Name:Muhammad Hamza Ahmad

Roll\_No:21L-6292

**Question1:**

* Firstly the column 'blood\_glucose\_level' is normalized using MinMaxScaler , this steps reduces error in further calculations
* Gender and Smoking\_history columns are Non\_numerical categorcal data , label encoder is used to Convert that data in to numeric categorical data.this step was necessary so that these columns can be used for analyzing data

**Question2**:

In the second Question we have to reduce the features , I have used PCA and have reduced the features to 4

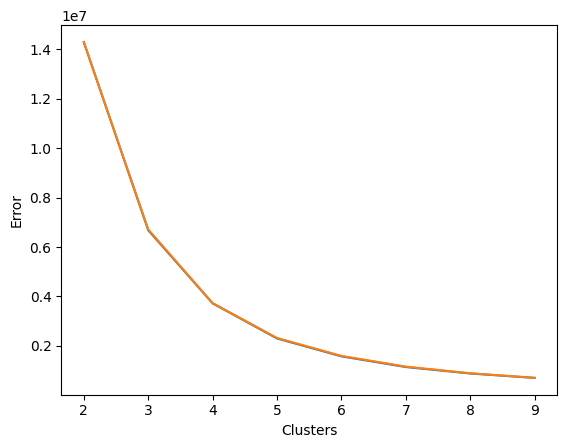
**Question3:**

Note:There are several parts in this questions which are actually one step ahead of the previous part , I have done all the parts in a single loop which is ecplained as follows :

K\_means algorithms is used for clustering in the first step we just have to find the best value of k where is the number of clusters , I have checked clusters between 2 and 10 ,

The error found for each k are follows: (The error is WSS)

|  |  |
| --- | --- |
| 2 | 6676571.59456394 |
| 3 | 3717750.989408488 |
| 4 | 2296570.832372234 |
| 5 | 1573942.561276988 |
| 6 | 1141410.758348758 |
| 7 | 879954.5061149084 |
| 8 | 701711.3639338909 |
| 9 | 14286242.50841634 |



For different values of K , K\_means is running multiple times and their report in getting stored in lists which are further made in to a dataframe

The K\_means is used on 2 distinct subset of the reduced dataset, total there are 4 columns and each subset have 2 columns . Their reports are stored in different lists so it can we compared afterwards for the best value ok k.

df1 = df.iloc[:, :2]

df2 = df.iloc[2:, :4]

Ths is how the dataframe is devided into two subsets.

To find the best value of K which runs perfectly in both subsets I have used

2 functions wss and bss and stored each of the value for every k in a their redpected lists using append() and at the end comparison is done to see which value of k gives the best result.

The following table shows the wss ,bss and iterations for each value of k and for both subsets .

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **k** | **wss** | **wss2** | **bss** | **bss2** | **iterations** | iteration2 |
| 0 | 2 | 1.428624e+07 | 1.429604e+07 | -1.428624e+07 | -1.429604e+07 | 5 | 7 |
| 1 | 3 | 6.702704e+06 | 6.689592e+06 | -6.702704e+06 | -6.689592e+06 | 12 | 4 |
| 2 | 4 | 3.713331e+06 | 3.721662e+06 | -3.713331e+06 | -3.721662e+06 | 5 | 3 |
| 3 | 5 | 2.328861e+06 | 2.316609e+06 | -2.328861e+06 | -2.316609e+06 | 7 | 5 |
| 4 | 6 | 1.581127e+06 | 1.594413e+06 | -1.581127e+06 | -1.594413e+06 | 4 | 4 |
| 5 | 7 | 1.148521e+06 | 1.162449e+06 | -1.148521e+06 | -1.162449e+06 | 4 | 8 |
| 6 | 8 | 8.759592e+05 | 8.962348e+05 | -8.759592e+05 | -8.962348e+05 | 3 | 5 |
| 7 | 9 | 7.089622e+05 | 7.076327e+05 | -7.089622e+05 | -7.076327e+05 | 6 | 7 |

Code:

kmeans1=KMeans(n\_clusters=k,max\_iter=100)

kmeans1.fit(df1)

wss\_values1.append(kmeans1.inertia\_)

kmeans2=KMeans(n\_clusters=k,max\_iter=100)

kmeans2.fit(df2)

wss\_values2.append(kmeans2.inertia\_)

bss\_values1.append(kmeans1.score(df1))

bss\_values2.append(kmeans2.score(df2))

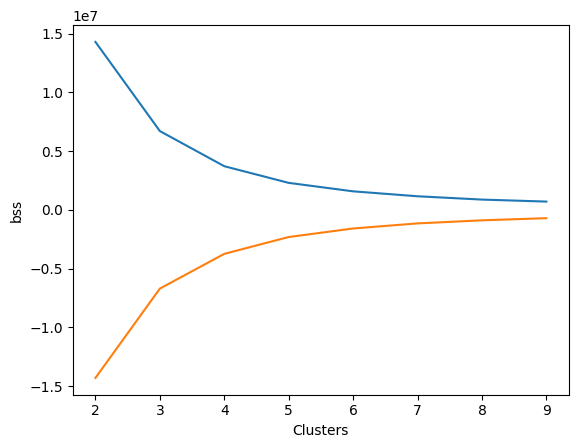
k\_values.append(k)

n\_iterations1 = kmeans1.n\_iter\_

n\_iterations2 = kmeans2.n\_iter\_

iteration\_values1.append(n\_iterations1)

iteration\_values2.append(n\_iterations2)



**Queston 4**:

In this Question I have simply used KNN algorithm using prebuild functions as knn=KNeighborsClassifier()

**Confusion Matrix:**

[[18170 144]

[ 773 913]]

**Classification\_Report:**

precision recall f1-score support

0 0.96 0.99 0.97 18274

1 0.88 0.53 0.66 1726

accuracy 0.95 20000

macro avg 0.92 0.76 0.82 20000

W\_avg 0.95 0.95 0.95 20000

**Question 5:**

