Project 1 Technical Report

Drake Song

Project Description

Project 1 aims to utilize three different machine learning algorithms (K-Nearest Neighbors, K-Means Clustering, and Naïve Bayes) on the same dataset to compare and contrast each algorithms' efficiency and characteristics. Since the labels of each data are already known, it is easy to determine the error produced by each algorithm. To be more specific about the nature of the problem, the algorithms in question are all used for classification purpose of the data.

Problem Statement

Given the currency data ('data_banknote_authent.csv' and 'swiss_banknote.csv') which machine learning algorithm out of K-Nearest Neighbors, K-Means Clustering, and Naïve Bayes demonstrate the most accurate model of the data and correctly predict test data's labels?

Feature Selection Rationale

The data for 2D plots come from 'data_banknote_authent.csv.' The features selected are: Column 1 and Column 2. These two features produced the biggest confidence intervals (farthest from 0) which signifies that there is a high confidence of clear distinction between the 'real' data and the 'fake' data. Here are the calculated confidence intervals:

Feature	Confidence Interval
Column 1	[3.9363, 4.3539]
Column 2	[4.6897, 5.8107]
Column 3	[-1.8055, -0.8976]
Column 4	[-0.1249, 0.3229]

The data for 3D plots come from 'swiss_banknote.csv.' The features selected are: diag_length, bottom_margin, top_margin. These three were also selected by examining the confidence intervals of all the features from the data set. Here are the confidence intervals:

Feature	Confidence Interval
length	[0.0427, 0.2493]
left_width	[-0.4447, -0.2693]
right_width	[-0.5645, -0.3815]
bottom_margin	[-2.4816, -1.9683]
top_margin	[-1.1442, -0.7858]
diag_length	[1.9260, 2.2080]

Model Description

K-Nearest Neighbors

K-Nearest Neighbors is a non-parametric classification tool (can also be used for regression purpose) that utilizes Euclidean distance (in most cases) to search for *k*-nearest neighbors and assign each data a class according to the majority vote of its neighbors. Although KNN algorithm provides a simple yet accurate output, the runtime complexity can get quite large depending on the size of the dataset due to the fact that for each data, the distance must be calculated.

K-Means Clustering

This algorithm aims to assign a group of similar data in to a cluster. There are two main steps in K-Means Clustering algorithm:

1. Assign each data point to the closest centroid (Euclidean distance)
2. Move centroid position based on means computed using centroid assignments

These two steps are usually repeated multiple times until a satisfactory result has been achieved.

Naïve Bayes

Naïve Bayes is a classification method that utilizes multiple probability algorithms and Bayes' Theorem to predict the label of data. This is a widely used technique that produces better models than other techniques such as logistic regression and utilizes less data which means faster runtime. Naïve Bayes calculate the probabilities of a data point being in certain groups and outputs the most probable group the data point belongs in.

Assumptions

K-Nearest Neighbors

There are two main assumptions when using KNN classification algorithm: the dataset is in a vector space and the closer the data are to each other, the similar they are in terms of features and more importantly, the class/label.

The important defaults used for the *fitcknn* function in MatLab are: 'euclidean' for distance calculation and 1 for 'NumNeighbors' (number of nearest neighbors to find for when classifying each point when predicting).

K-Means Clustering

For 2D plot, initial cluster assignment is done through random initialization. 3D plot uses what MatLab calls *k-means++ Algorithm* which uses heuristics to find centroids.

In determining the number of clusters, because it is know that the data has two classes, real and fake, K = 2 was used as the number of clusters. Using the elbow method produced K≈4, however, it was more important to follow the data rather than the result of the elbow method.

When finding updated position of the centroids, the following equation was used:

$$\mu_k := \frac{1}{|C_k|} \sum_{i \in C_k} x^{(i)}$$

Here C_k represents the number of data that is assigned to the cluster *k*. This computes the mean of the points that are assigned to the clusters.

Naïve Bayes

Bayes' Theorem assumes that the data are independent. To be more exact, Naïve Bayes assumes that the presence of a feature in data is not directly related to other features in the dataset. This is important in calculating the probabilities.

There is one main assumption (default) that is used by MatLab. When determining the model of distribution of the data, *fitcnb* uses Normal Gaussian Distribution which is generally used in most cases.

Model Analysis

K-Nearest Neighbors

Because *k* (number of neighbors to analyze) is set to 1, the model (Figure 1-1) is extremely accurate in predicting the labels for the training dataset. This raises the question of overfitting the data; however, due to the nature of given data, that doubt is quickly dismissed. Setting k to something bigger such as 3 causes more error to show up since there are many overlapping data in the middle of the graph which makes the data hard to predict. In fact, the overlapping data in the middle is the cause of the error shown when running the test data (Figure 1-2) [Circle is predicted label. Plus (+) is true label.].

Looking at 3D plots (Figure 4-1 and 4-2), because there are significantly less number of data and *k* is still 1, the prediction is really accurate which is demonstrated by 0 error for both training and test data.

K-Means Clustering

KMC for the 2D plot turned out to be a bit tricky. From looking at K-Nearest Neighbors plot (Figure 1-1), it is observable that there is an overlap in the middle and that there are no clear clusters within the data. This plays a huge role when using KMC on this dataset. Looking at Figure 2-1, the algorithm is forming a cluster at the top half of the data and another cluster at the bottom half of the data (the final cluster positions is marked by magenta 'x'). This is a reasonable approach considering the fact that there are no clear distinctions between data. Also, the fact that there are differing numbers of 'real' and 'fake' data on top of having no clear clusters adds on to the error that is already pretty large. Because the algorithm is trying to find the middle ground between the data points, it tends to even out the number of data per cluster in this case. This causes a problem because there are about 100 more 'real' data than there are 'fake' data. Due to the training data having little success, the test data also produce a large number of error (Figure 2-2).

In order to correct this big error, multiple techniques were tried, however, they all failed to produce the actual labels. Some of the distance methods used are: 'cityblock', 'cosine', and 'correlation', all available through MatLab. I've also tried varying the number of features and types of features used to train the data. Other initialization methods of centroids were used such as 'cluster', 'sample', and 'uniform'. The only solution that showed promise was decreasing both the number of features and the number of data; however, this is not what we're looking for here in this project.

Due to the random initialization of clusters in the beginning and because the algorithm's nature is to simply cluster similar data together, there is a possibility that the algorithm wrongly labels the clusters. In Figure 2-3, the blue cluster is on the top half of the data while the red cluster is on the bottom half of the data. Because the initial centroid for the blue cluster was assigned towards the top while the initial centroid for the red cluster was assigned towards the bottom, this instance of the algorithm falsely labels the clusters which causes a huge error in both training data and testing data (Figure 2-4). However, this shouldn't cause any problems nor negatively affect the results of KMC since it's pretty much the same as when the clusters are correctly labeled.

The 3D plots produce better results (Figure 5-1). This is mainly due to the data having clear distinctions between the clusters. This clear distinction between data helps the algorithm to form two clusters which produce small to no error in both training and testing data (Figure 5-2). The error that is caused by random initialization is still present though (Figure 5-3 and Figure 5-4). Even though MatLab's *k-means++ Algorithm* uses heuristics, it still uses randomly initialization as the first step to determine where to start. This randomly initialization, as mentioned for 2D plots, can cause problems where the labels of the clusters are falsely assigned (Figure 5-3 and Figure 5-4).

Naïve Bayes

Looking at Figure 3-1, it is clear that Naïve Bayes performs better than K-Means Clustering but not as well as K-Nearest Neighbors. In fact, it seems like there is a vertical line

that separates the data into two groups: blue (left) and red (right). This is most likely due to the fact that Naïve Bayes assumes independence of features within the data; in fact, the absence and the presence of features in a dataset does not affect the probabilities calculated in determining the most probable classification of data. With this in mind, it is reasonable that the algorithm assigns the top part of the data as blue instead of red (the part where it looks like a peninsula with the left part blue and right part red when in fact the entire peninsula should be red). KNN assigns these points as red because that particular algorithm looks at the data's neighbors and determines which group it belongs in. NB, however, does not take its neighbors in consideration. In fact, it uses mainly probabilities of a data belonging in certain groups to predict the labels. In this case, we can see that the algorithm has determined a vertical line in the middle of the data where the probabilities split. Because of this line, the middle overlapping part of the data (refer to Figure 1-1 for accurate representation of the data) is not correctly labeled; rather, the algorithm strictly follows the probability and assigns it to the most probable classification.

As we mentioned above for KMC 3D, we know that there is a clear distinction between the clusters of data with no overlap. This is also helpful for Naïve Bayes classification; the algorithm draws an invisible probability line that it did for 2D data somewhere between the two clusters of data (Figure 6-1) and correctly predicts the classification for the data. This is why the error is little to none in both training and testing data (Figure 6-2).

Conclusion

It is clear that for 'data_banknote_authent.csv' (2D), K-Nearest Neighbors performed the best and for 'swiss_banknote.csv' (3D), all three classification techniques performed well. KNN did the best for 2D due to how the data is portrayed. There are two main problems with the data that was chosen: no clear clusters and overlap of data points from different classes. The only reason KNN was successful is because of k (number of neighbors to observe) is set to one. If it were to set something bigger than one, KNN would've produced more errors due to the overlap in the middle of the data. Naïve Bayes on the other hand, produced errors mostly around the middle where the data from two different classes overlap. As mentioned before, this is mainly due to the fact that NB doesn't consider the neighbors of a data point; rather, the algorithm calculates the probabilities of the data belonging in different classes and chooses the most probable one. K-Means Clustering did horrible for this dataset even when disregarding the error caused by random initialization of the first set of centroids. Because there are no clear clusters or rather clear distinction between the two classes of data (even after using confidence intervals to figure out the best features to analyze), KMC has no choice but to simply divide the data into two halves.

The results are quite different for 'swiss_banknote.csv' (3D). This is mainly due to the fact that the data used to analyze here are nicely presented and distinctive. Looking at Figure 4-1 as the example, it is clear that the data are clearly separated into two distinctive clusters. Other than for KMC where the error only come up when the random initialization of the centroids mislabel the cluster, all three algorithms are showing little to no error.

Andrew Ng once said, “It’s not who has the best algorithm that wins; it’s who has the most data.” However, the second part of the quote should say “best” data not “most” data. From this project, it is clear to see that depending on what kind of data we’re working with, the efficiency and accuracy of different algorithms change. In fact, there is no single algorithm that does perfectly for any type of data. Also, having the most data is not always helpful when the wrong set of features are used to analyze the data (although having a lot of data does help tremendously in most cases). Choosing the most appropriate features and having the right amount of data are the two important steps in setting up the data so that the chosen algorithm for analysis performs well.

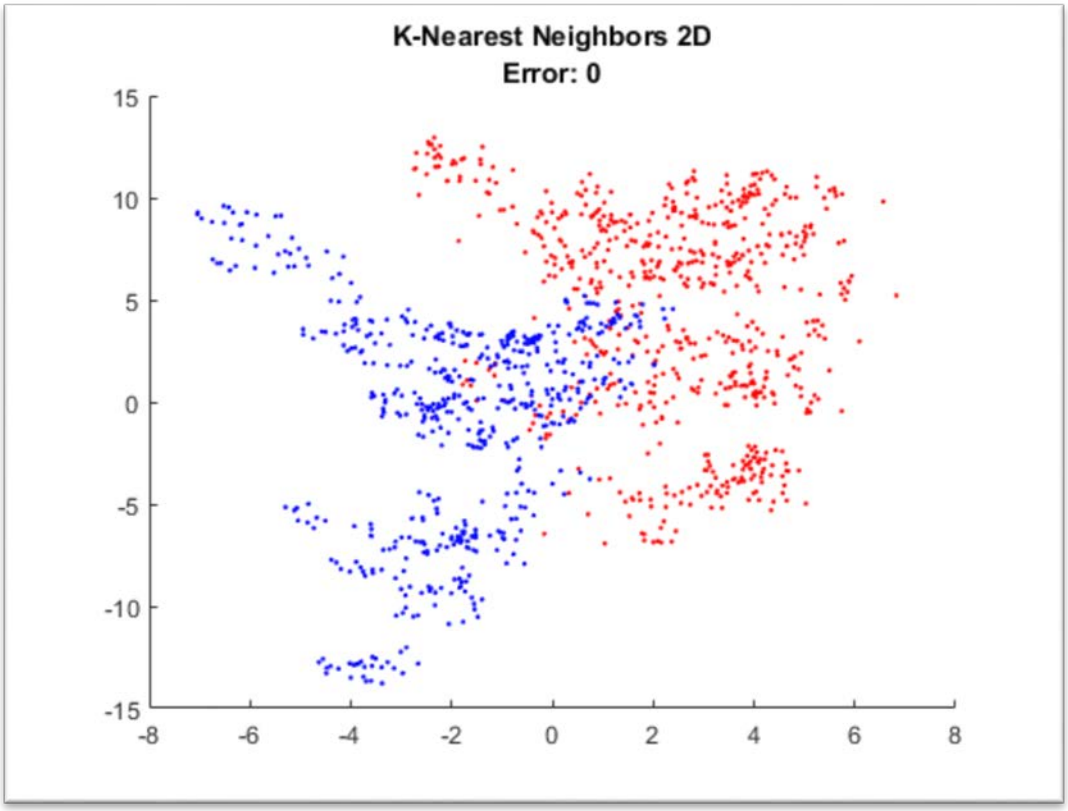

Figure 1-1

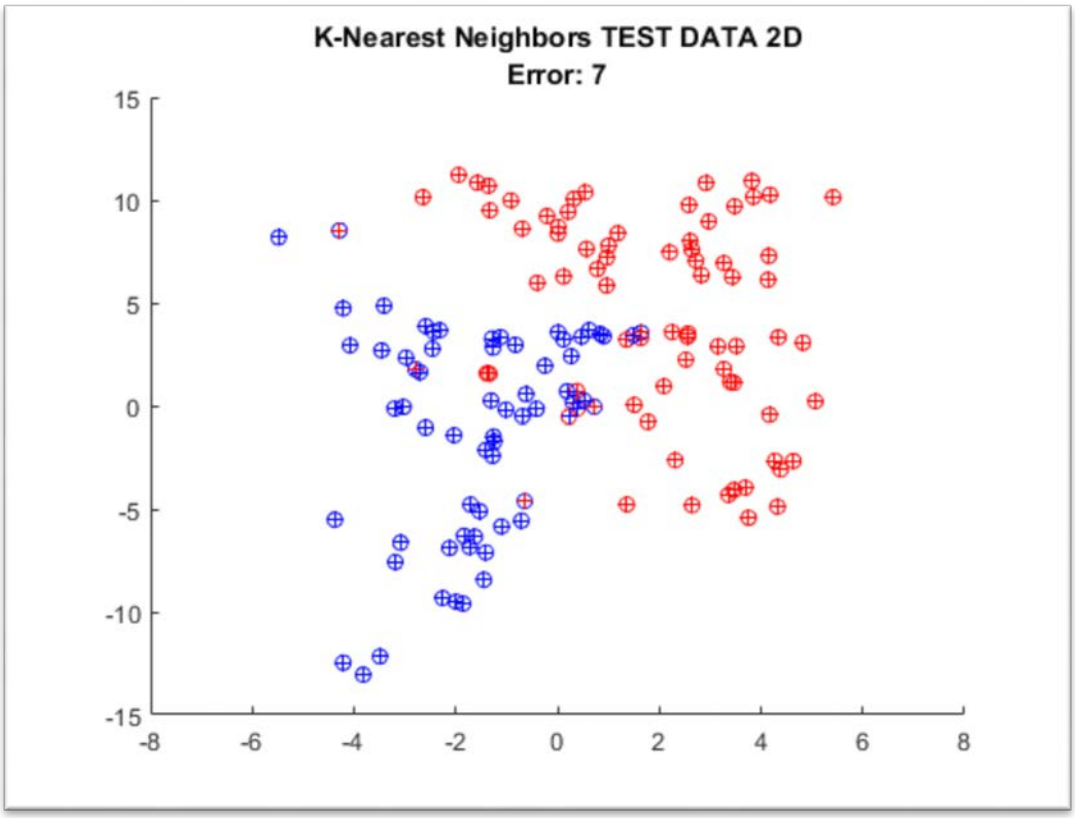

Figure 1-2

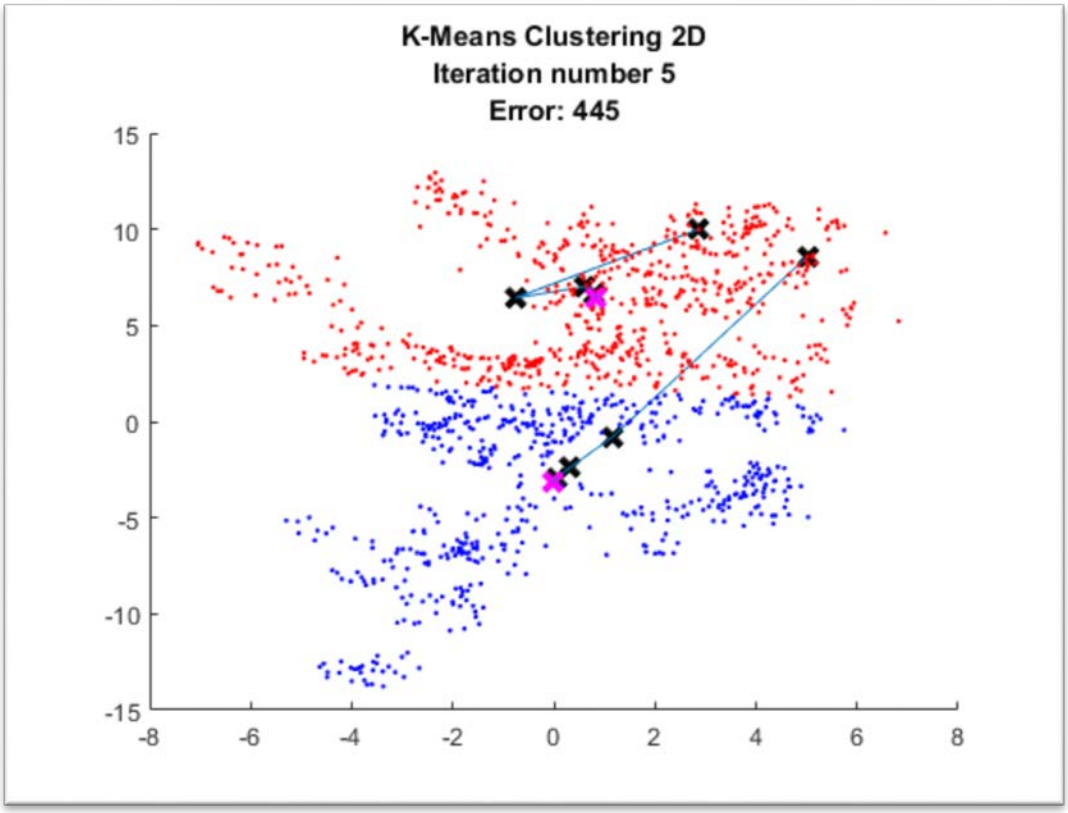

Figure 2-1

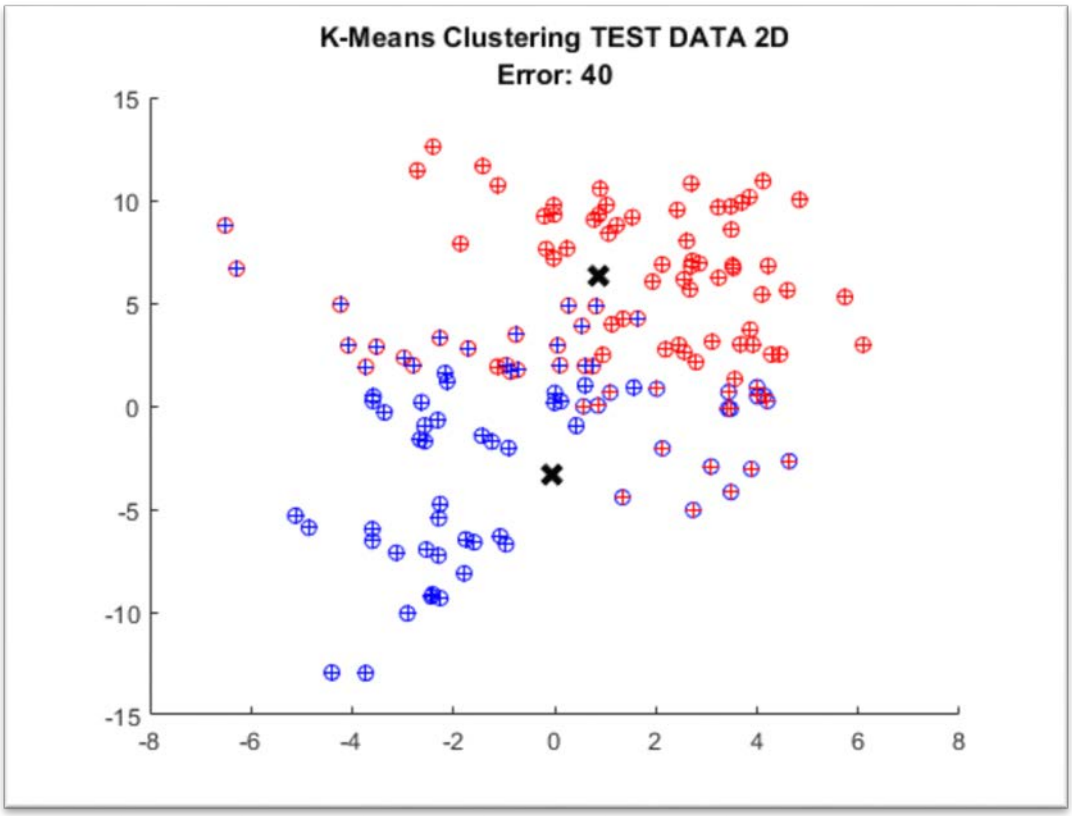

Figure 2-2

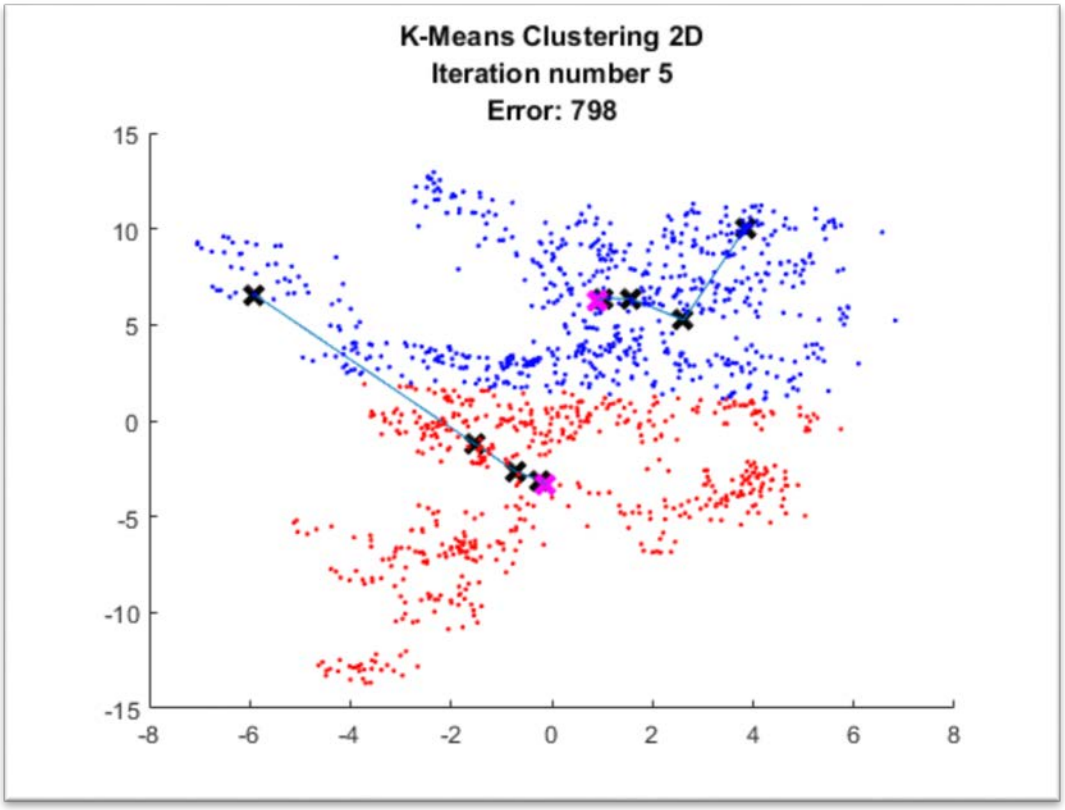

Figure 2-3

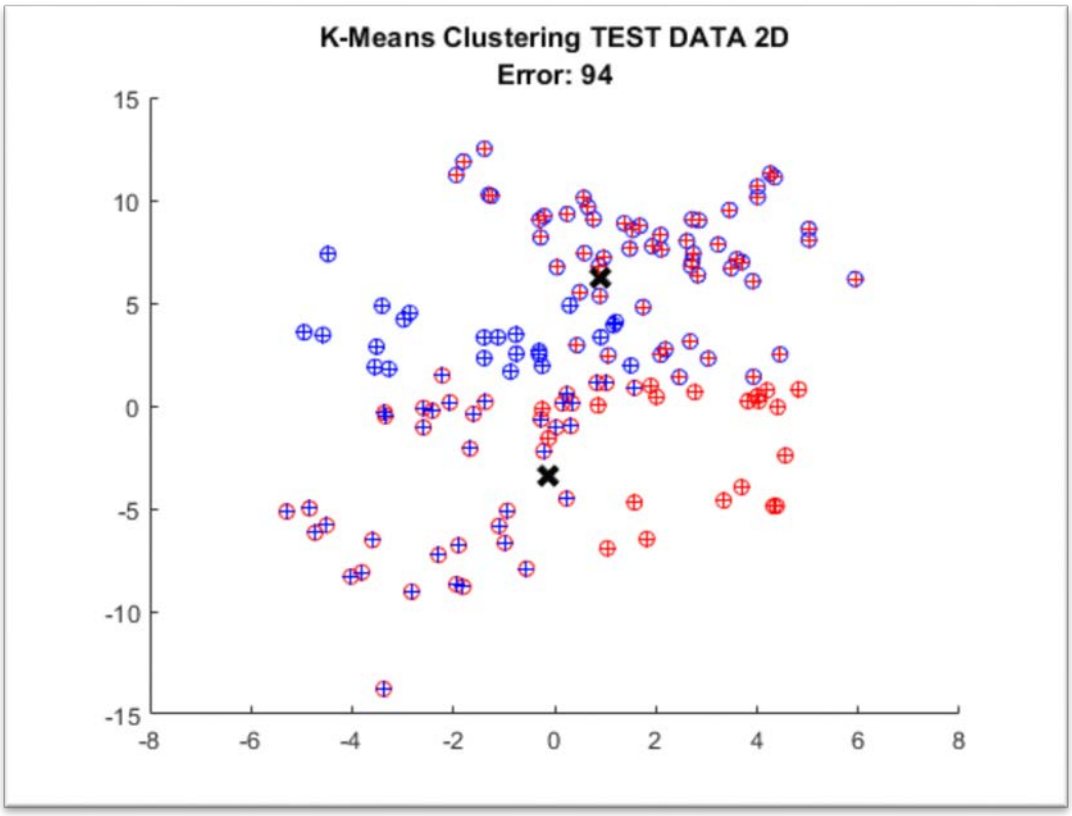

Figure 2-4

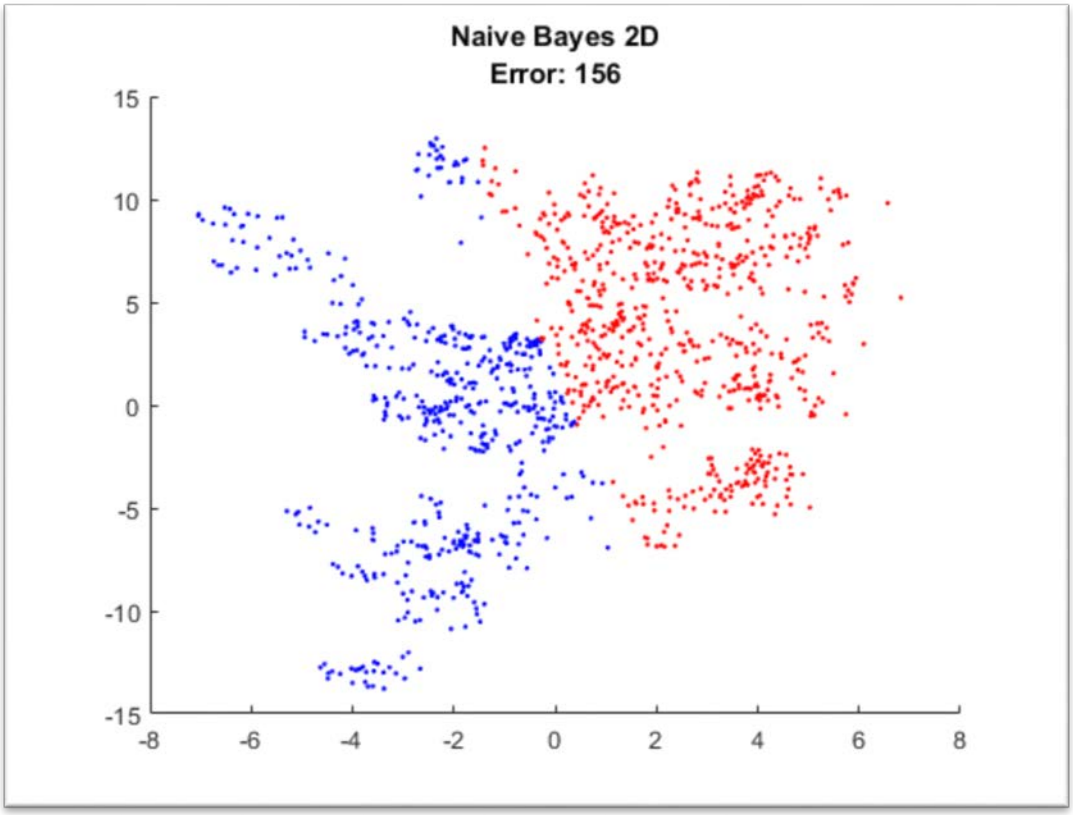

Figure 3-1

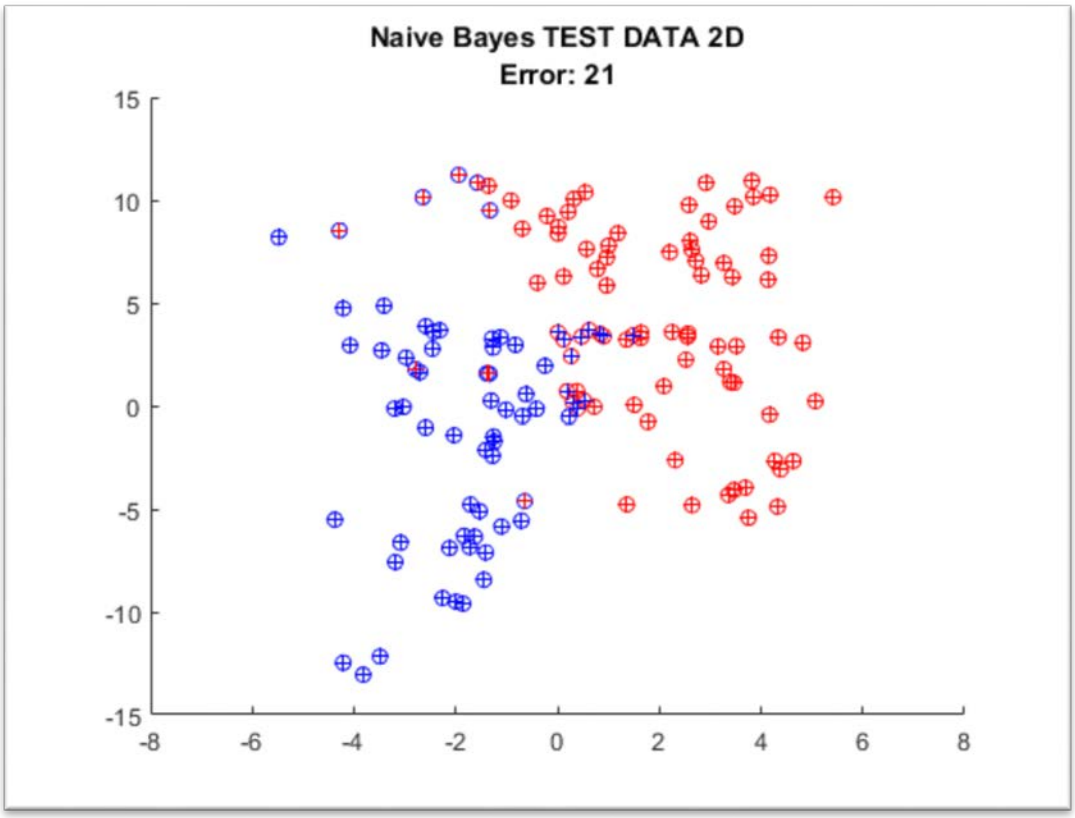

Figure 3-2

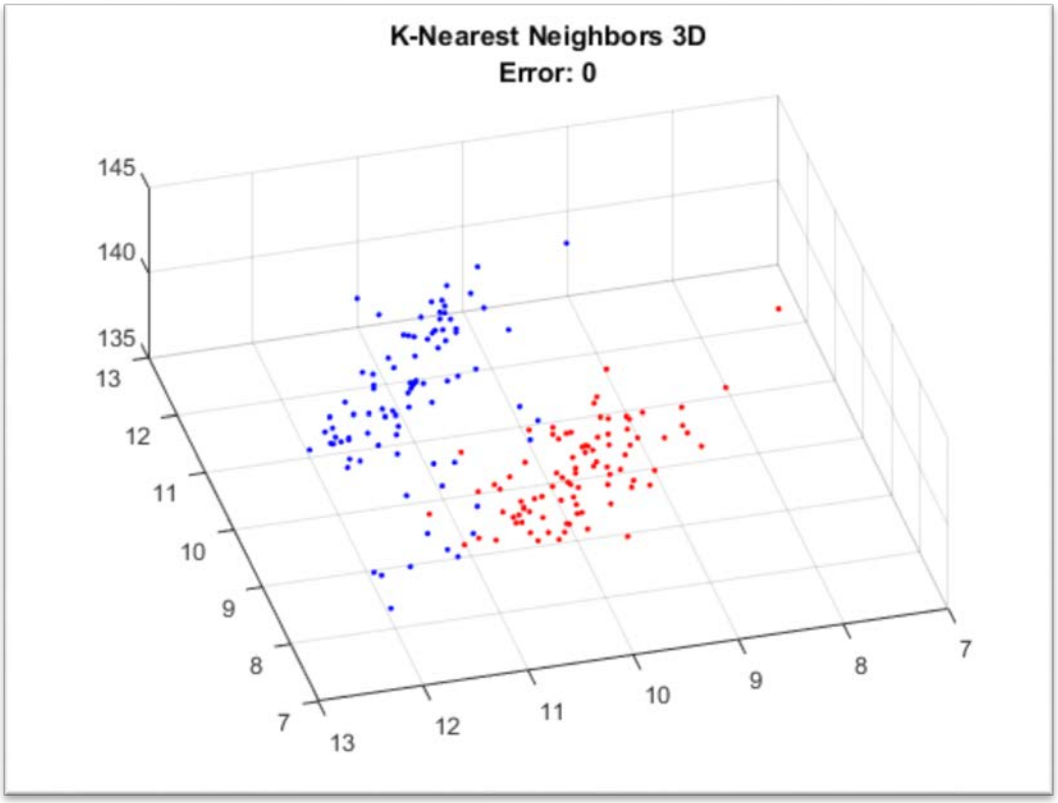

Figure 4-1

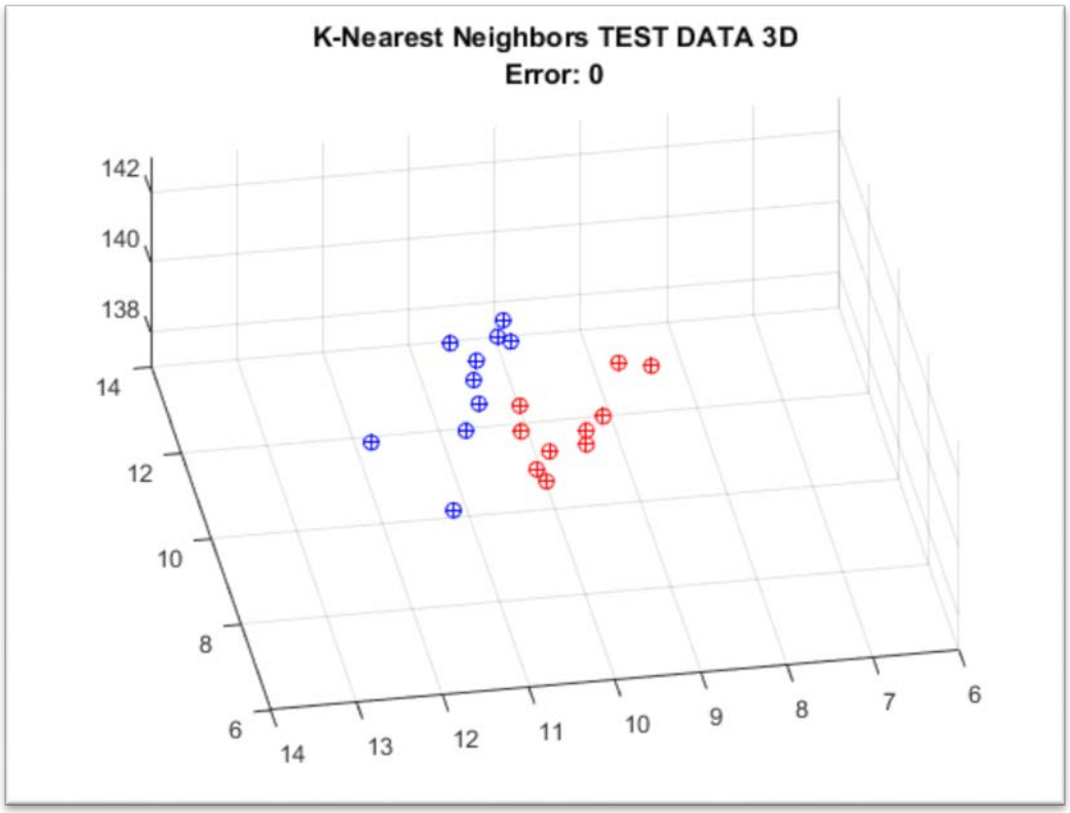

Figure 4-2

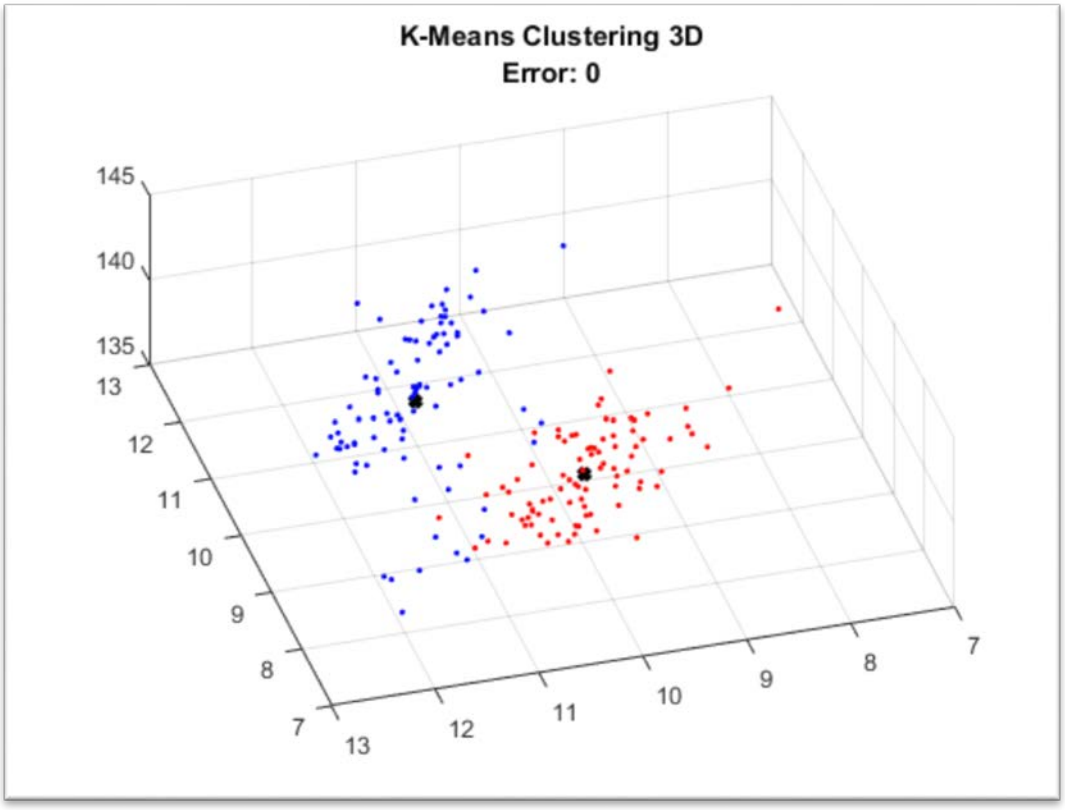

Figure 5-1

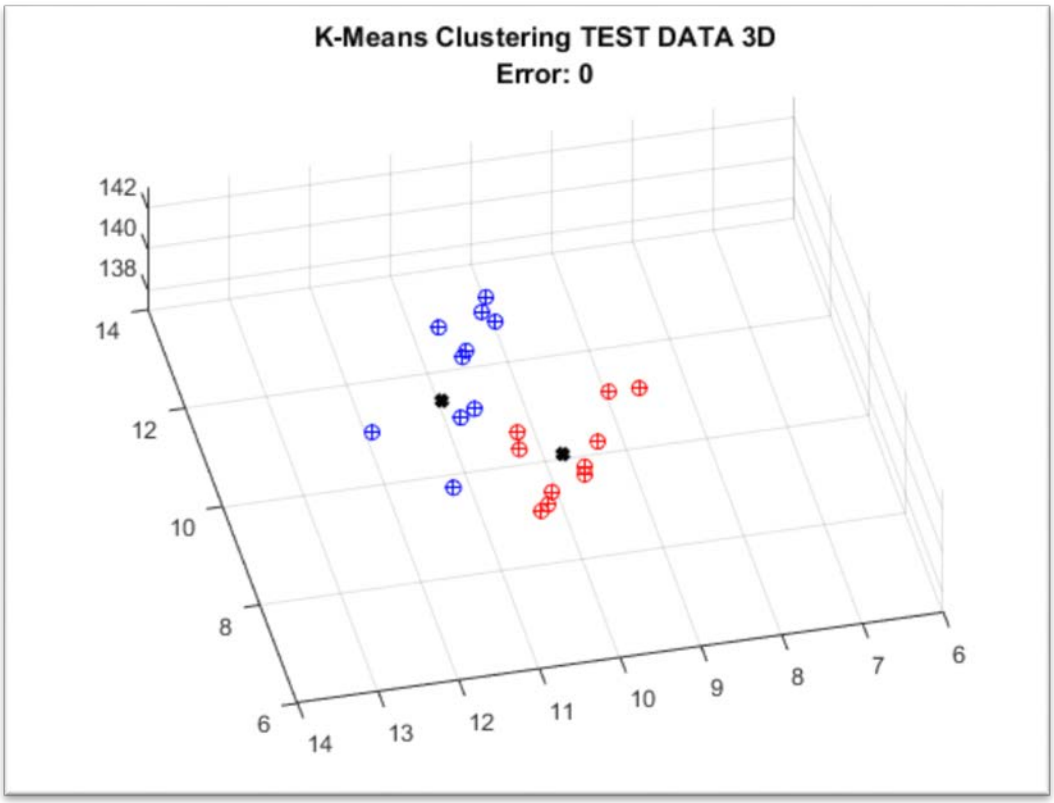

Figure 5-2

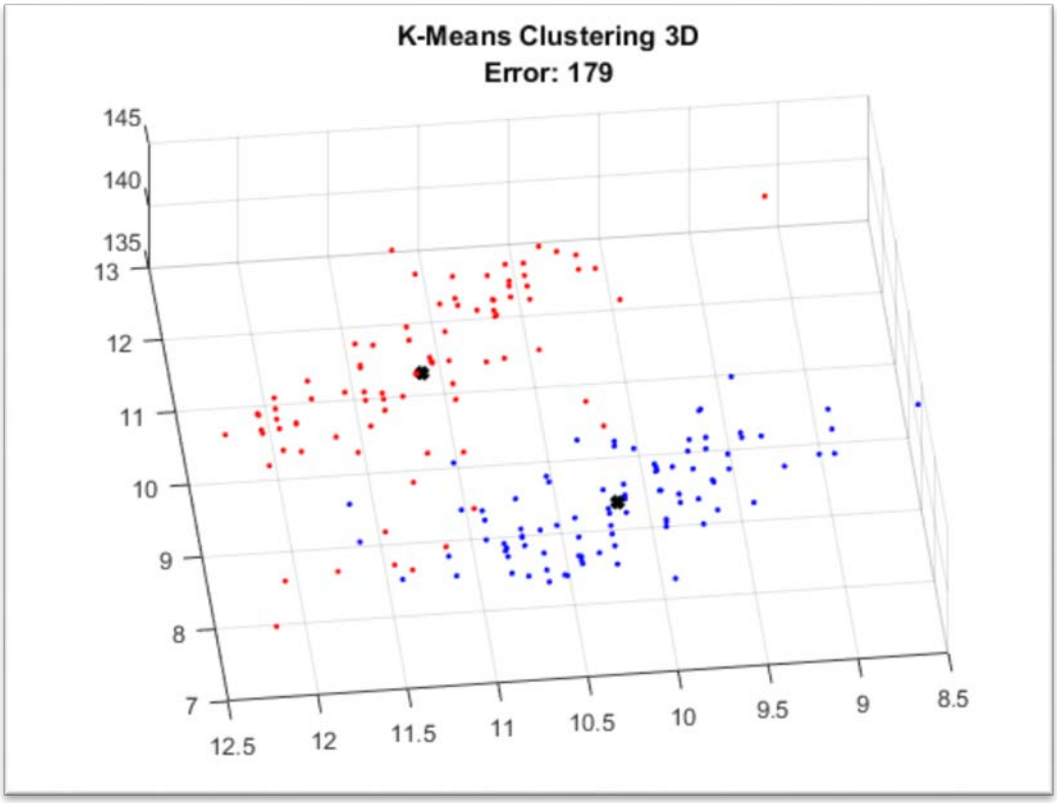

Figure 5-3

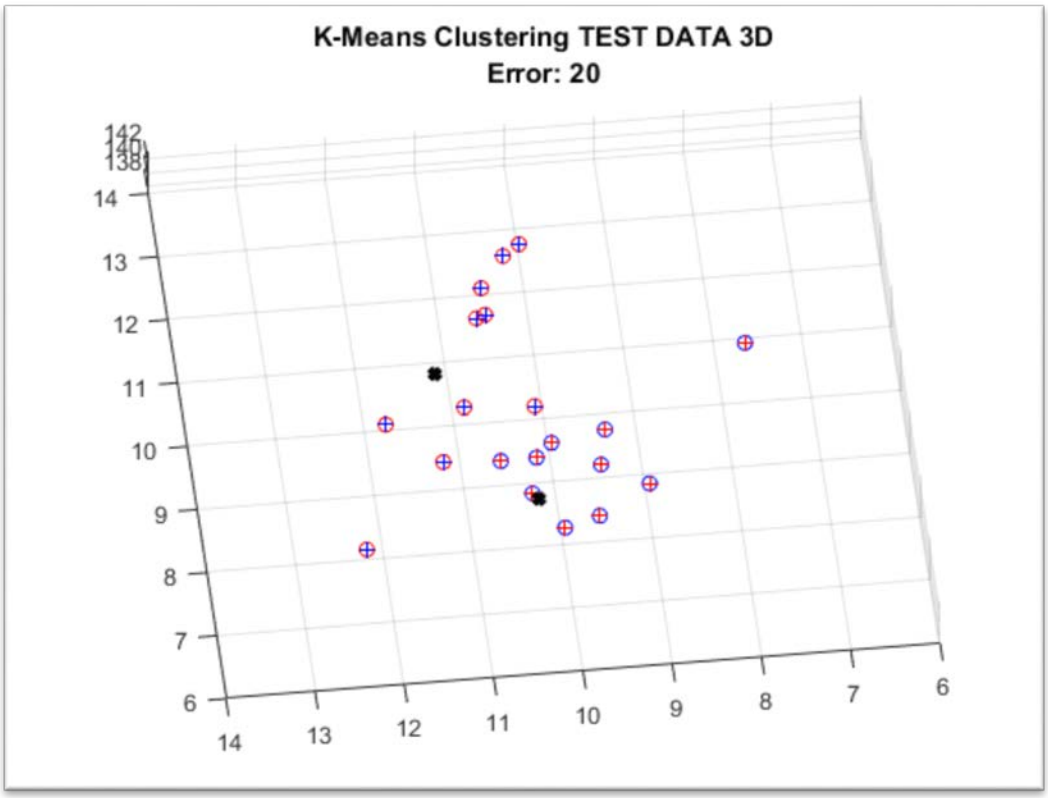

Figure 5-4

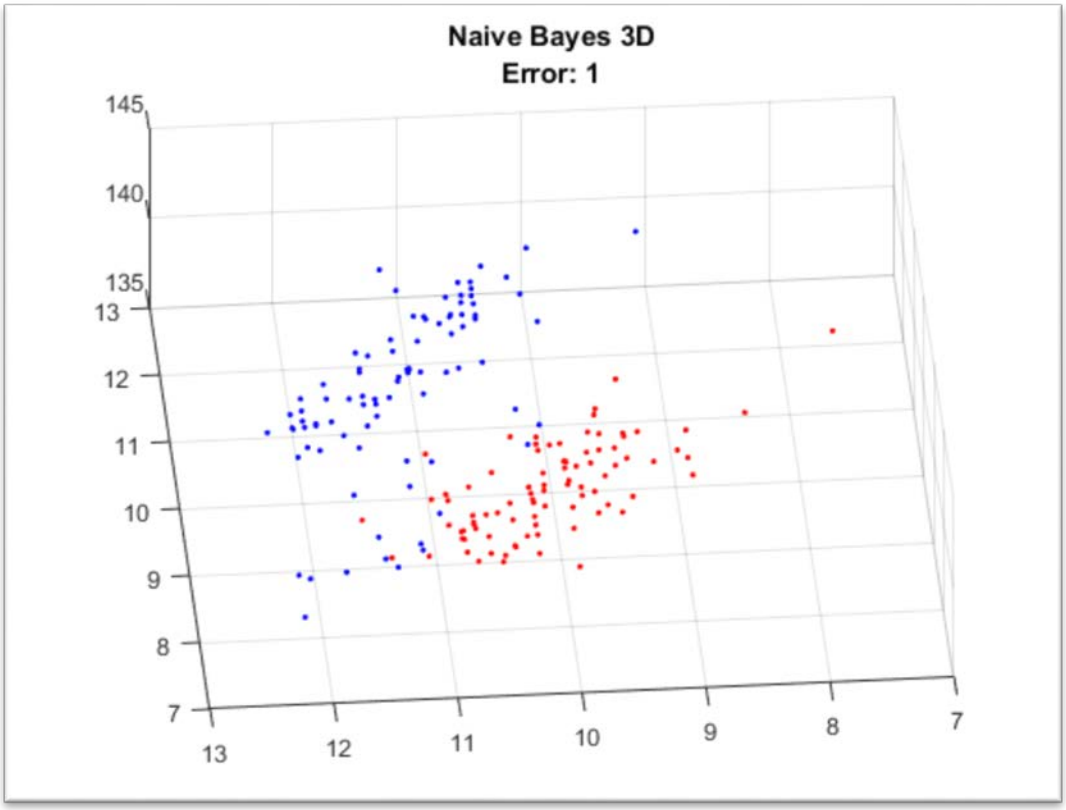

Figure 6-1

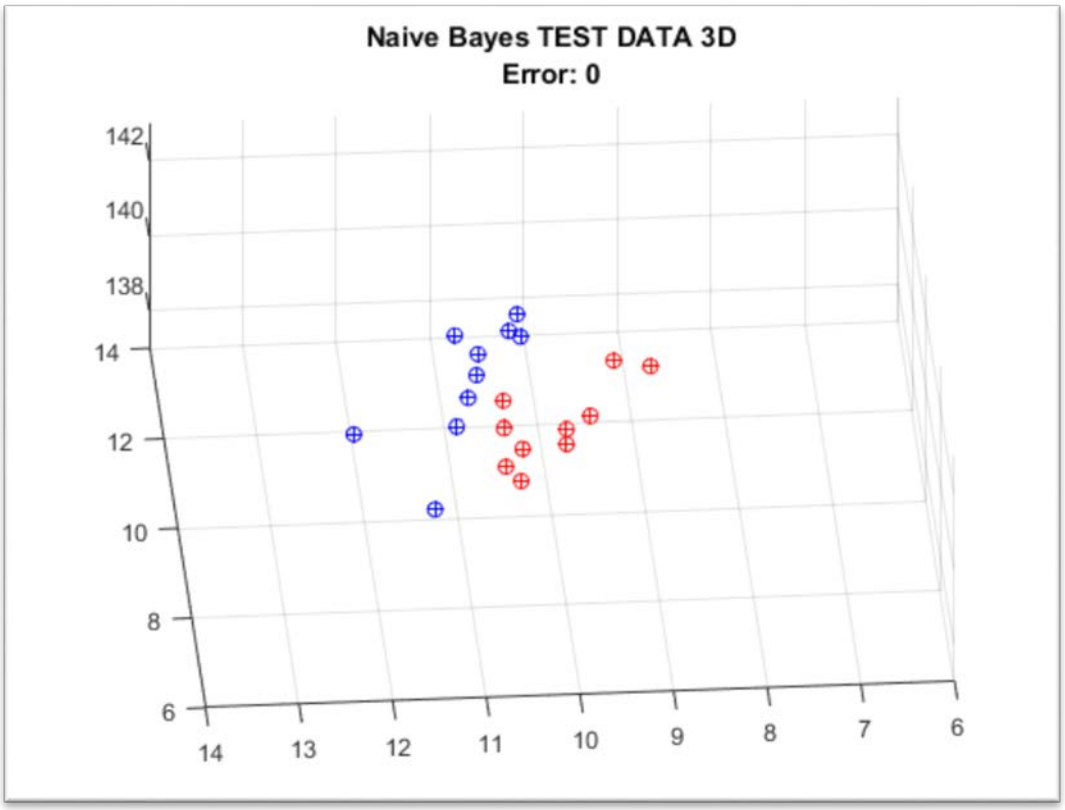

Figure 6-2