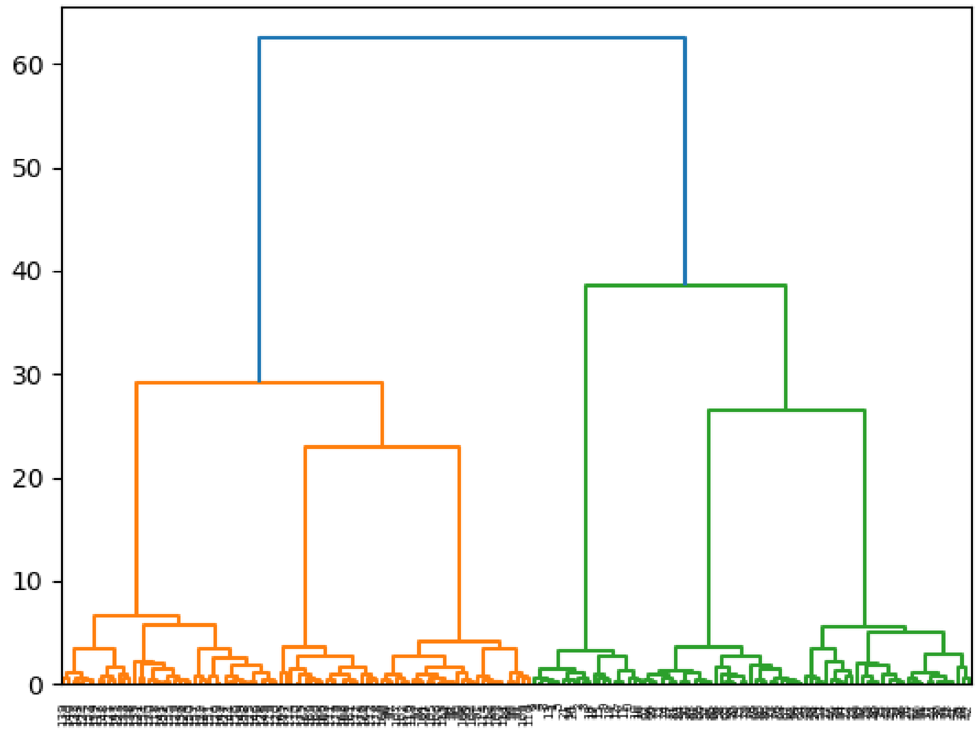
K means clustering for dataset A with k=6:



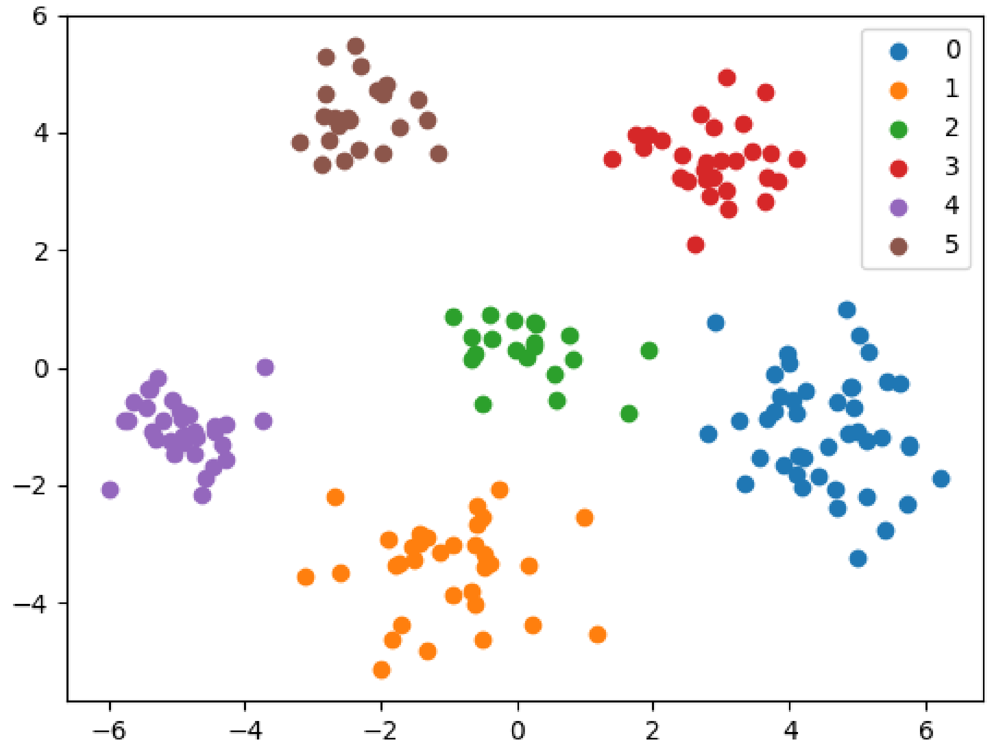
K means clustering for dataset B with k=6:



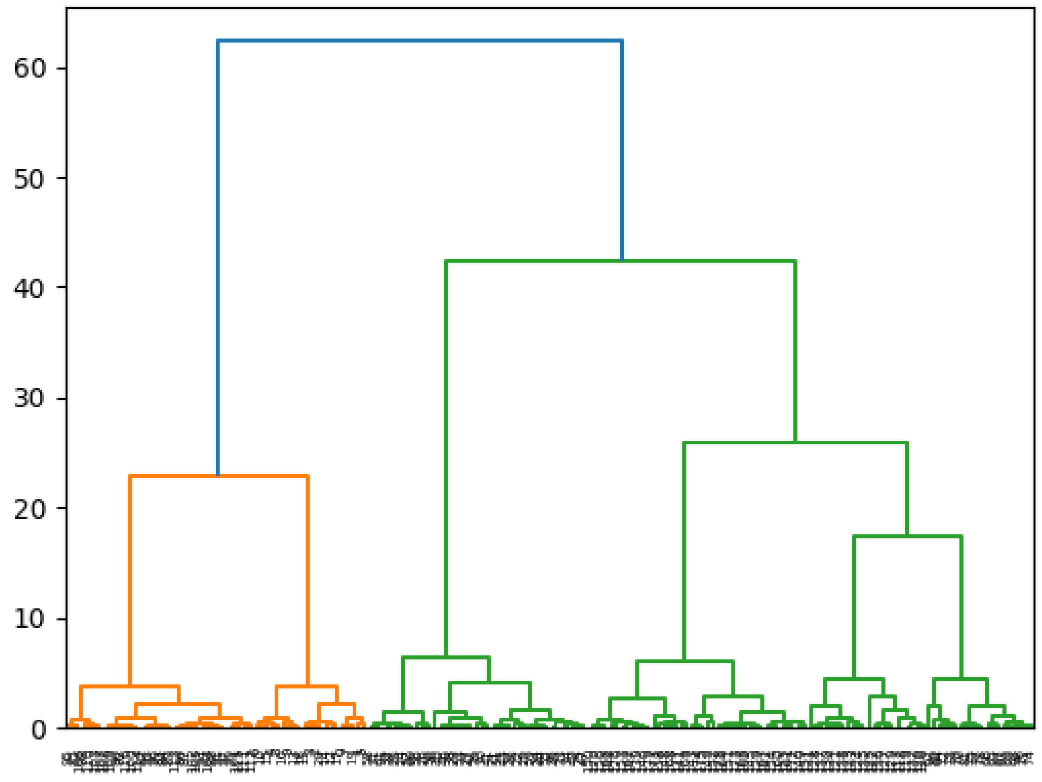
Data hierarchy for dataset A:



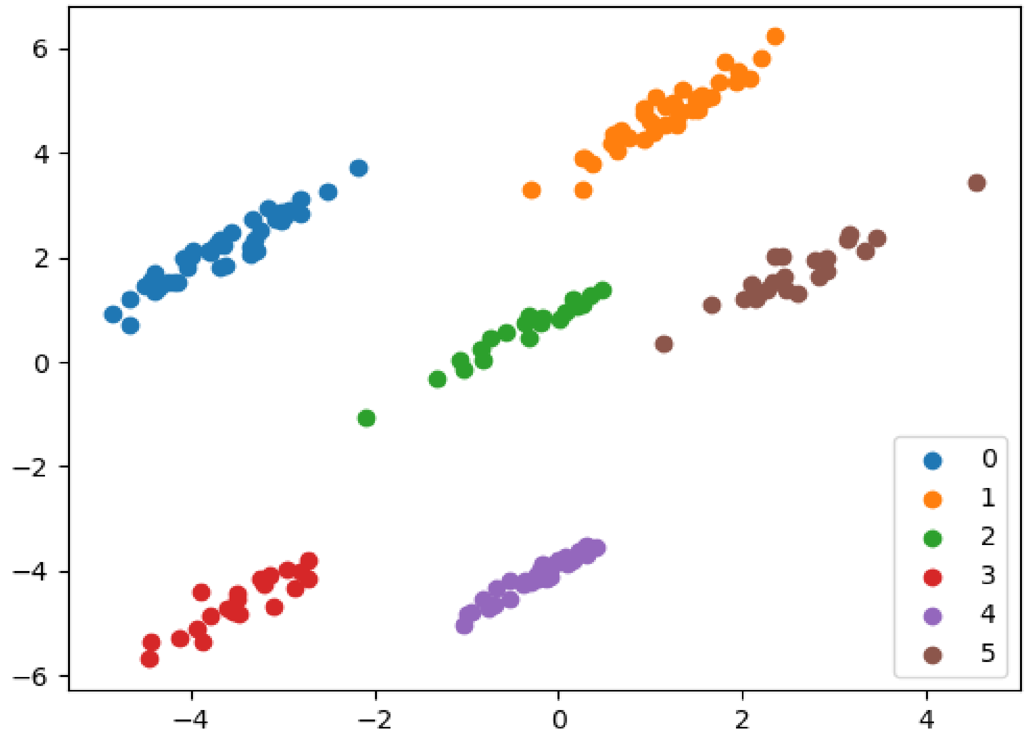
Hierarchical clustering for dataset A with n-clusters=6:



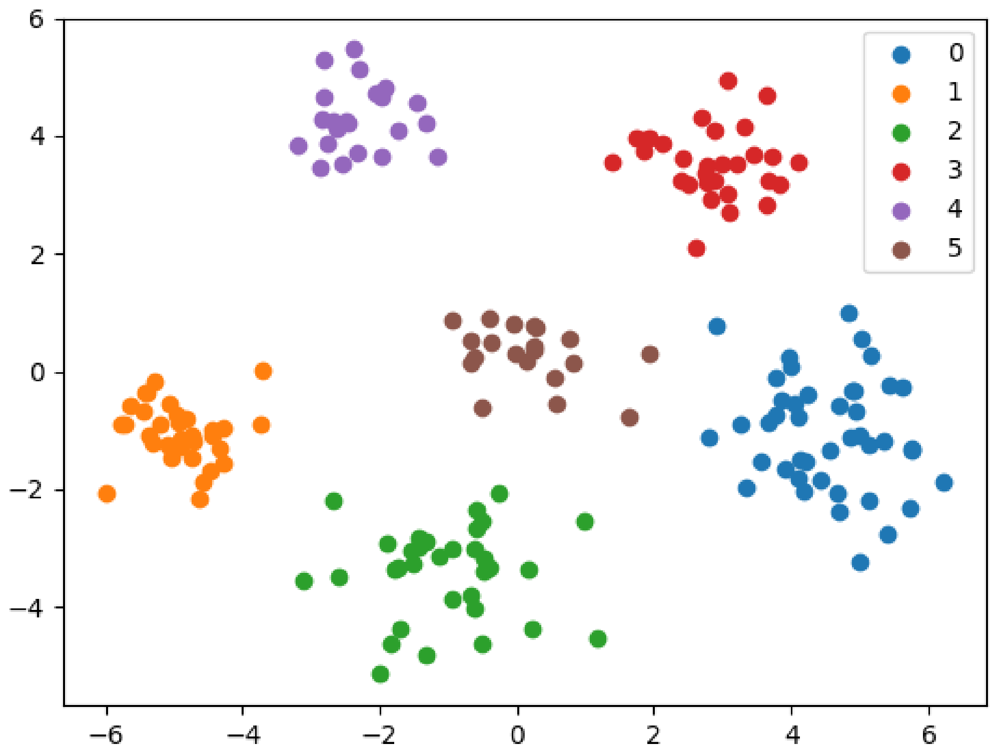
Data hierarchy for dataset B:



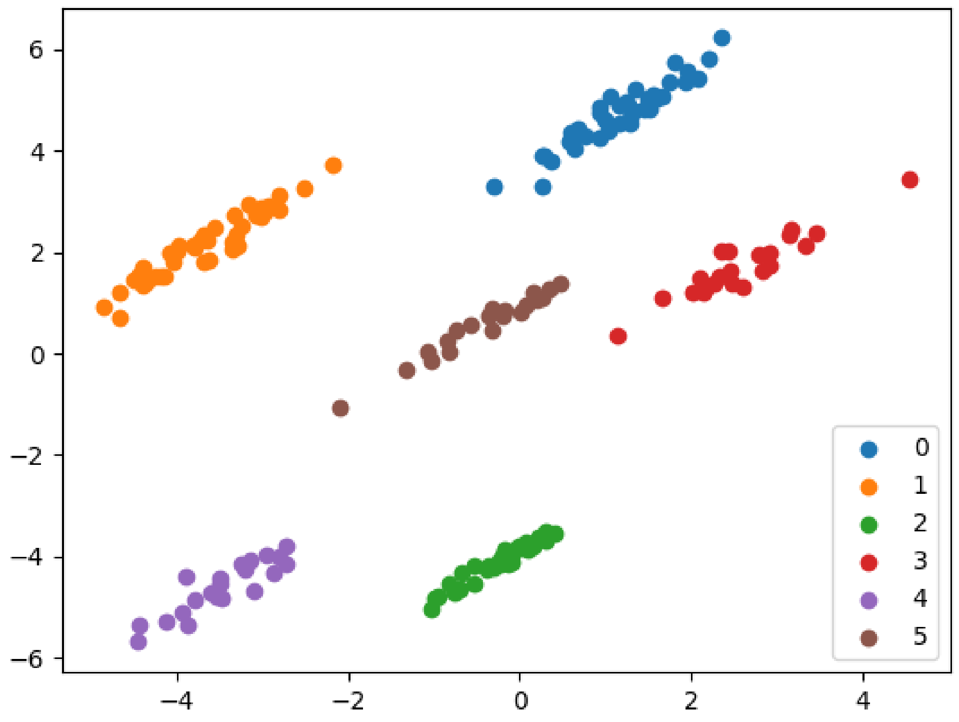
Hierarchical clustering for dataset B with n-clusters=6:



Mean-shift clustering for dataset A:



Mean-shift clustering for dataset B:

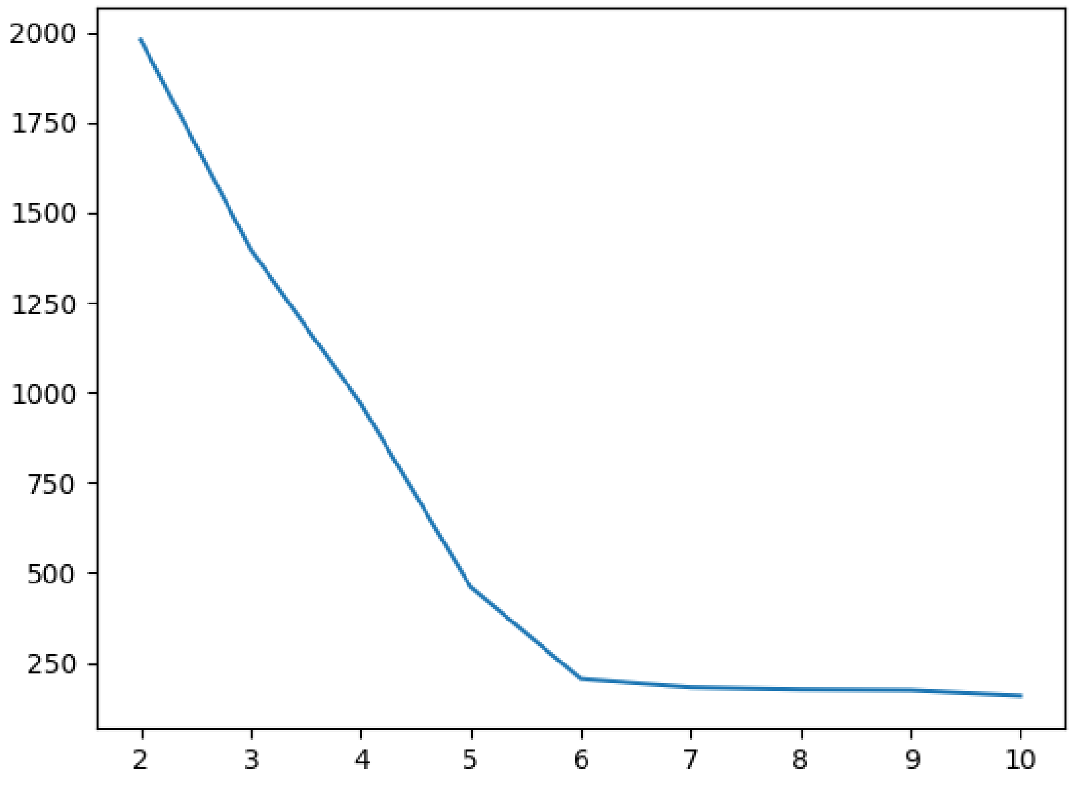


According to the cluster plot, the result are virtually the same using k-means with k=6, hierarchical clustering with n-clusters=6, and mean-shift with bandwidth=2. This is because there are obvious natural clusters in each dataset.
By using k-means clustering, we are iterating several times to find the stable k centroids, and by using hierarchical clustering, we are ranking the data points by their distance, and then cut the hierarchy where there are 6 clusters. By using mean-shift clustering, we shift the cluster mean each time, and since there are 6 natural clusters, we end up having the same cluster as the other algorithms.

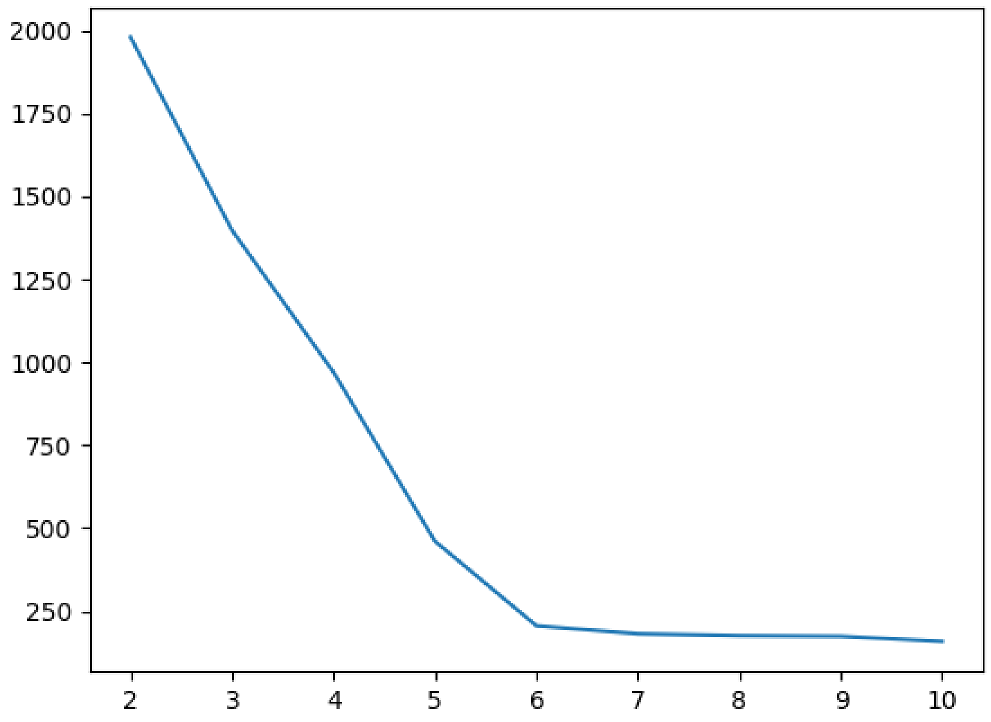
2.C

Apply K-means clustering on data set A using K from 2 to 10. Plot the representation error for each value of K. Include in your report a plot of the representation error for data set A and one cluster quality metric.

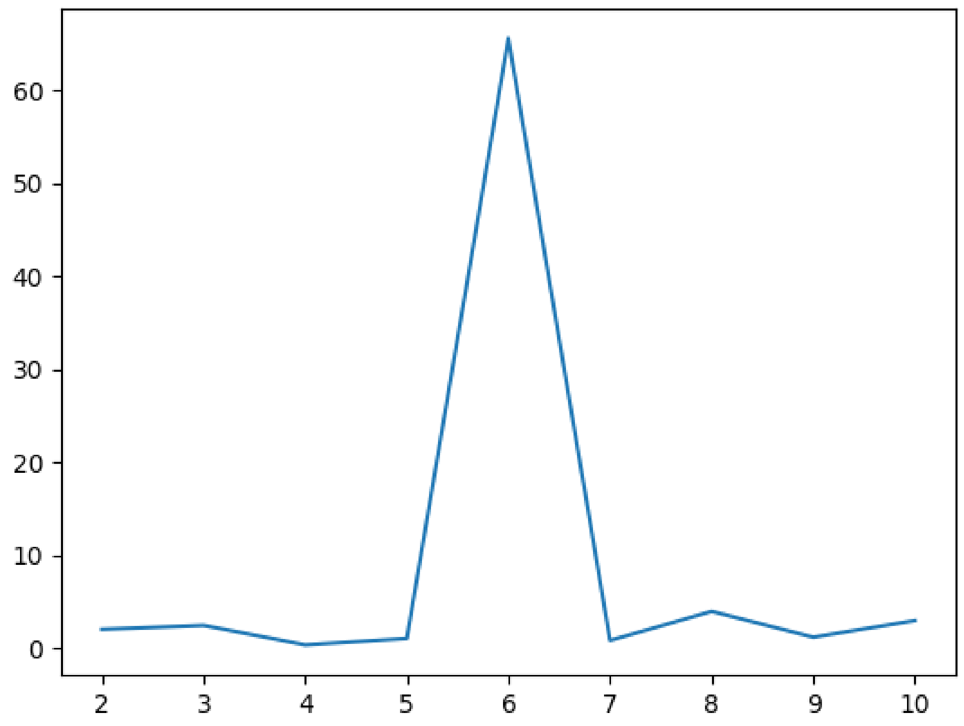
Representation error of k-means clustering with k from 2 to 10 for dataset A:



Rissannon Minimum Description Length of k-means clustering with k from 2 to 10 for dataset A:



Krzanowski and Lai of k-means clustering with k from 2 to 10 for dataset A:

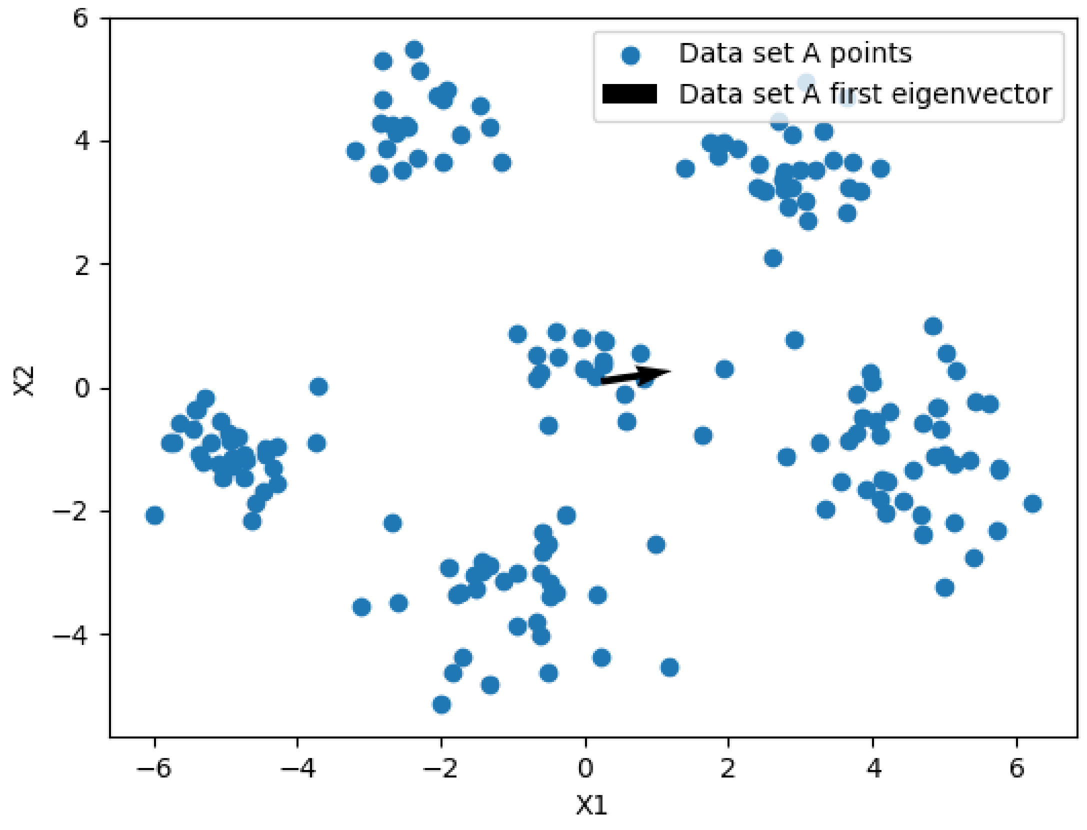


According to the result, representation error keeps dropping with the increasing of k, but the slope becomes more gentle and towards horizontal as k increases, k=6 seems to be the turning point where the slope becomes horizontal, which is the same as Rissannon Minimum Description Length, although there is no obvious turning point where the slope starts to climb up. According to the Krzanowski and Lai curve, there is an obvious maximum value when k is equal to 6, which means there are 6 natural clusters.

2.D

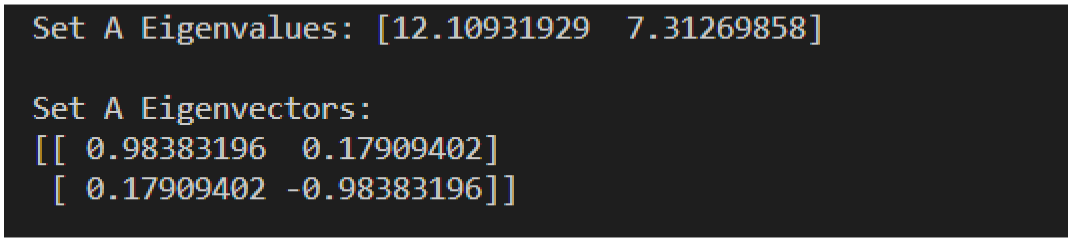
Generate a plot of the data with the direction of the first eigenvector shown as an arrow on the plot. Does the direction make sense given the data? Include in your report a plot of the original data with the eigenvector drawn on the plot.

Plot of data set A points, data set A first eigenvector:



This plot represents the data from set A where x=X1 and y=X2. The first eigenvector obtained from running PCA on the unwhitened data is also plotted. This eigenvector is shown as an arrow pointing in the primary direction of the data's variance. The direction of the arrow indicates that the direction of variance is primarily to the right and slightly up.

Eigenvalues and eigenvectors for set A:

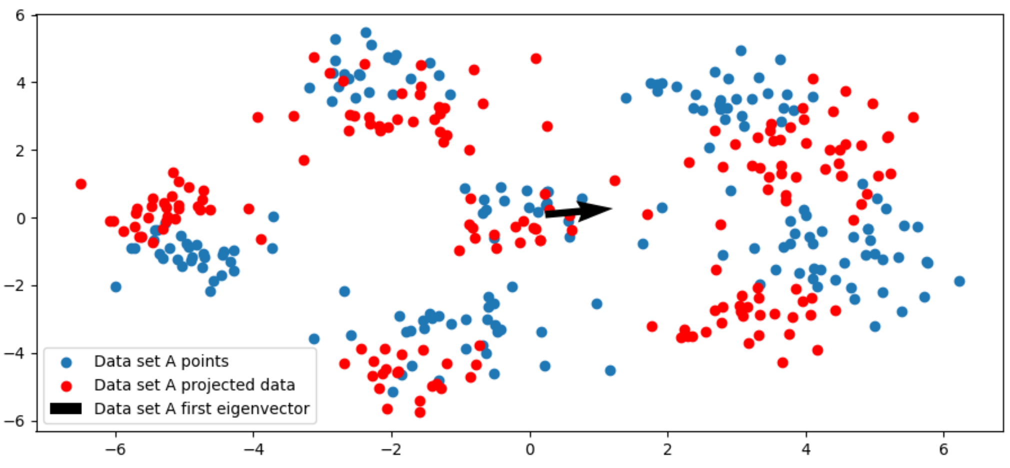


```
Set A Eigenvalues: [12.10931929  7.31269858]

Set A Eigenvectors:
[[ 0.98383196  0.17909402]
 [ 0.17909402 -0.98383196]]
```

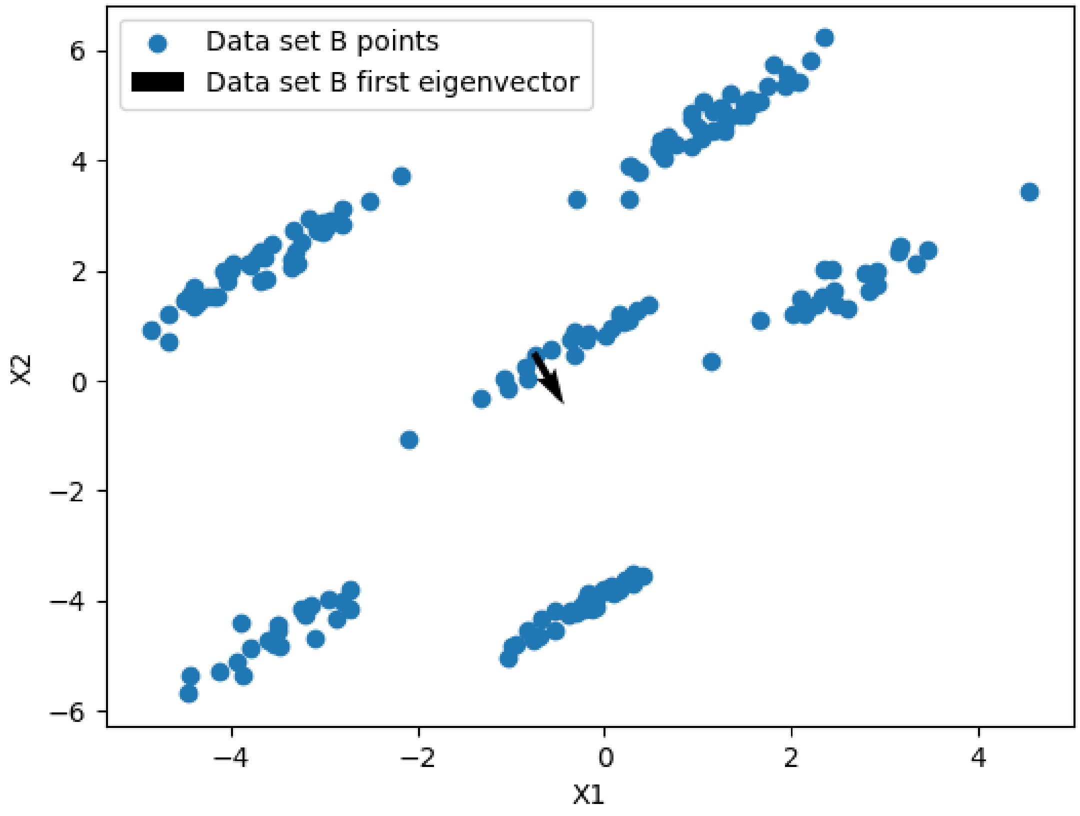
From this we see that the primary direction of variance explained by the first eigenvector is to the right as indicated by the plotted arrow. Moreover, we see that the eigenvalue of the first eigenvector is about 12.11 and the eigenvalue of the second eigenvector is about 7.31 meaning that about 62% of the data is explained by the first eigenvector. These values imply that there is not a large difference in the amount of variance captured by either eigenvector. Therefore, the data does not have much directionality.

Given the data, the eigenvector and eigenvalues are not surprising. There is not much blatant directionality in the data so one would not expect any one eigenvector to capture most of the variance. When we plot the projected data as shown below, we can see that the distribution of the data points on the components does not change much from the initial data.

Plot of data set A points, data set A projected data, data set A first eigenvector:

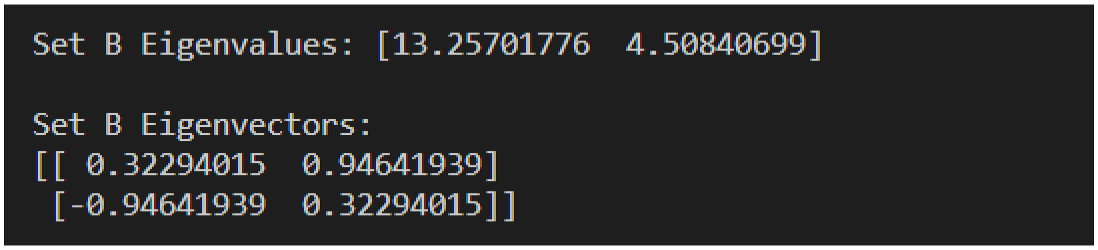


Plot of data set B points, data set B first eigenvector:



This plot represents the data from set B where x=X1 and y=X2 with the eigenvector obtained from conducting PCA on the unwhitened data plotted as a black arrow on top. The direction of the arrow implies that the primary direction of variance is down and slightly right.

Eigenvalues and eigenvectors for set B:

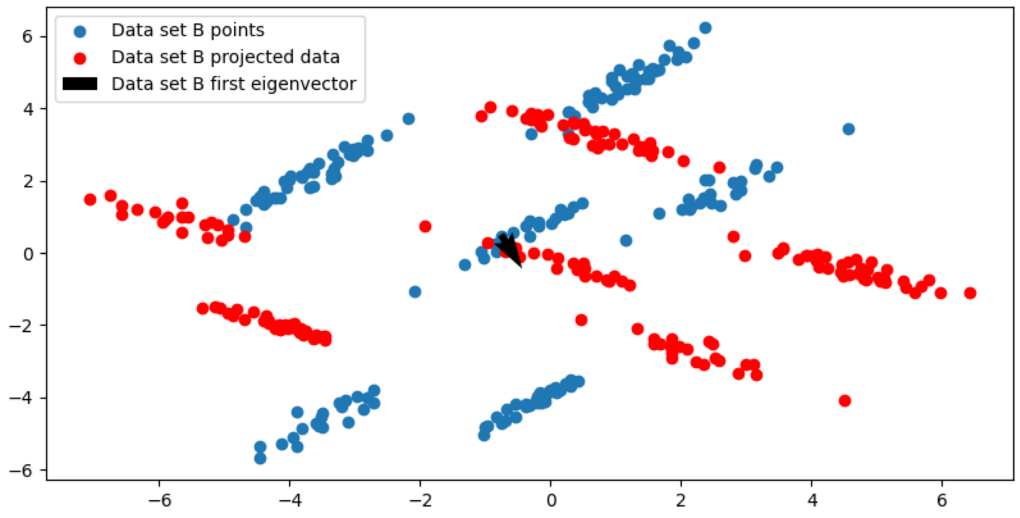


```
Set B Eigenvalues: [13.25701776  4.50840699]

Set B Eigenvectors:
[[ 0.32294015  0.94641939]
 [-0.94641939  0.32294015]]
```

From this data we see that the primary direction of variance captured by the first eigenvector is down and to the right as indicated by the arrow. Moreover, we see that the eigenvalue of the first eigenvector is about 13.26 and the eigenvalue of the second eigenvector is about 4.51 meaning that about 75% of the variance is captured by the first eigenvector. These values imply that the first eigenvector captures a majority of the variance in the data.

Given the data, there is one thing that is surprising and one thing that is not. The unsurprising result is that the first eigenvector captures the majority of the variance. When looking at the plotted data, there appears to be a large amount of variance in the diagonal direction. However, what is surprising is that this variance is primarily in the negative direction for X2. When looking at the graph, most of the directionality appears to be pointing up and to the right (i.e. the positive direction for X1 and the positive direction for X2). However, the eigenvectors indicate that is not the case. In reality, the primary direction of variance is in the positive direction for X1 and the Negative direction for X2. This can be seen more prominently when we plot the projected data on top of the regular data as seen below.

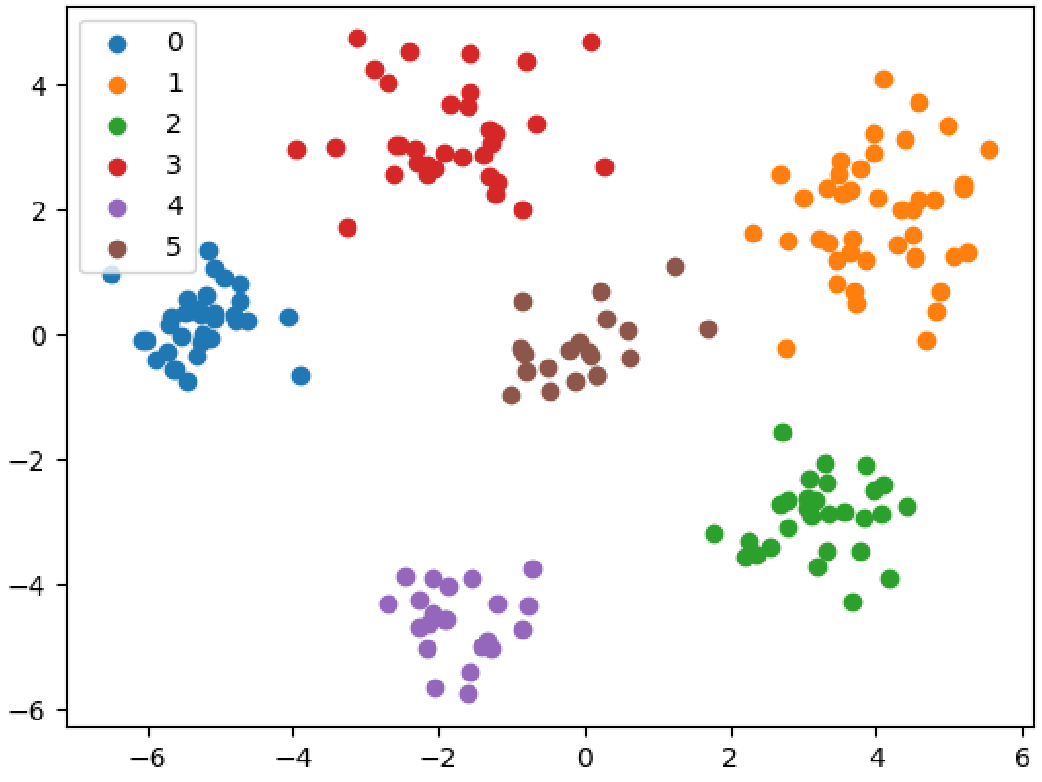
Plot of data set B points, data set B projected data, data set B first eigenvector:



2.E

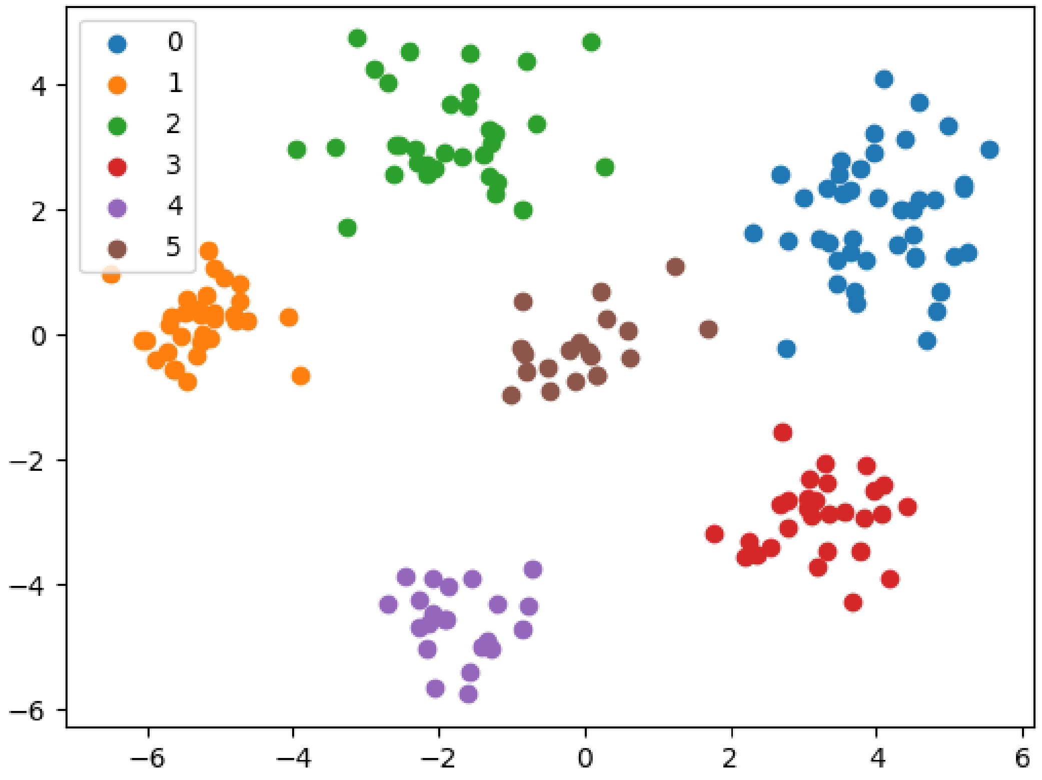
Does using the projected points improve the clustering process? As an extension, you could experiment with weighting the eigenvectors differently in your distance metric (e.g. weight by eigenvalue). Include in your report a plot of your clustering results for K=6, coloring each cluster by color.

K-means applied to Set A projected data:



Above is a plot of the result of applying K-means clustering to set A's projected data where k=6. From this plot we can see that using the projected points was effective at finding the 6 clusters in the data set. However, when we compare this plot to the K-means routine run on the raw (i.e. non-projected) data points, we see that the K-means clustering process is not improved by using the projected points.

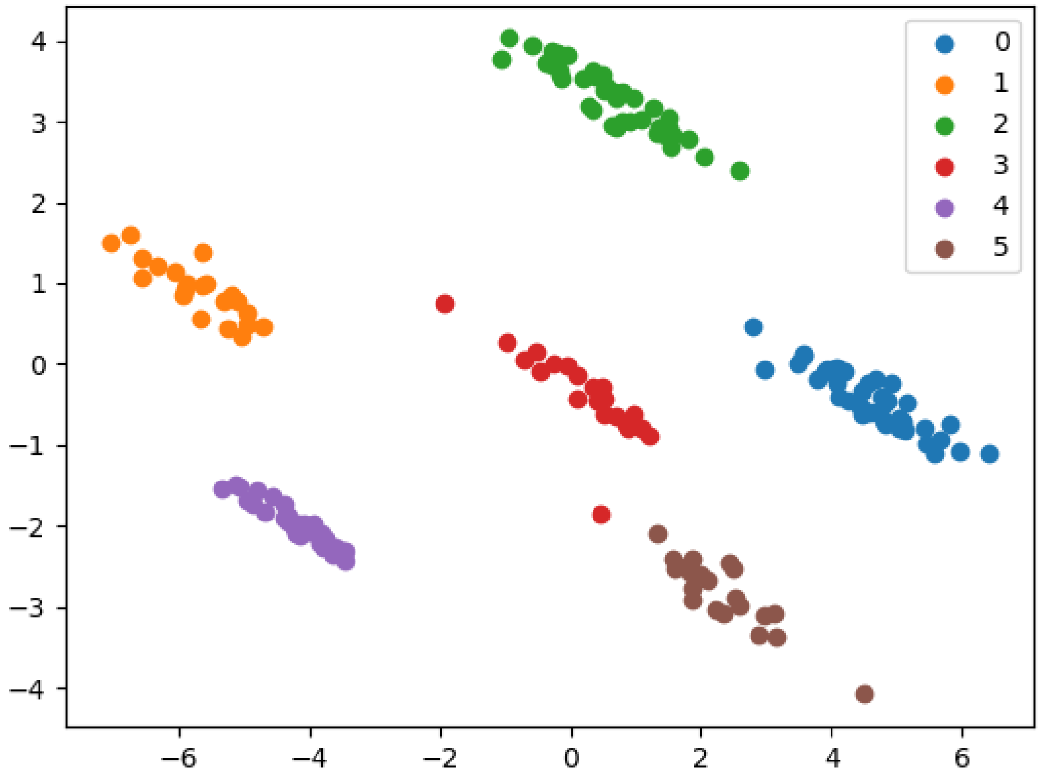
Mean shift applied to Set A projected data:



Above is a plot of the result of applying Mean shift clustering to set A's projected data. This plot shows us that using Mean shift on the projected points is effective at finding the 6 natural clusters in the data without needing to specify the number of clusters beforehand. However, similar to K-means, when comparing this plot to the K-means plot run on the raw data, we see that the clustering process is not improved by using the projected data.

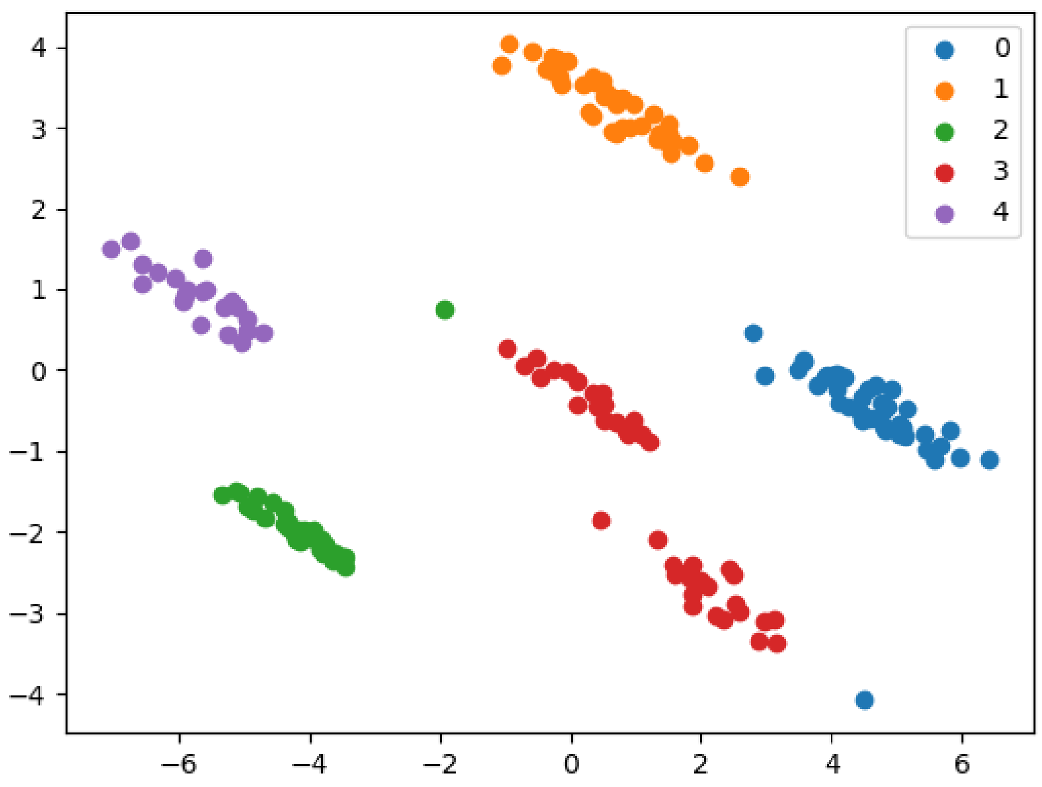
Taking both of the above plots together, we see that using the projected data did not improve the clustering process for either K-means or Mean shift. This implies that PCA is not necessary for clustering this data set because the features are already relatively independent. Therefore, these features are effective at clustering the data.

K-means applied to Set B projected data:



Above is a plot of the result of applying K-means clustering to set B’s projected data where k=6. From this plot we can see that using the projected points was relatively effective at finding the 6 clusters in the data set except for a few outliers. Comparing this plot with the plot of K-means applied to the raw data shows that the clustering process is not improved by using the projected data.

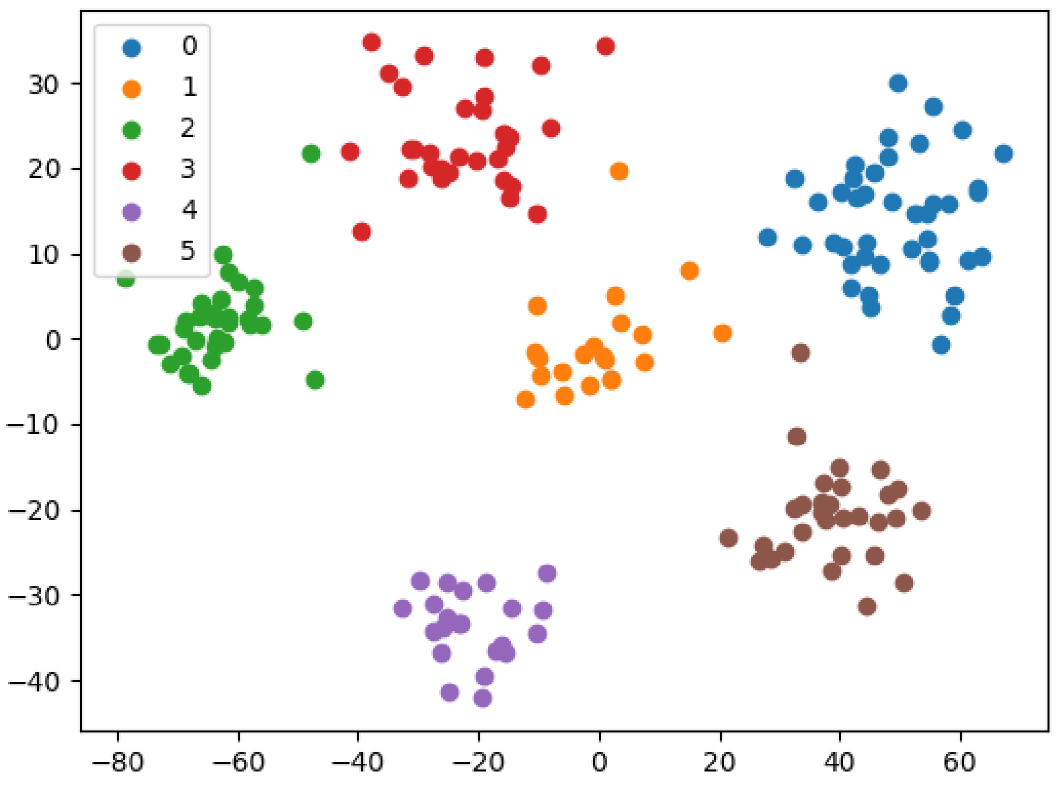
Mean shift applied to Set B projected data:



Above is a plot of the result of applying Mean shift clustering to set B's projected data. This plot shows us that using Mean shift on the projected points is not effective at finding the natural clusters in the data. The red cluster shown on the plot should be two distinct clusters but the Mean shift classifier was unable to detect that difference. Therefore, the reduced dimensionality of the projected data resulted in a worse outcome for the K-means routine compared to using the raw data.

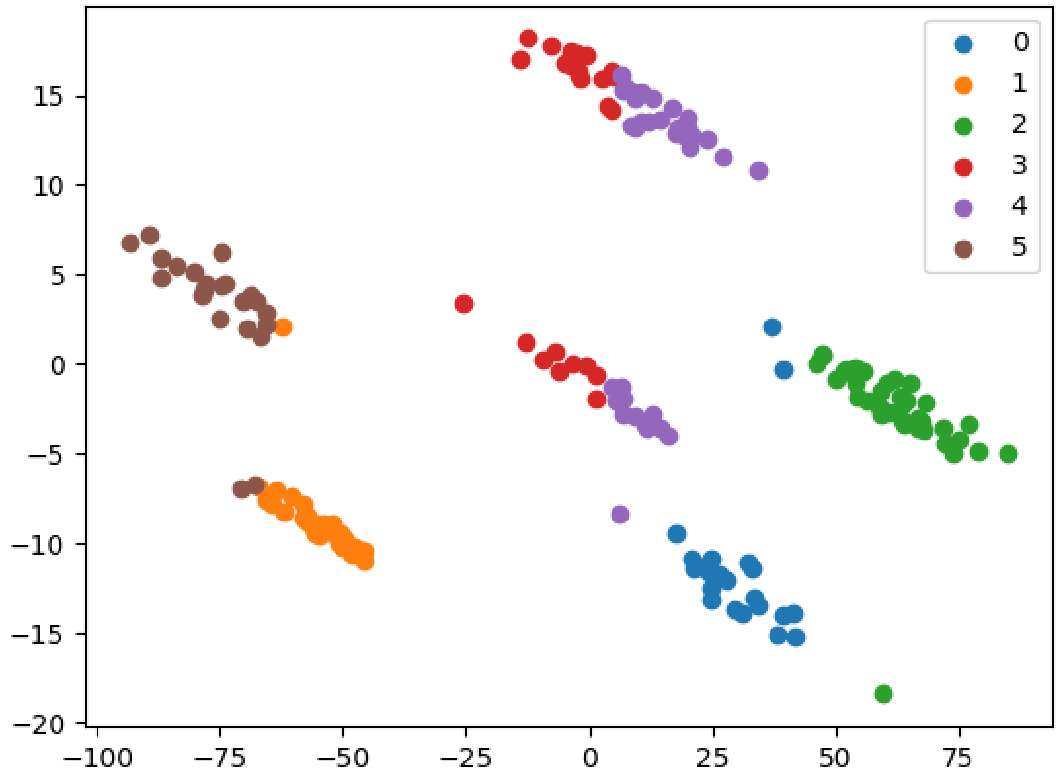
Taking both of the above plots together, we see that using the projected data did not improve the clustering process for either K-means and worsened the clustering process for Mean shift. This worse outcome for Mean shift implies that projected data should not be used for clustering the data because using the raw features has a more effective clustering outcome.

K-means applied to Set A projected data with distance weighted by eigenvalues:



Above is a plot of K-means applied to the projected data for data set A with its eigenvectors weighted by the eigenvalues. Weighting by eigenvalues did not have much of an effect on the clustering as the same 6 natural clusters were still identified.

K-means applied to Set B projected data with distance weighted by eigenvalues:



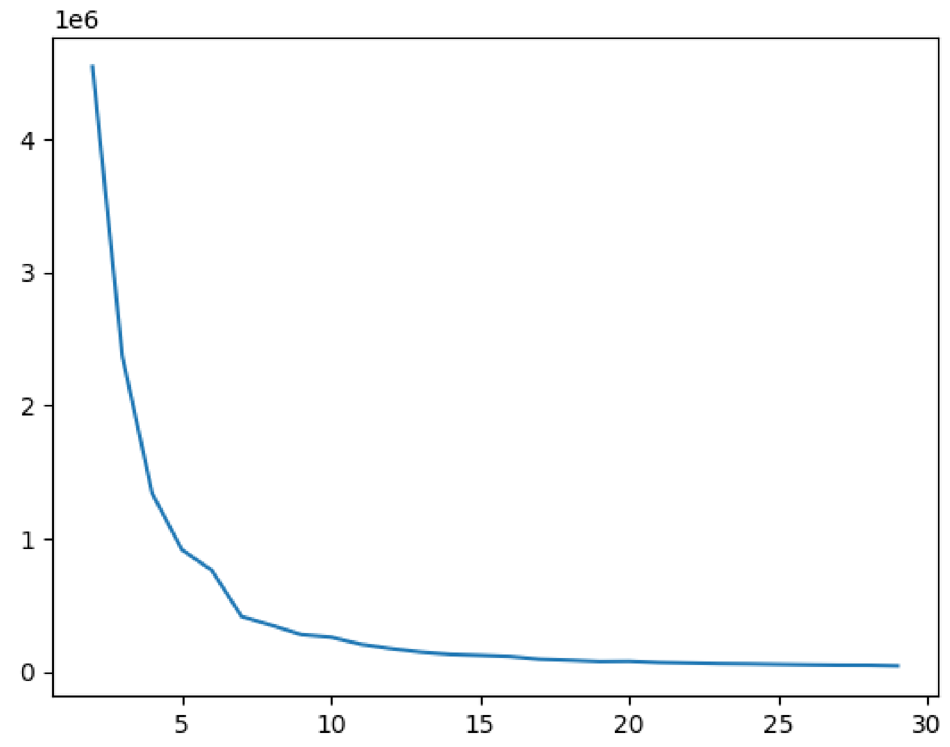
Above is a plot of K-means applied to the projected data for data set B with its eigenvectors weighted by eigenvalues. Weighting by eigenvalues had a largely negative effect as most of the natural clusters were misidentified.

Task 3

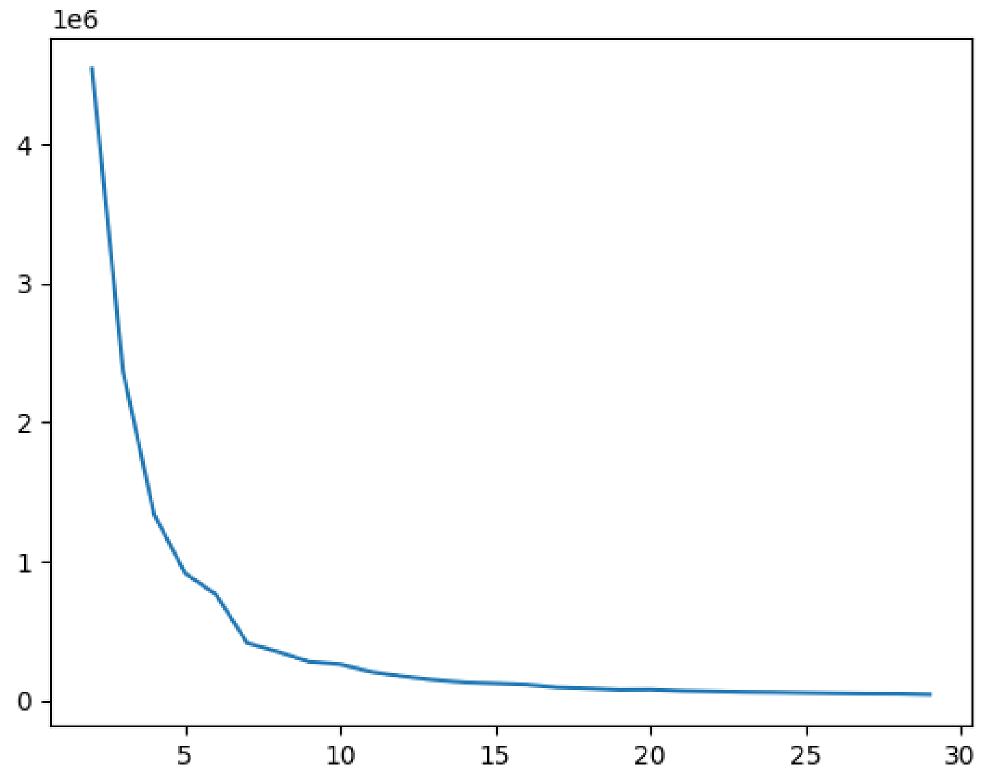
Use K-means to cluster your own data for a range of K you think is appropriate (keep the numbers small). Are there natural clusters in your data set? If you apply the cluster quality metric you implemented earlier, what does it tell you about the best K for your data set?

For this task, we are using wine_cluster.csv dataset, which contains 179 rows and 13 features, it is saved under the datasets folder.

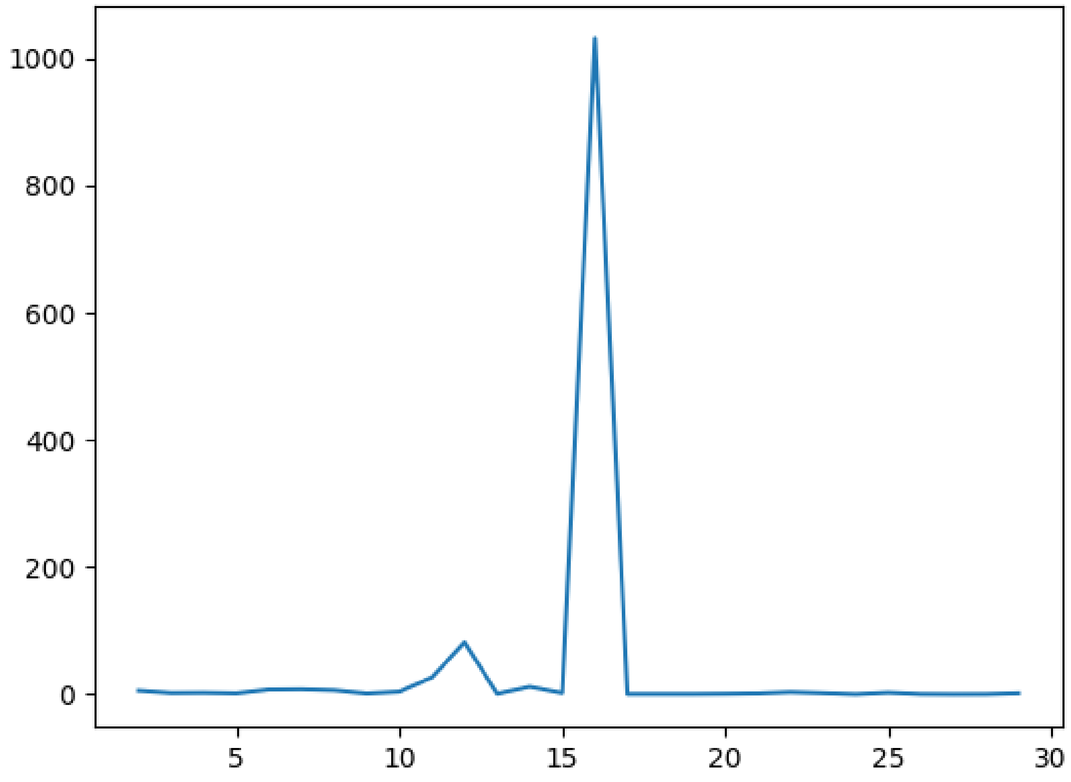
Representation error of k-means clustering with k from 2 to 30 for wine dataset:



Rissannon Minimum Description Length of k-means clustering with k from 2 to 30 for wine dataset:



Krzanowski and Lai of k-means clustering with k from 2 to 30 for wine dataset:



According to the result, representation error keeps dropping with the increasing of k, but the slope becomes more gentle and towards horizontal as k increases, k=10 seems to be the turning point where the slope becomes horizontal, which is the same as Rissannon Minimum Description Length, although there is no obvious turning point where the slope starts to climb up.

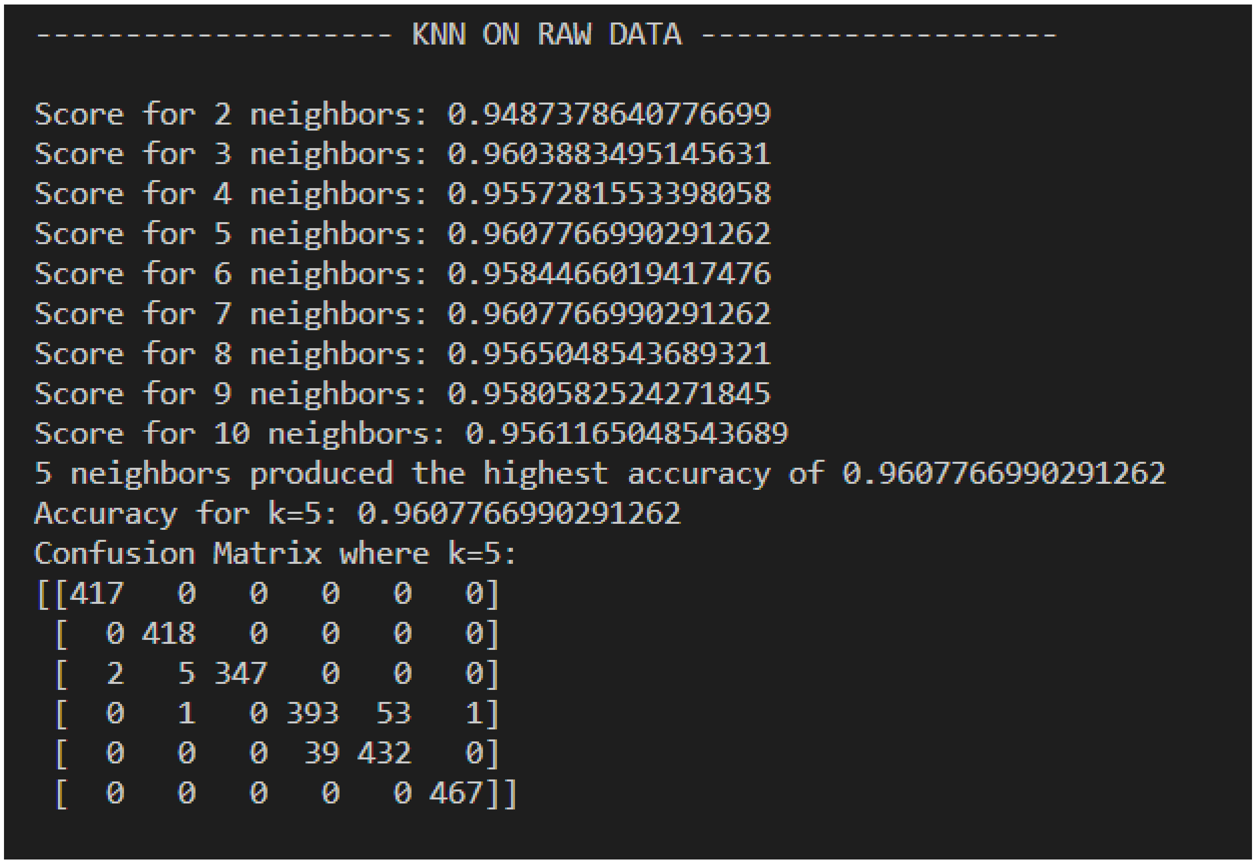
According to the Krzanowski and Lai curve, there is an obvious maximum value when k is equal to around 16, which means there are around 16 natural clusters.

Task 4

In your report discuss the performance of the two classifiers and the results of the PCA projection.

The UCI Activity Recognition data set was used for this task. The activity labels for the data set are "Walking" (label 1), "Walking_Upstairs" (label 2), "Walking_Downstairs" (label 3), "Sitting" (label 4), "Standing" (label 5), and "Laying" (label 6). Although the UCI data set came pre-partitioned into training and test sets, we recombined and repartitioned the data into 25% test and 75% training as the instructions originally stated to do. The following images display the results of various classifiers on the data set.

KNN Classifier on raw UCI data set:



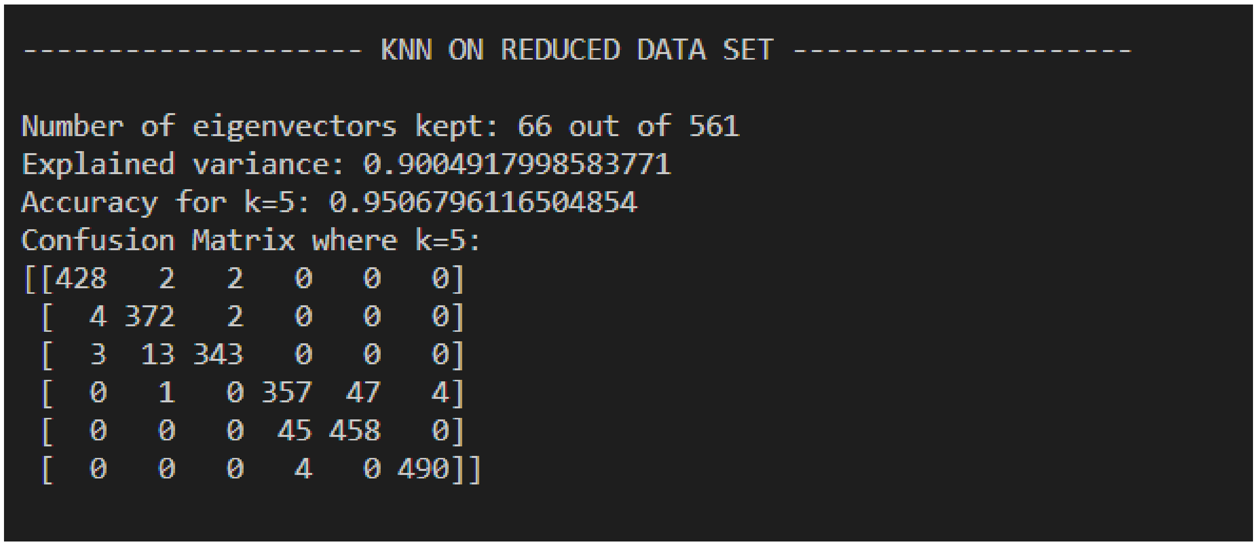
```
-------------------- KNN ON RAW DATA --------------------

Score for 2 neighbors: 0.9487378640776699
Score for 3 neighbors: 0.9603883495145631
Score for 4 neighbors: 0.9557281553398058
Score for 5 neighbors: 0.9607766990291262
Score for 6 neighbors: 0.9584466019417476
Score for 7 neighbors: 0.9607766990291262
Score for 8 neighbors: 0.9565048543689321
Score for 9 neighbors: 0.9580582524271845
Score for 10 neighbors: 0.9561165048543689
5 neighbors produced the highest accuracy of 0.9607766990291262
Accuracy for k=5: 0.9607766990291262
Confusion Matrix where k=5:
[[417   0   0   0   0   0]
 [  0 418   0   0   0   0]
 [  2   5 347   0   0   0]
 [  0   1   0 393  53   1]
 [  0   0   0  39 432   0]
 [  0   0   0   0   0 467]]
```

The above image shows the accuracy and confusion matrix for a K-Nearest Neighbor model run on the given UCI data set. Before building the classifier, the `find_optimal_num_neighbors` function in `knn.py` was run to determine the value of k that would produce the highest scoring classifier for the given training and test data. This determined that the optimal value for k was 5. Next, `build_and_fit_knn_classifier` in `knn.py` was run to build the K-Nearest Neighbor classifier with k=5.

From the above metrics of the knn classifier on the raw data, we see that it has an accuracy of about 0.96. This implies that knn is an effective classifier for the given data set. Moreover, we see from the confusion matrix that most of the activity labels were correctly predicted. The activities which the classifier most often predicted incorrectly were “Sitting” (label 4) and “Standing” (label 5): “Sitting” was incorrectly labeled as

“Standing” 53 times and “Standing” was incorrectly labeled as “Sitting” 39 times. This implies that some of the features correlated with “Standing” and “Sitting” are correlated.

KNN Classifier on 66 feature reduced UCI data set:

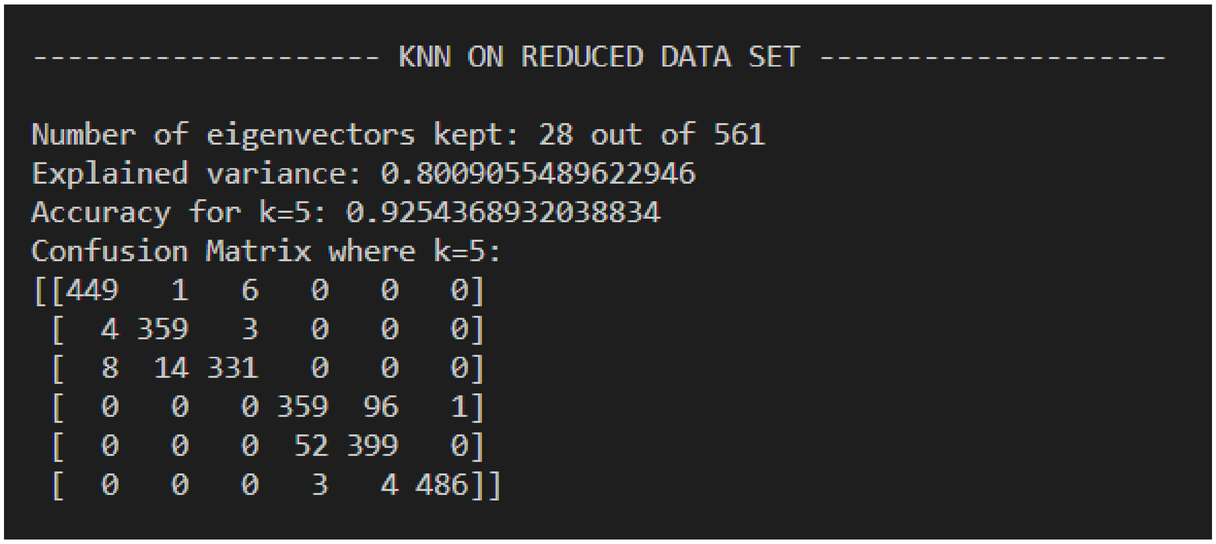


```
-------------------- KNN ON REDUCED DATA SET --------------------

Number of eigenvectors kept: 66 out of 561
Explained variance: 0.9004917998583771
Accuracy for k=5: 0.9506796116504854
Confusion Matrix where k=5:
[[428   2   2   0   0   0]
 [  4 372   2   0   0   0]
 [  3  13 343   0   0   0]
 [  0   1   0 357  47   4]
 [  0   0   0  45 458   0]
 [  0   0   0   4   0 490]]
```

The above image shows the accuracy and confusion matrix for the second K-Nearest Neighbor classifier which was trained on a reduced data set with 66 dimensions. The reduced data set was obtained by running PCA on the raw data set, and keeping enough dimensions of the projected data to explain 90% of the variance in the raw data set. The value of k was kept at 5 to minimize difference from the original classifier to enable easier comparison.

From the above metrics we can see that the KNN classifier for the reduced data set has an accuracy of 0.95. This is a decrease of only 0.01 compared to the KNN classifier for the raw data meaning that the feature selection was highly successful because many of the features in the original data are highly correlated. Moreover, the confusion matrix shows that the same mistakes were made by the classifier on the reduced data as the classifier on the raw data: “Sitting” and “Standing” were the most frequently confused.

KNN Classifier on 28 feature reduced UCI data set:

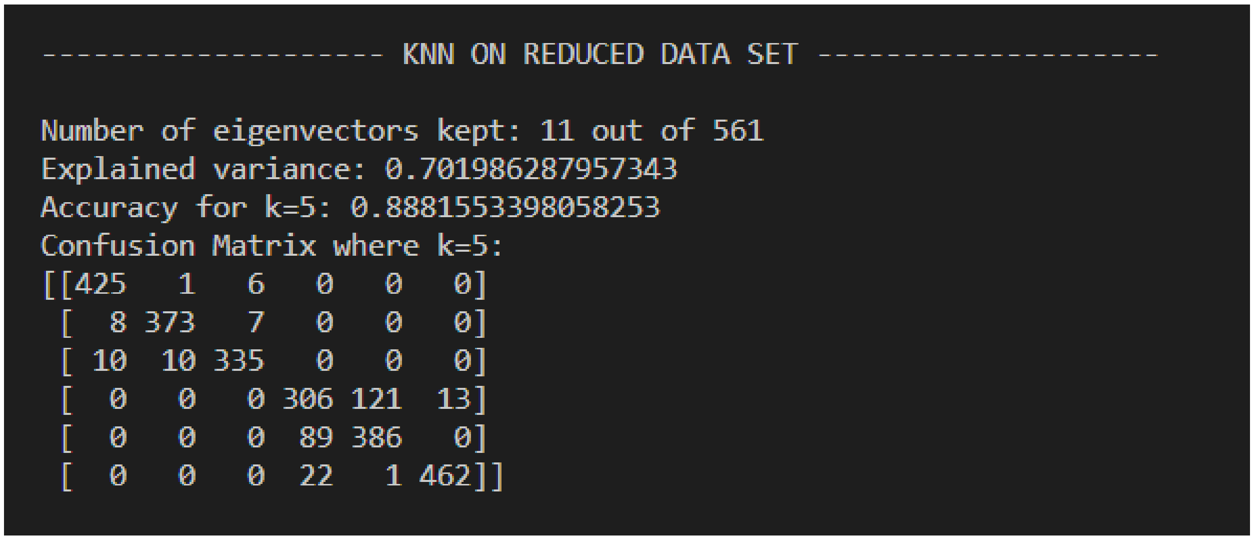


```
-------------------- KNN ON REDUCED DATA SET --------------------

Number of eigenvectors kept: 28 out of 561
Explained variance: 0.8009055489622946
Accuracy for k=5: 0.9254368932038834
Confusion Matrix where k=5:
[[449   1   6   0   0   0]
 [  4 359   3   0   0   0]
 [  8  14 331   0   0   0]
 [  0   0   0 359  96   1]
 [  0   0   0  52 399   0]
 [  0   0   0   3   4 486]]
```

As part of the extension, the number of dimensions after PCA was reduced further to 28 which accounted for about 80% of the variance in the data. The value of k was kept at 5 to minimize difference from the original classifier to enable easier comparison.

From the above metrics we can see that the KNN classifier for the reduced data set has an accuracy of about 0.93. This is a decrease of about 0.03 compared to the classifier for the raw data meaning that the feature selection did have a more significant impact on the accuracy than the second classifier did. The accuracy is still relatively high, but the confusion matrix shows that it is making the same mistakes as the first two classifiers at a more frequent rate: “Sitting” was mistaken for “Standing” 96 times and “Standing” was mistaken for “Sitting” 52 times.

KNN Classifier on 11 feature reduced UCI data set:



```
-------------------- KNN ON REDUCED DATA SET --------------------

Number of eigenvectors kept: 11 out of 561
Explained variance: 0.701986287957343
Accuracy for k=5: 0.8881553398058253
Confusion Matrix where k=5:
[[425   1   6   0   0   0]
 [  8 373   7   0   0   0]
 [ 10  10 335   0   0   0]
 [  0   0   0 306 121  13]
 [  0   0   0  89 386   0]
 [  0   0   0  22   1 462]]
```

Again as part of the extension, the number of dimensions after PCA was reduced even further to 11, which accounted for about 70% of the variance in the data. The value of k was kept at 5 to minimize difference from the original classifier to enable easier comparison.

From the above metrics we can see that the KNN classifier for the reduced data set has an accuracy of about 0.89. This is a decrease of about 0.07 compared to the classifier for the raw data meaning that the feature selection had the largest negative impact on classifier accuracy so far. Looking at the confusion matrix, we see that the “Sitting”/“Standing” mistake was made at an even greater rate. Moreover, there are larger increases labeling mistakes for other activity types.

Taking the results of these four classifiers together, we can infer that the most effective classifiers were the KNN classifier for the raw data set and the KNN classifier for the reduced data set with 66 dimensions (i.e. that which captured 90% of the variance). These two classifiers only had an accuracy difference of about 0.01. Therefore, if one wanted to reduce the complexity of the feature space but retain about the same accuracy, they should prefer the KNN classifier on the reduced data set. On the other hand, if they did not want to sacrifice feature space complexity, they should prefer the KNN classifier on the raw data set.

Reflection

Some of the things we learned during the course of this project include:

- During task 1, we learned how to implement a (both normalized and non_normalized euclidean) **distance metric** for determining the distance between samples in a data set, and how to use that distance metric within the **K-Nearest Neighbor classification algorithm**. And we also learnt the **majority voting** with all the k nearest neighbors to determine the best matched category. We also learned how to evaluate the precision of a classification algorithm and how to read (and plot) the resulting **confusion matrix.** We also modified the code by giving k a different number/ try different price ranges, trying normalized/un_normalized data to find the optimal classifier and saw how they actually affect the precision. We also had a closer look at the actual datasets and printed out the result that this classifier provides, and understood how this can really help us with prediction in real life.
- During task 2 and 3, we learned how to cluster data using various sklearn clustering algorithms (including **k-means, mean-shift, and hierarchical clustering**), and understood how those algorithms work behind the scenes. We also learnt how to evaluate the cluster quality by calculating and plotting the

representation errors, cluster quality metrics including **Rissannon Minimum Description Length, Krzanowski and Lai**. We had a better and more straightforward understanding of the cluster quality by comparing the data plot with the cluster quality metrics, and saw there are 6 natural clusters in dataset A, and **Krzanowski and Lai** reached the maximum value when k=6.

- During task 4, we learned how to pull data from the UCI Machine Learning Repository and use that real world data to build effective **KNN classifiers**. We also learned how to utilize **PCA** to carry out feature reduction to reduce the dimensionality of our data while retaining the accuracy of our classifiers.