Week 13

Onur TEKINER

Introduction to Python SUM/23

7/27/23

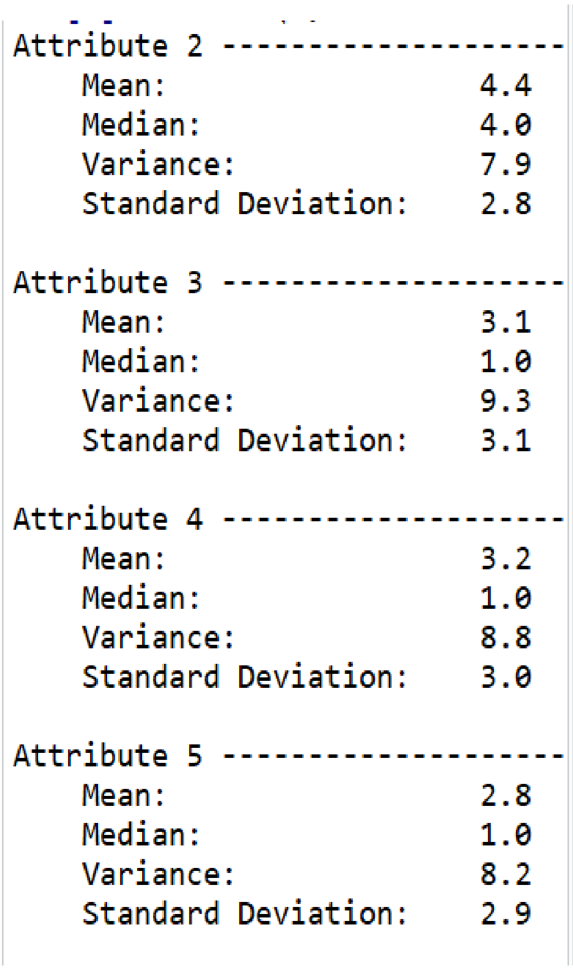
Final Project

Phase 1

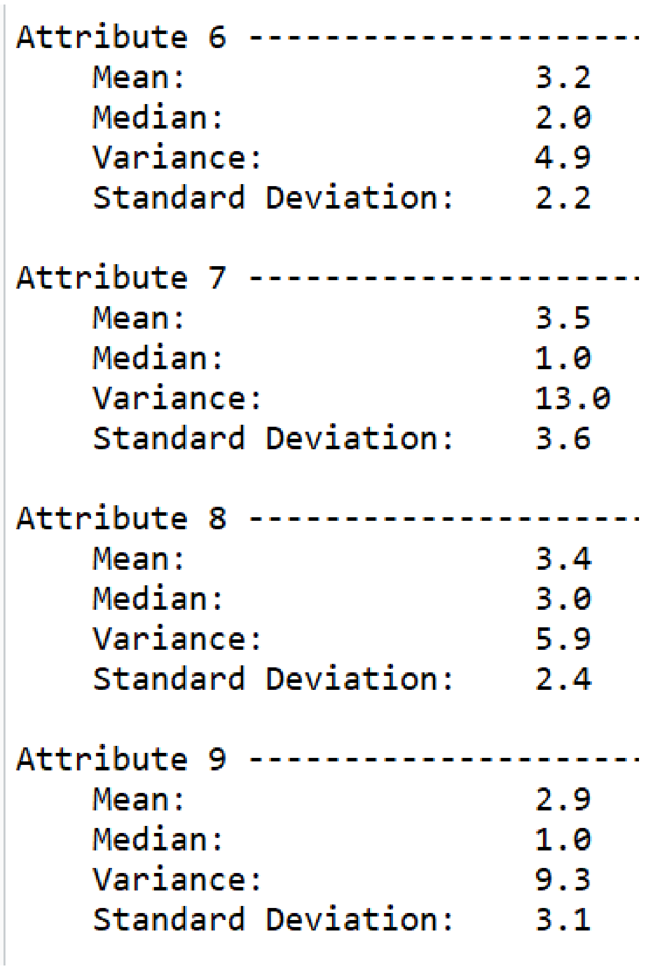
This dataset recorded clinical cases by Dr.Wolberg. There are 699 rows in the dataset. There are 11 columns;

- The scn column is a unique number for each patient
- A2 to A10 have values between 1-10
- The class column values are either 2 (benign) or 4 (malignant)

Let's see A2 to A10 columns' mean, median, variance, and standard deviation;



```
Attribute 2 --------------------
    Mean:                    4.4
    Median:                  4.0
    Variance:                7.9
    Standard Deviation:      2.8

Attribute 3 --------------------
    Mean:                    3.1
    Median:                  1.0
    Variance:                9.3
    Standard Deviation:      3.1

Attribute 4 --------------------
    Mean:                    3.2
    Median:                  1.0
    Variance:                8.8
    Standard Deviation:      3.0

Attribute 5 --------------------
    Mean:                    2.8
    Median:                  1.0
    Variance:                8.2
    Standard Deviation:      2.9
```

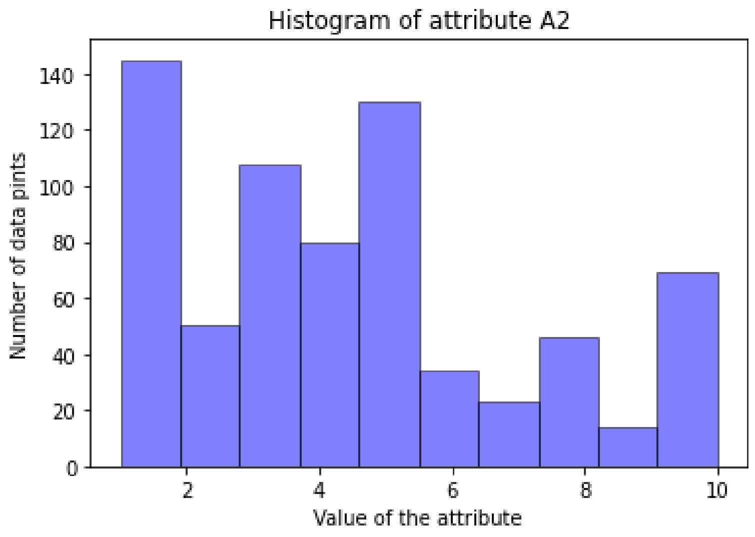


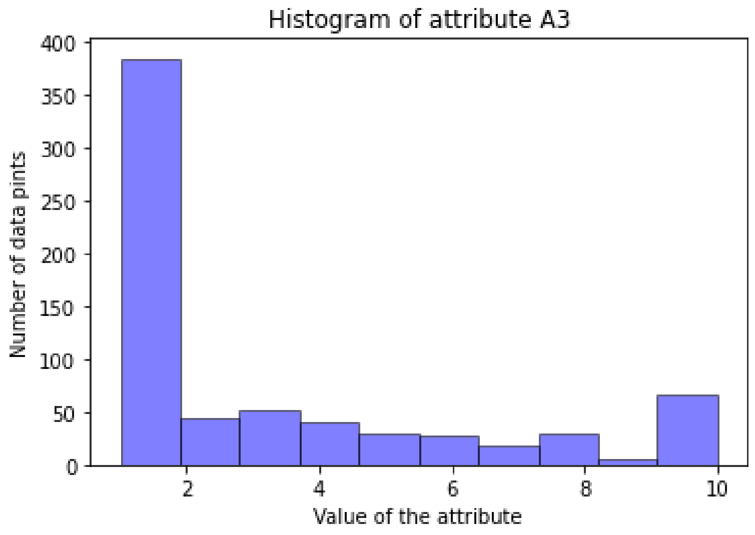
```
Attribute 6 --------------------
    Mean:                    3.2
    Median:                  2.0
    Variance:                4.9
    Standard Deviation:      2.2

Attribute 7 --------------------
    Mean:                    3.5
    Median:                  1.0
    Variance:                13.0
    Standard Deviation:      3.6

Attribute 8 --------------------
    Mean:                    3.4
    Median:                  3.0
    Variance:                5.9
    Standard Deviation:      2.4

Attribute 9 --------------------
    Mean:                    2.9
    Median:                  1.0
    Variance:                9.3
    Standard Deviation:      3.1
```

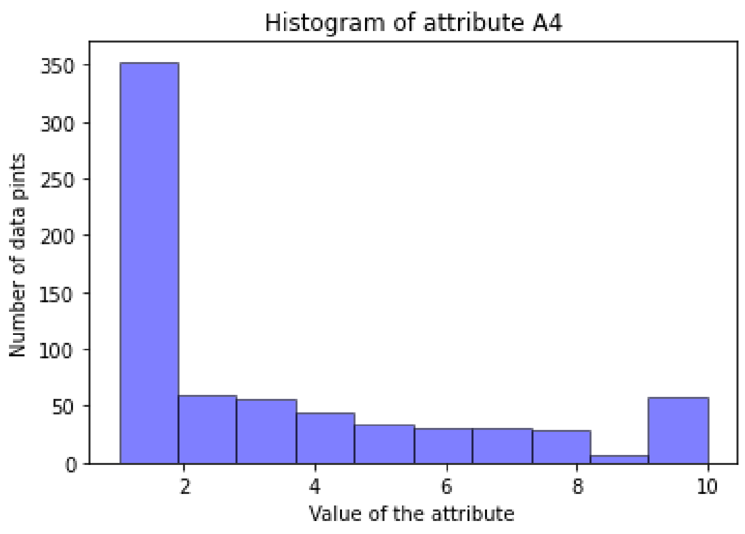
```
Attribute 10 --------------------
    Mean:                   1.6
    Median:                 1.0
    Variance:               2.9
    Standard Deviation:     1.7
```

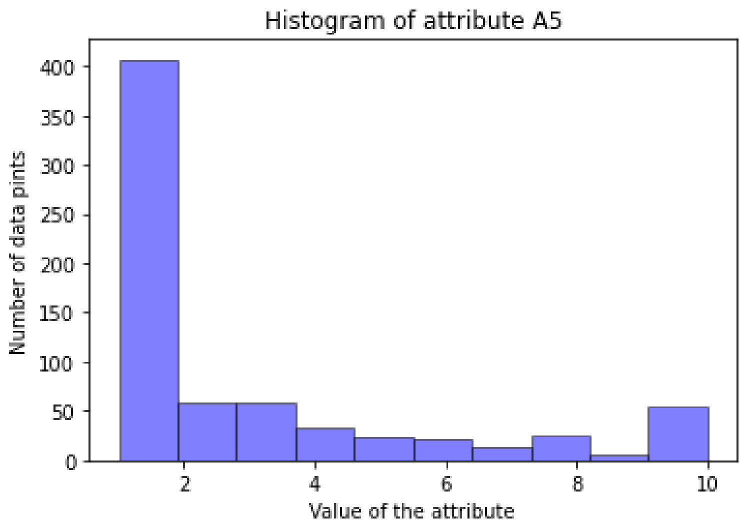
We see that most attributes' means are so much higher than their medians. Also, attribute 7 is so much higher variance than the other attributes' variances.

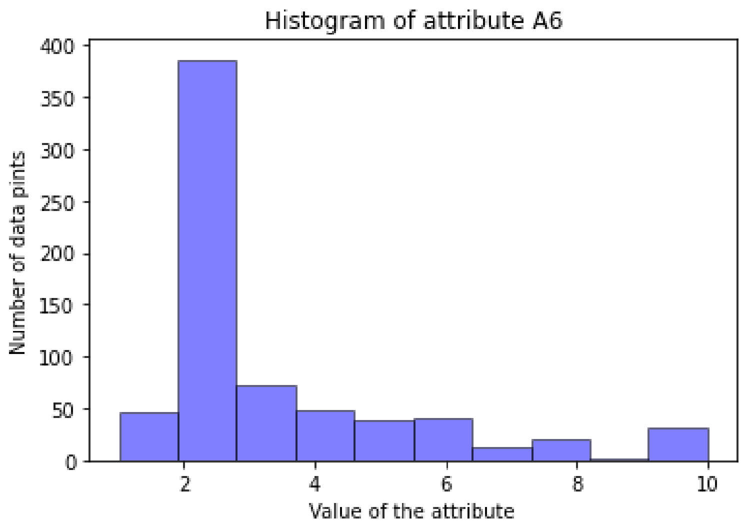
Let's see how the histogram graphs show the attributes.

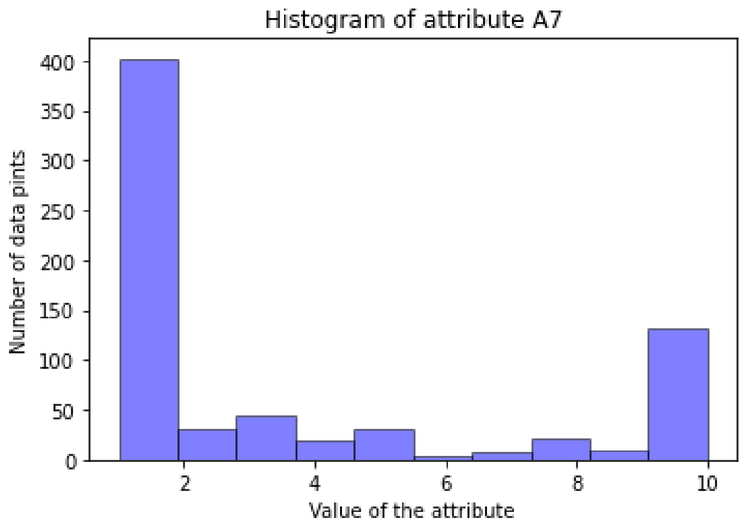


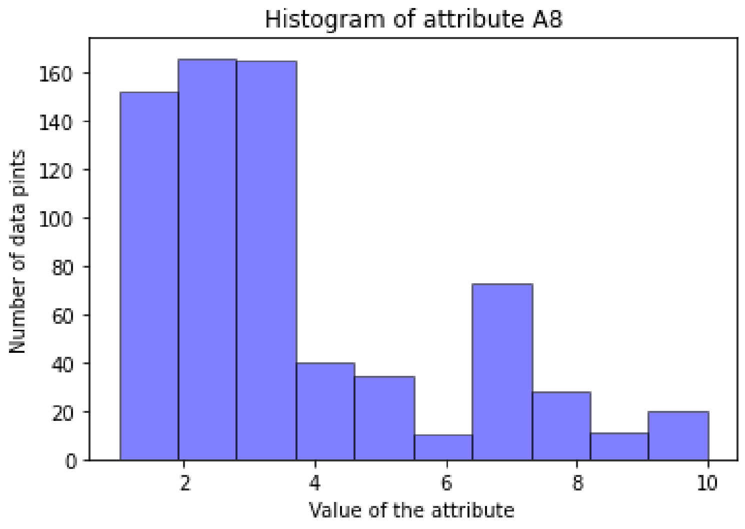


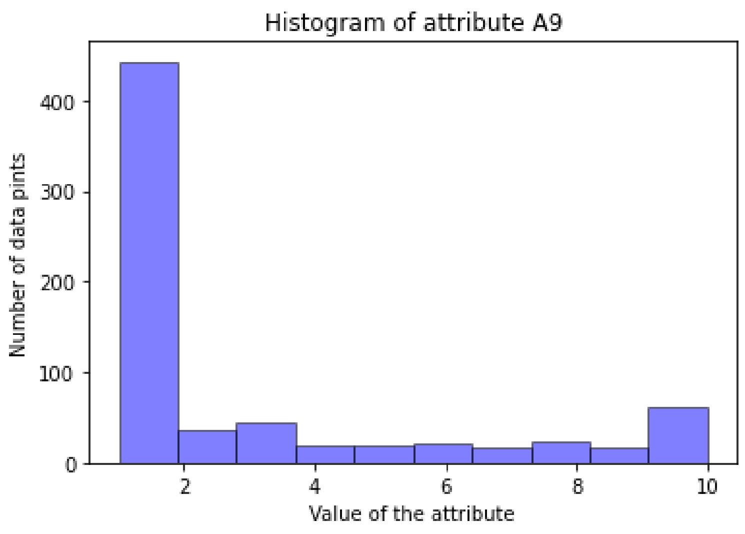

Histogram of attribute A4
Number of data pints
Value of the attribute
0
50
100
150
200
250
300
350
2
4
6
8
10

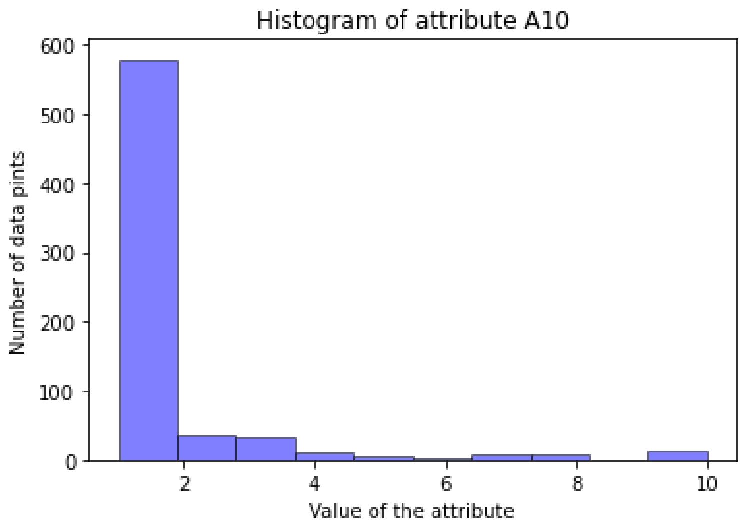

Histogram of attribute A5
Number of data pints
Value of the attribute
0
50
100
150
200
250
300
350
400
2
4
6
8
10


Histogram of attribute A6
Number of data pints
Value of the attribute
0
50
100
150
200
250
300
350
400
2
4
6
8
10


Histogram of attribute A7
Number of data pints
Value of the attribute
0
50
100
150
200
250
300
350
400
2
4
6
8
10


Histogram of attribute A8
Number of data pints
Value of the attribute
0
20
40
60
80
100
120
140
160
2
4
6
8
10





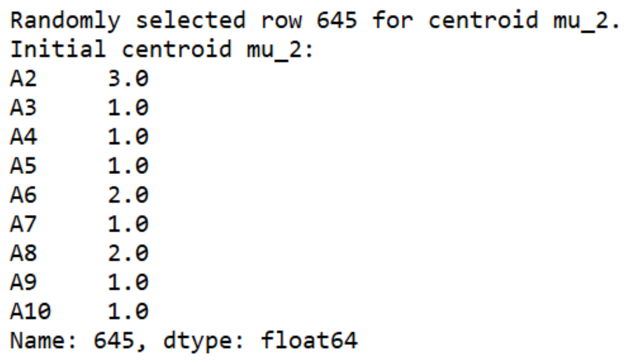
Summary of histograms;

- The attribute 2 values are more widely distributed beside the other attributes
- Every attribute is right-skewed
- The value of one is especially highly repeated in the attributes
- Attribute 6 contains mostly values of 2
- The value of 1 is very highly repeated in the attributes except attribute 6
- In attribute 8, the values of 2 and 3's density are higher than the values of 1

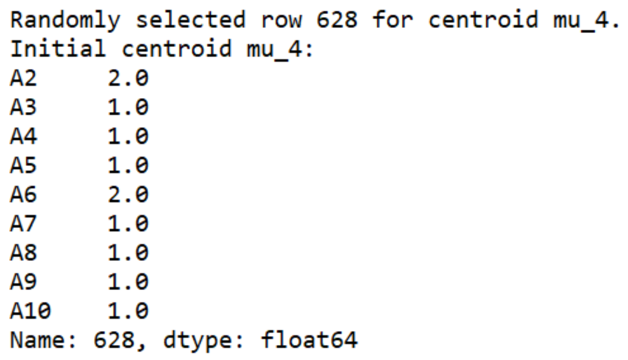
<u>Phase 2</u>

I am going to use the k-means clustering technique for predicting class. The number of clusters is two because the predicting columns have two class types, which are 2 and 4.

For initial centroids, I selected two random rows from the dataset. My first random row's index is 645, which is predicted with 2 (benign), and the second random row's index is 628, which is predicted with 4 (malignant).



```
Randomly selected row 645 for centroid mu_2.
Initial centroid mu_2:
A2     3.0
A3     1.0
A4     1.0
A5     1.0
A6     2.0
A7     1.0
A8     2.0
A9     1.0
A10    1.0
Name: 645, dtype: float64
```



```
Randomly selected row 628 for centroid mu_4.
Initial centroid mu_4:
A2     2.0
A3     1.0
A4     1.0
A5     1.0
A6     2.0
A7     1.0
A8     1.0
A9     1.0
A10    1.0
Name: 628, dtype: float64
```

According to these initial centroids, the rest of the rows assign predicted class, either 2 or 4, depending on their Euclidian distance. After every row predicted in the dataset, I calculated the mean of each current cluster for replacing initial centroids, then repeated the calculation of distances and re-assigned each row to clusters again. This calculation of new centroids and re-assigning predicted class continued until there was no data point switch in the clusters.

Our program iterated five times in this case to find the final centroids point of each cluster.

There are two updated centroid points below;

```
Program ended after 5 iterations.

Final centroid mu_2:
A2     7.173913
A3     6.800000
A4     6.734783
A5     5.739130
A6     5.478261
A7     7.930435
A8     6.108696
A9     6.039130
A10    2.569565
dtype: float64

Final centroid mu_4:
A2     3.071886
A3     1.374658
A4     1.500457
A5     1.427963
A6     2.144381
A7     1.393848
A8     2.142248
A9     1.338410
A10    1.195553
dtype: float64
```

First top 20 rows with unique patient numbers, class, and predicted class columns;

```
Final cluster assignment:

        Scn  Class  Predicted_Class
0   1000025      2                4
1   1002945      2                2
2   1015425      2                4
3   1016277      2                2
4   1017023      2                4
5   1017122      4                2
6   1018099      2                4
7   1018561      2                4
8   1033078      2                4
9   1033078      2                4
10  1035283      2                4
11  1036172      2                4
12  1041801      4                4
13  1043999      2                4
14  1044572      4                2
15  1047630      4                4
16  1048672      2                4
17  1049815      2                4
18  1050670      4                2
19  1050718      2                4
20  1054590      4                2
```

As we see from the top 20 rows, the predicted class didn't work well compared to the actual class. This kind of situation is a weak spot of the k-means clustering technique.

In the next phase, I will solve this issue by swapping the classes.

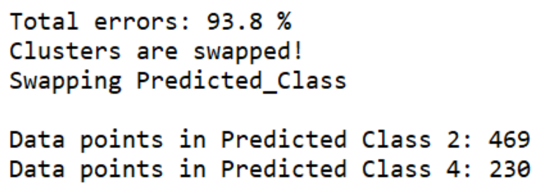
Phase 3

Now we have predicted class and real class. So, we can calculate how successful our cluster technique is by comparing the predicted class to the actual class.

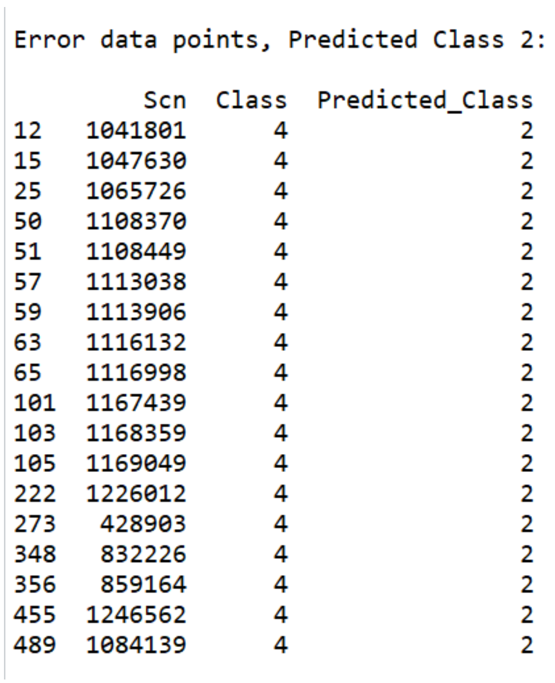
The program picks two random numbers without knowing their actual classes. The k-means clustering technique is good for separating data points from each other, but it is not good for assigning the first two (there are only two classes in this dataset) numbers to clusters.

In the case of the program predicting an actual benign cluster for a cluster of malignant or vice versa, the program will swap the classes. So, when the program calculates that the predicted class gives an error of more than fifty percent, the program will switch classes 2 and 4 to 4 and 2, then compare the updated predicted class with the actual class again.

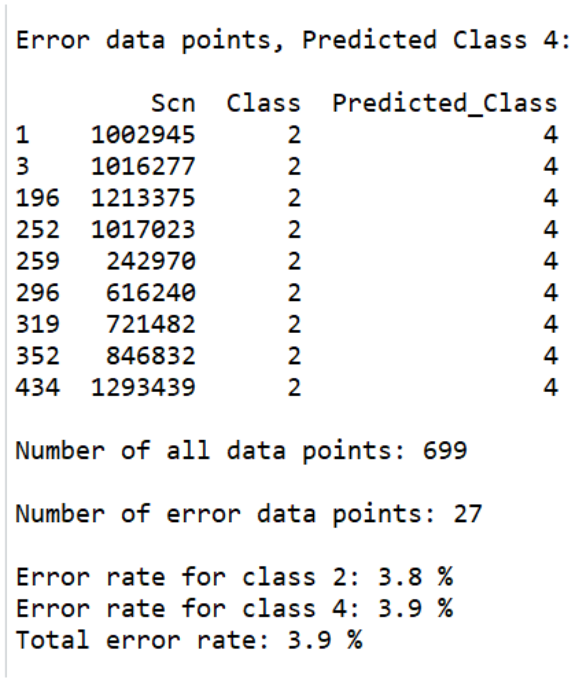
Now let's see the total results of our k-means clustering technique and how well it works in our dataset;



```
Total errors: 93.8 %
Clusters are swapped!
Swapping Predicted_Class

Data points in Predicted Class 2: 469
Data points in Predicted Class 4: 230
```



```
Error data points, Predicted Class 2:

         Scn  Class  Predicted_Class
12   1041801      4                2
15   1047630      4                2
25   1065726      4                2
50   1108370      4                2
51   1108449      4                2
57   1113038      4                2
59   1113906      4                2
63   1116132      4                2
65   1116998      4                2
101  1167439      4                2
103  1168359      4                2
105  1169049      4                2
222  1226012      4                2
273   428903      4                2
348   832226      4                2
356   859164      4                2
455  1246562      4                2
489  1084139      4                2
```



```
Error data points, Predicted Class 4:

         Scn  Class  Predicted_Class
1    1002945      2                4
3    1016277      2                4
196  1213375      2                4
252  1017023      2                4
259   242970      2                4
296   616240      2                4
319   721482      2                4
352   846832      2                4
434  1293439      2                4

Number of all data points: 699

Number of error data points: 27

Error rate for class 2: 3.8 %
Error rate for class 4: 3.9 %
Total error rate: 3.9 %
```

As we see in our results:

- Classes are swapped with each other to find the correct cluster for prediction. The total error is 3.9 percent
- The total number of errors is only 27 out of 699
- The program's total prediction of malignant is 230 in the dataset, and the wrong prediction is only 9. The prediction of malignant error rate is 3.8 percent

- The program's total prediction of benign is 469 in the dataset, and the wrong prediction is only 18. The prediction of benign error rate is 3.9 percent

It is an extraordinarily successful classification. The k-means clustering technique does work in this dataset.

<u>Python’s programs and algorithms descriptions;</u>

Phase 1

- **read_to_pandas():** upload the data in the pandas from the CPU
- **fill(df):** input value is the data frame. The program fills null values with the mean of the column in the input data frame
- **stat_data():** select only A2 to A10 columns excluding unique patient number and class column
- **grap(df):** making histograms of selected columns
- **main():** run every program properly and bring the results

Phase 2

- **read_to_pandas()**
- **fill(df)**
- **initial_centroids():** selecting two random numbers for initial centroids
- **distance(row,mu_2,mu_4):** calculating Euclidian distance of “row” to “mu_2” and “mu_4”. If the distance is closer to “mu_2”, returns 2; otherwise, returns 4
- **main():** run every program in it. Print outputs and calculate iterations

Phase3

- **read_to_pandas()**
- **fill(df)**
- **initial_centroids()**
- **distance(row,mu_2,mu_4)**
- **finding_stats(final_df):**final_df has 3 columns; Scn, class, and predicted class. This program calculates all types of errors
- **main():** run every program in it. Calculate new centroids and assign row classifications if needed. Print the results