Final Project – I590 – PYTHON PROGRAMMING

Title – Prediction on Wisconsin Breast Cancer Dataset

Author -

Name - Arijit Sinha
Dept. - Data Science, School of Informatics, Computing, and Engineering, Indiana University
Email – arisinha@iu.edu

Table of contents –

Abstract –

Implement "k-means" algorithm for Wisconsin Breast Cancer data using Python. K-Means Clustering is one of the popular clustering algorithm. The goal of this algorithm is to find groups (clusters) in the given data.

Introduction -

Breast cancer is a rising issue among women. A cancer's stage is a crucial factor in deciding what treatment options to recommend, and in determining the patient's prognosis. Today, in the United States, approximately one in eight women over their lifetime has a risk of developing breast cancer. An analysis of the most recent data has shown that the survival rate is 88% after 5 years of diagnosis and 80% after 10 years of diagnosis. With early detection and treatment, it is possible that this type of cancer will go into remission. In such a case, the worse fear of a cancer patient is the recurrence of the cancer.

With K-Means algorithm, we will clustering and predicting on Wisconsin Breast Cancer Data. Below are the details on the dataset, Dr. Wolberg reports his clinical cases. There are total 11 columns or features in this dataset. Column/Attribute Information:

Number of Instances: 699

Number of Attributes: 10 and 1 Class Attribute

Number of classes: 2

Class Description	Description	# of Cases
2	Benign	458
4	Malignant	241

Features Description -

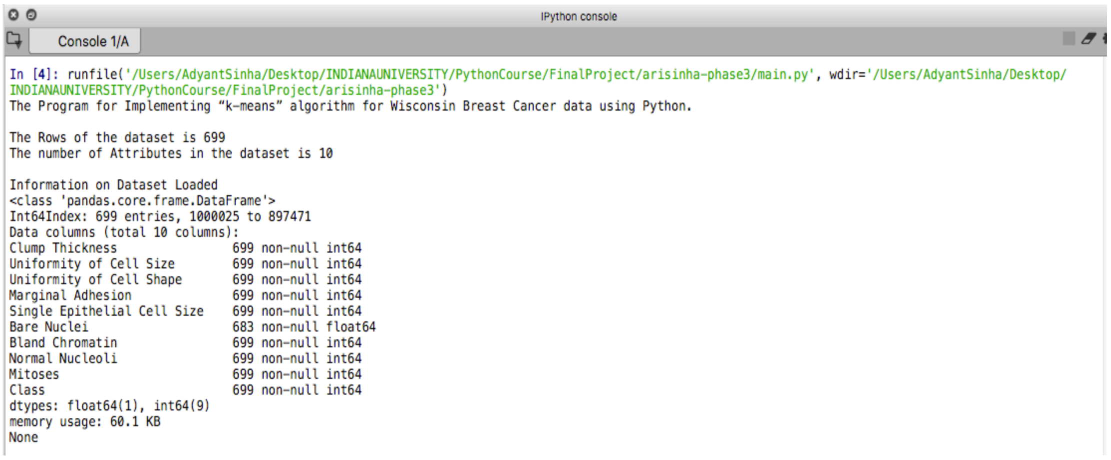
1. Sample code number: id number
2. Clump Thickness: 1 - 10
3. Uniformity of Cell Size: 1 - 10
4. Uniformity of Cell Shape: 1 - 10
5. Marginal Adhesion: 1 - 10
6. Single Epithelial Cell Size: 1 - 10
7. Bare Nuclei: 1 - 10 8. Bland Chromatin: 1 - 10
8. Normal Nucleoli: 1 - 10
9. Mitoses: 1 - 10
10. Class: (2 for benign, 4 for malignant)

Code Implementation and Results -

There are 3 phases which has been conducted to predict the class for the 699 observations.

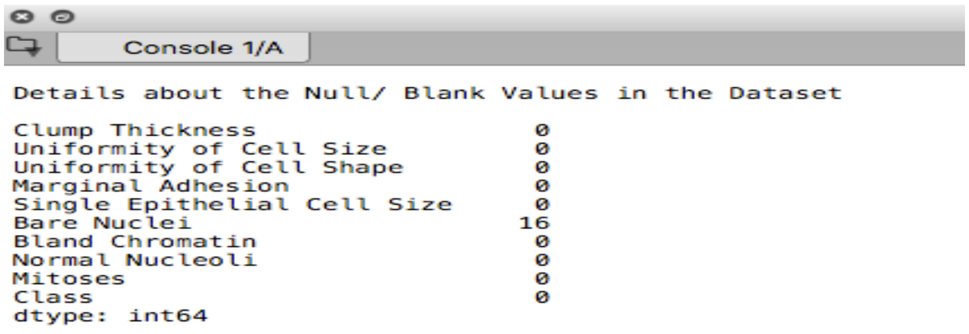
Phase 1 – Data Cleaning and Statistics from the attributes

1. Download the data and load it in Python 3
    a. We have downloaded the data file from url - https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancerwisconsin.data



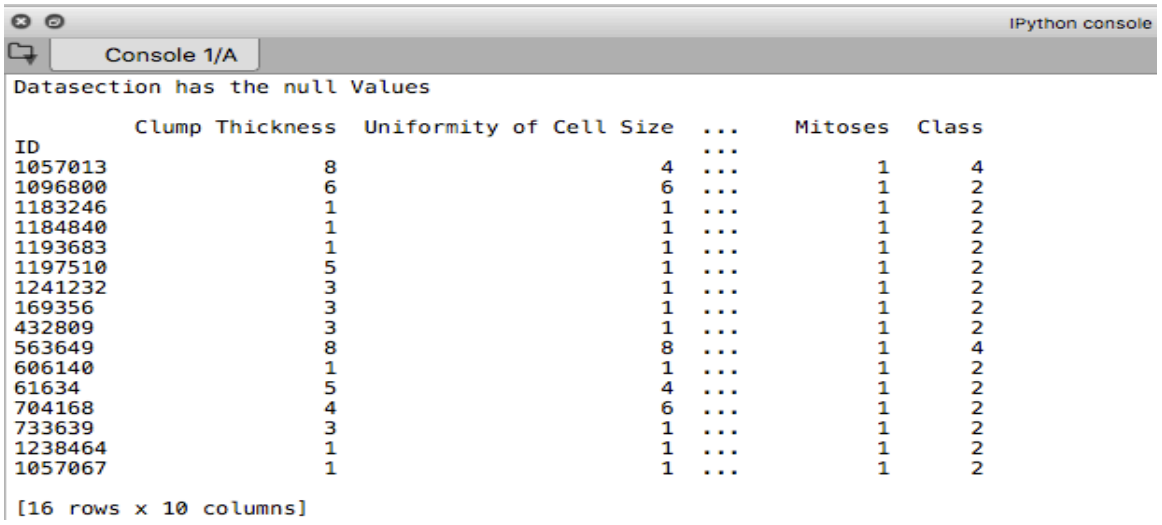
```
In [4]: runfile('/Users/AdyantSinha/Desktop/INDIANAUNIVERSITY/PythonCourse/FinalProject/arisinha-phase3/main.py', wdir='/Users/AdyantSinha/Desktop/
INDIANAUNIVERSITY/PythonCourse/FinalProject/arisinha-phase3')
The Program for Implementing “k-means” algorithm for Wisconsin Breast Cancer data using Python.

The Rows of the dataset is 699
The number of Attributes in the dataset is 10

Information on Dataset Loaded
<class 'pandas.core.frame.DataFrame'>
Int64Index: 699 entries, 1000025 to 897471
Data columns (total 10 columns):
Clump Thickness                699 non-null int64
Uniformity of Cell Size        699 non-null int64
Uniformity of Cell Shape       699 non-null int64
Marginal Adhesion              699 non-null int64
Single Epithelial Cell Size    699 non-null int64
Bare Nuclei                    683 non-null float64
Bland Chromatin                699 non-null int64
Normal Nucleoli                699 non-null int64
Mitoses                        699 non-null int64
Class                          699 non-null int64
dtypes: float64(1), int64(9)
memory usage: 60.1 KB
None
```

2. Impute missing values
    a. Upon analyzing the data, there are 16 records in Bare Nuclei which is having “?”, we have used the mean of the non-null records from the column to replace the “?” with mean value of 3.54.



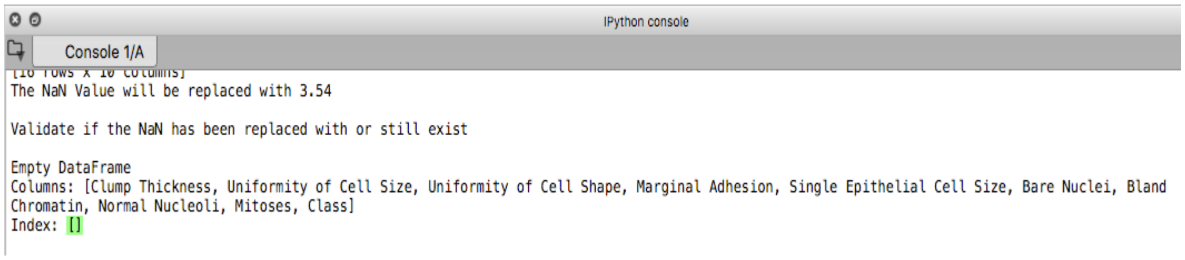
```
Details about the Null/ Blank Values in the Dataset

Clump Thickness                 0
Uniformity of Cell Size         0
Uniformity of Cell Shape        0
Marginal Adhesion               0
Single Epithelial Cell Size     0
Bare Nuclei                    16
Bland Chromatin                 0
Normal Nucleoli                 0
Mitoses                         0
Class                           0
dtype: int64
```

Detail record from the dataset having Null Values - Upon investigation, Column A7 (Bare Nuclei) has 16 Null Values for ID's - 1057013, 1096800, 1183246, 1184840, 1193683, 1197510, 1241232, 169356, 432809, 563649, 606140, 61634, 704168, 733639 ,1238464 ,1057067.



```
Datasection has the null Values

         Clump Thickness  Uniformity of Cell Size  ...   Mitoses  Class
ID                                                 ...
1057013                8                        4  ...         1      4
1096800                6                        6  ...         1      2
1183246                1                        1  ...         1      2
1184840                1                        1  ...         1      2
1193683                1                        1  ...         1      2
1197510                5                        1  ...         1      2
1241232                3                        1  ...         1      2
169356                 3                        1  ...         1      2
432809                 3                        1  ...         1      2
563649                 8                        8  ...         1      4
606140                 1                        1  ...         1      2
61634                  5                        4  ...         1      2
704168                 4                        6  ...         1      2
733639                 3                        1  ...         1      2
1238464                1                        1  ...         1      2
1057067                1                        1  ...         1      2

[16 rows x 10 columns]
```

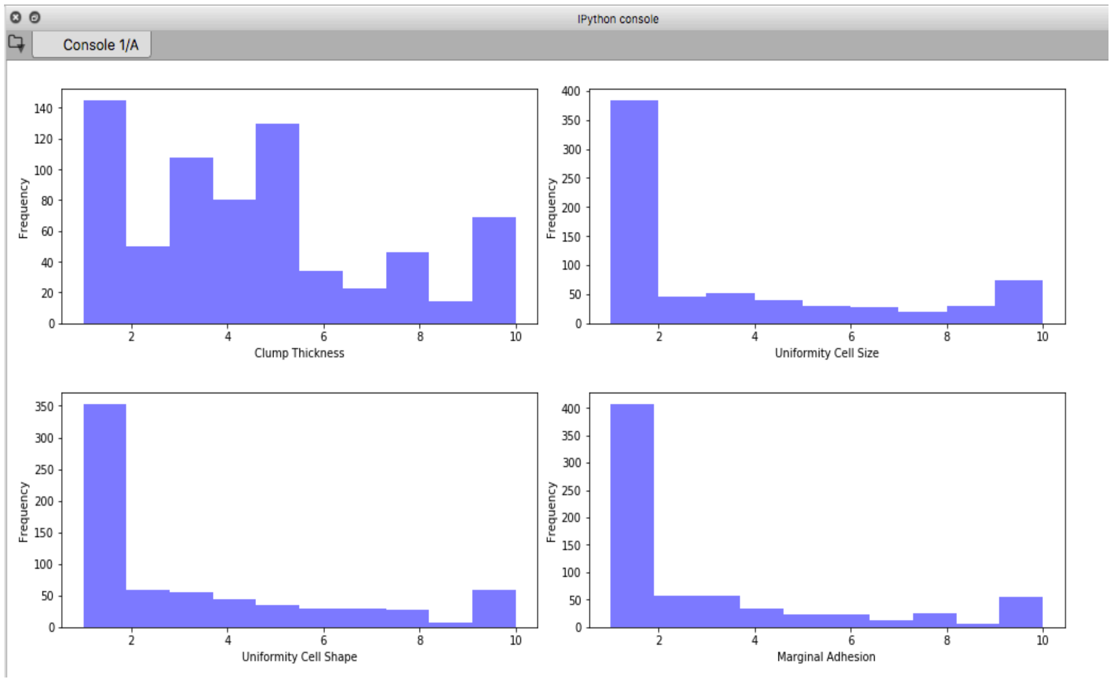
After Fixing the null values with replacing with mean value –

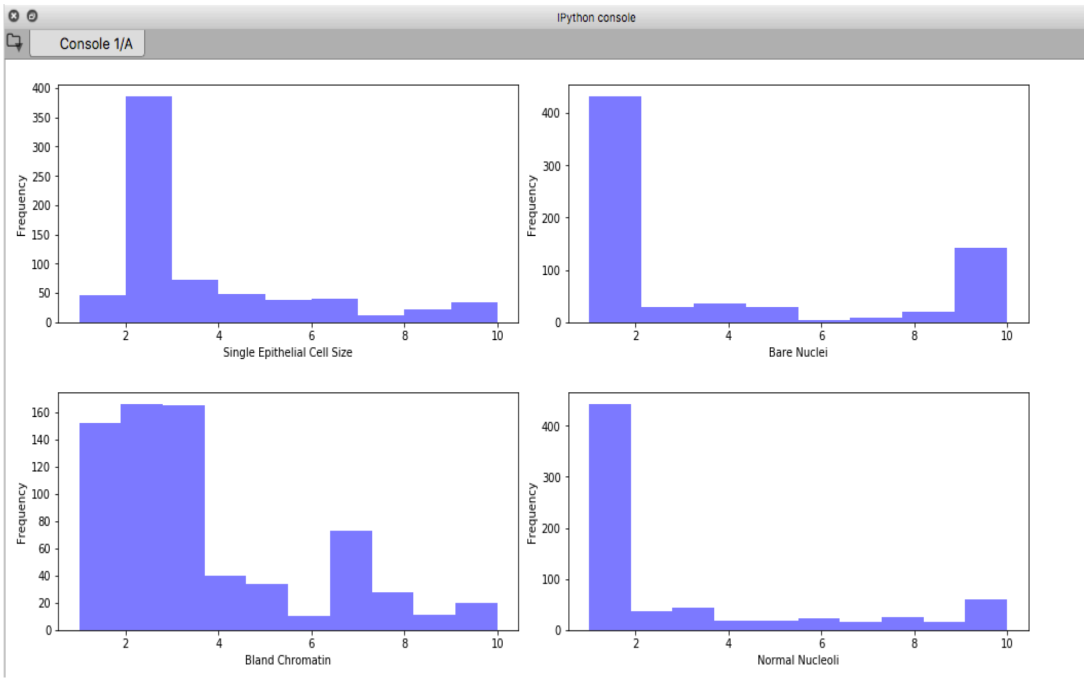


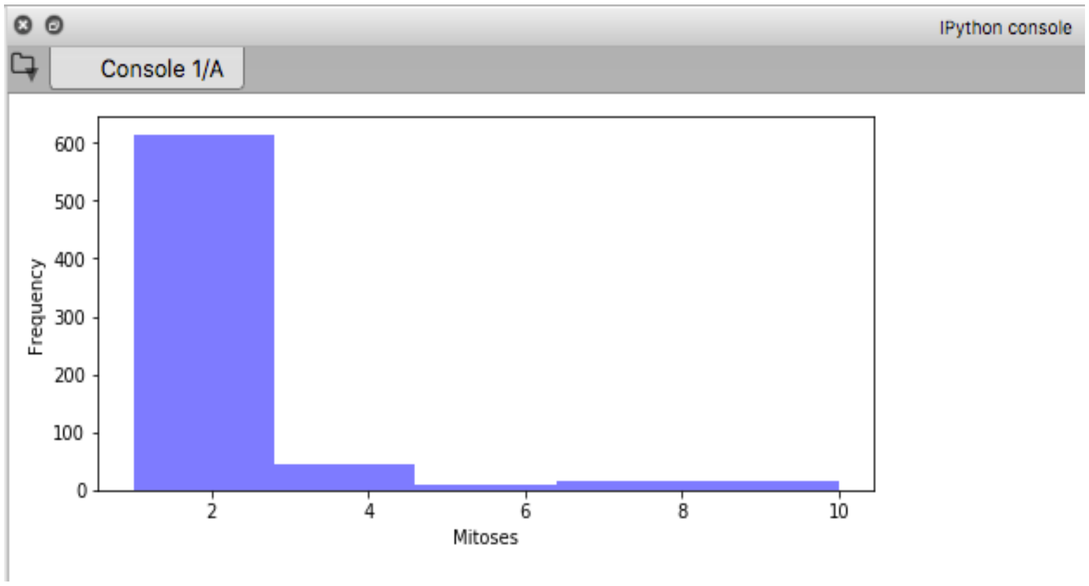
```
The NaN Value will be replaced with 3.54

Validate if the NaN has been replaced with or still exist

Empty DataFrame
Columns: [Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland
Chromatin, Normal Nucleoli, Mitoses, Class]
Index: []
```

3. Plot basic graphs
    a. Plotted the histogram for all the columns to see the distribution of the data

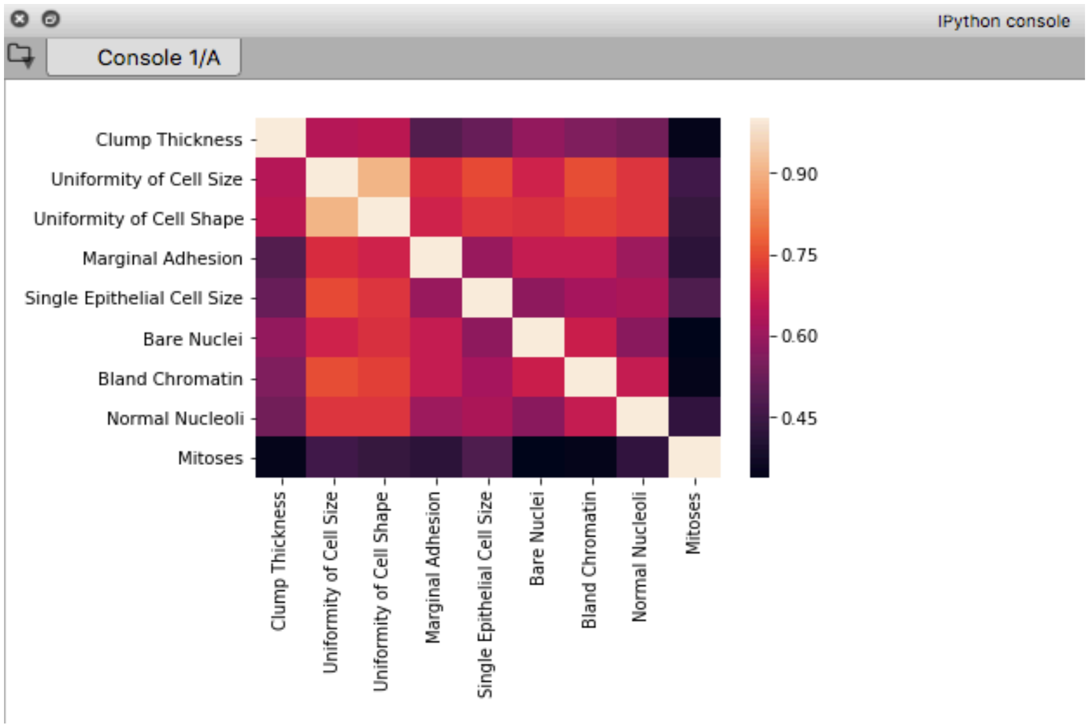
Computed and plotted the histogram for each of the attributes with varying the bin sizes, which indicate the frequency of each dimensions of the attributes.


IPython console
Console 1/A
Frequency
Clump Thickness
Uniformity Cell Size
Uniformity Cell Shape
Marginal Adhesion


IPython console
Console 1/A
Frequency
Single Epithelial Cell Size
Bare Nuclei
Bland Chromatin
Normal Nucleoli

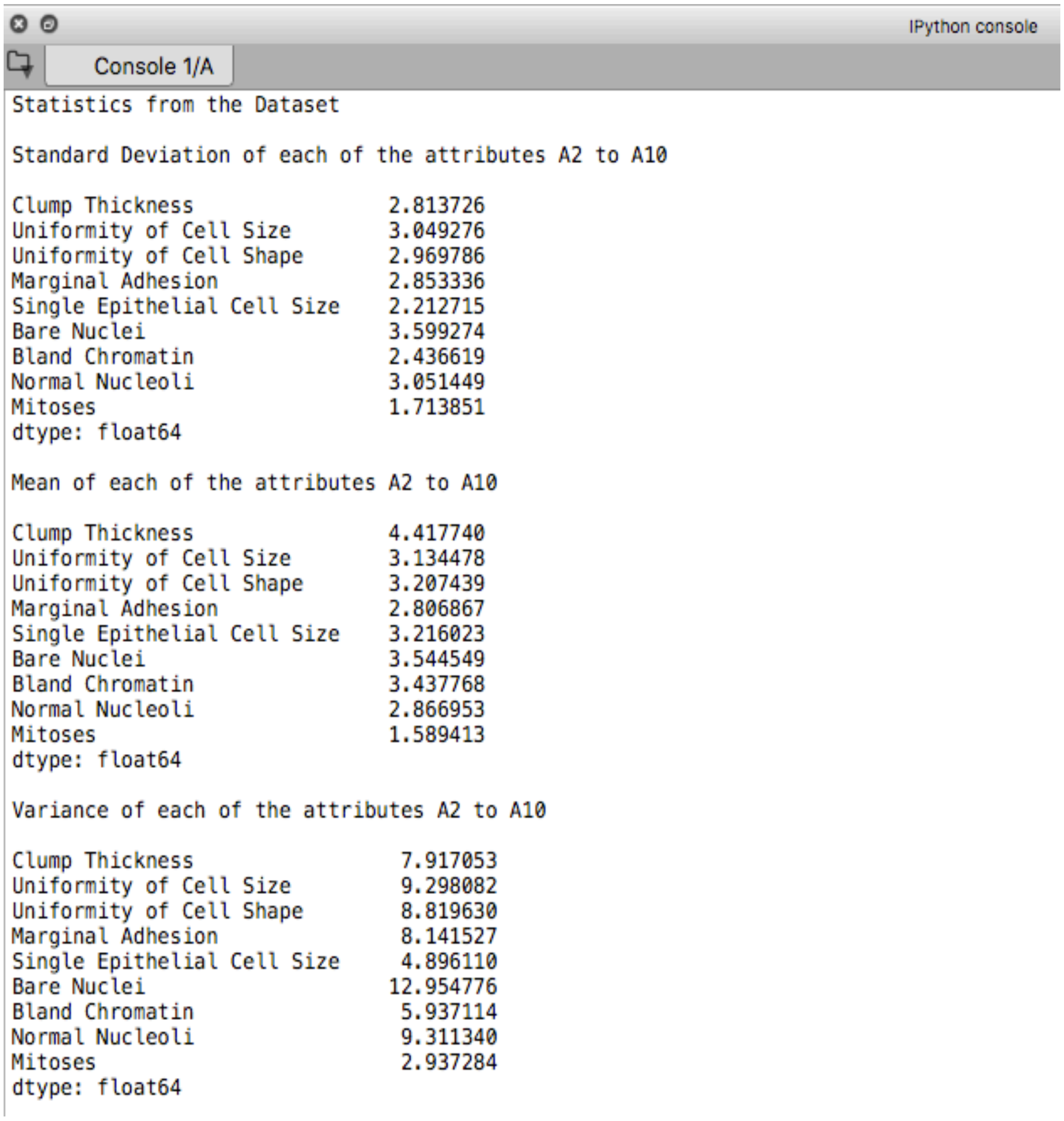


Correlational Matrix between each attributes -

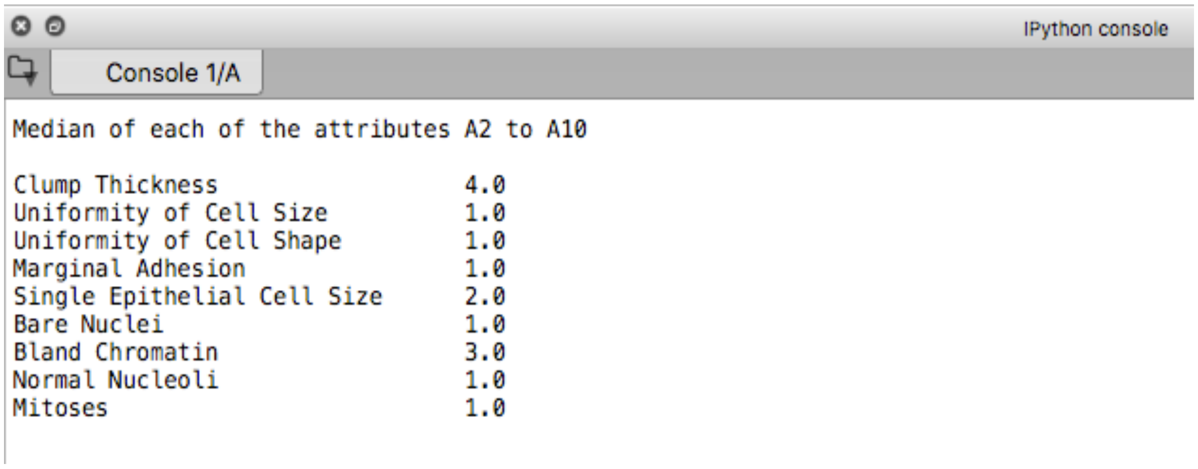


4. Compute data statistics
    a. Computed the different statistics for the columns for mean, median, standard deviation and Variance.

Below is the calculation for the different statistics for Mean, Median, Standard Deviation and Variance from the data in the Dataset.



```
Statistics from the Dataset

Standard Deviation of each of the attributes A2 to A10

Clump Thickness               2.813726
Uniformity of Cell Size       3.049276
Uniformity of Cell Shape      2.969786
Marginal Adhesion             2.853336
Single Epithelial Cell Size   2.212715
Bare Nuclei                   3.599274
Bland Chromatin               2.436619
Normal Nucleoli               3.051449
Mitoses                       1.713851
dtype: float64

Mean of each of the attributes A2 to A10

Clump Thickness               4.417740
Uniformity of Cell Size       3.134478
Uniformity of Cell Shape      3.207439
Marginal Adhesion             2.806867
Single Epithelial Cell Size   3.216023
Bare Nuclei                   3.544549
Bland Chromatin               3.437768
Normal Nucleoli               2.866953
Mitoses                       1.589413
dtype: float64

Variance of each of the attributes A2 to A10

Clump Thickness                7.917053
Uniformity of Cell Size        9.298082
Uniformity of Cell Shape       8.819630
Marginal Adhesion              8.141527
Single Epithelial Cell Size    4.896110
Bare Nuclei                   12.954776
Bland Chromatin                5.937114
Normal Nucleoli                9.311340
Mitoses                        2.937284
dtype: float64
```



```
Median of each of the attributes A2 to A10

Clump Thickness                4.0
Uniformity of Cell Size        1.0
Uniformity of Cell Shape       1.0
Marginal Adhesion              1.0
Single Epithelial Cell Size    2.0
Bare Nuclei                    1.0
Bland Chromatin                3.0
Normal Nucleoli                1.0
Mitoses                        1.0
```

Phase 2 – Initialization, Assignment and Recalculation/ Update

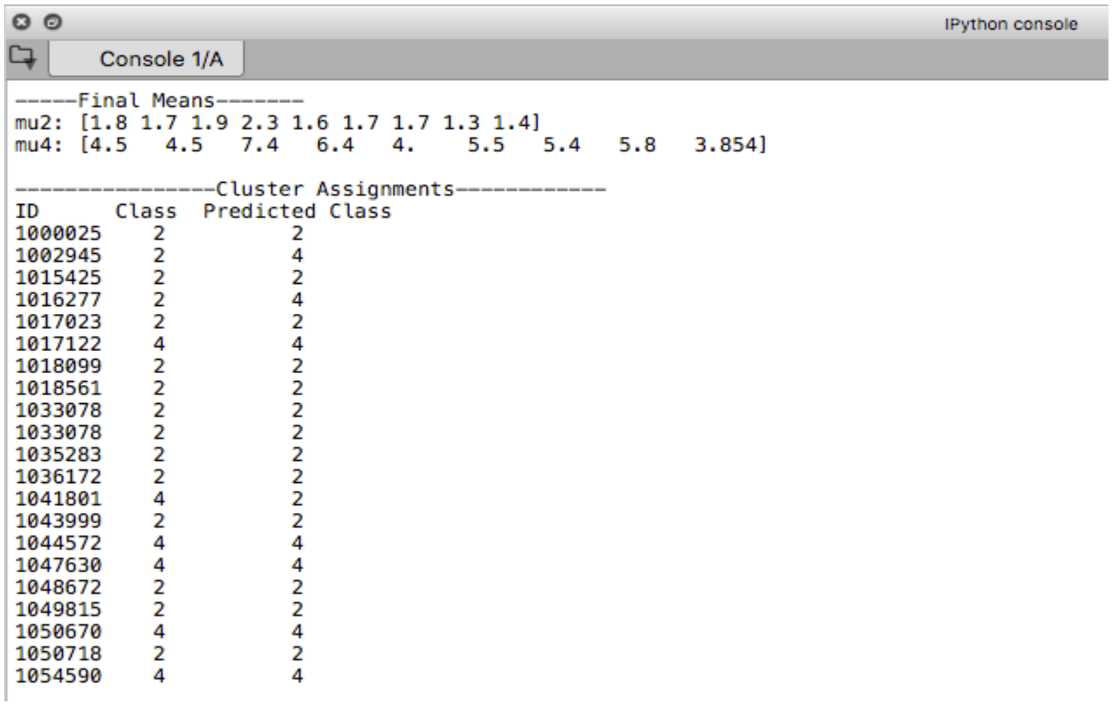
In this phase, we have followed 3 steps for calculating the clustering of the datapoints. Below are the details on the steps.

1. Initialization
    a. In this step, we have selected the two random points and considered as initial mean values for K=2 clusters using “df.sample” function of pandas.

2. Assignment
    a. From the previous step, we will be calculating the distance of each dimensional attribute to the random points (means μ2 and μ4) and assign them to either cluster 2 and cluster 4, and ensure the data point is closer to the mean.

3. Recalculation
    a. From the above steps, we will now recalculate the mean (μ2 and μ4) and reassign the data points to either of the cluster 2 and cluster 4.
    b. We will Iterate the step till below two conditions are met –
        i. All the data points don’t change their cluster assignment from previous run.
        Or
        ii. Iterate the results for 1500 times and consider the last run cluster assignments for the data points.

Below are the screenshot of the final means and first 20 predictions of the observations.



```
-----Final Means-------
mu2: [1.8 1.7 1.9 2.3 1.6 1.7 1.7 1.3 1.4]
mu4: [4.5   4.5   7.4   6.4   4.    5.5   5.4   5.8   3.854]

----------------Cluster Assignments------------
ID       Class  Predicted Class
1000025    2          2
1002945    2          4
1015425    2          2
1016277    2          4
1017023    2          2
1017122    4          4
1018099    2          2
1018561    2          2
1033078    2          2
1033078    2          2
1035283    2          2
1036172    2          2
1041801    4          2
1043999    2          2
1044572    4          4
1047630    4          4
1048672    2          2
1049815    2          2
1050670    4          4
1050718    2          2
1054590    4          4
```

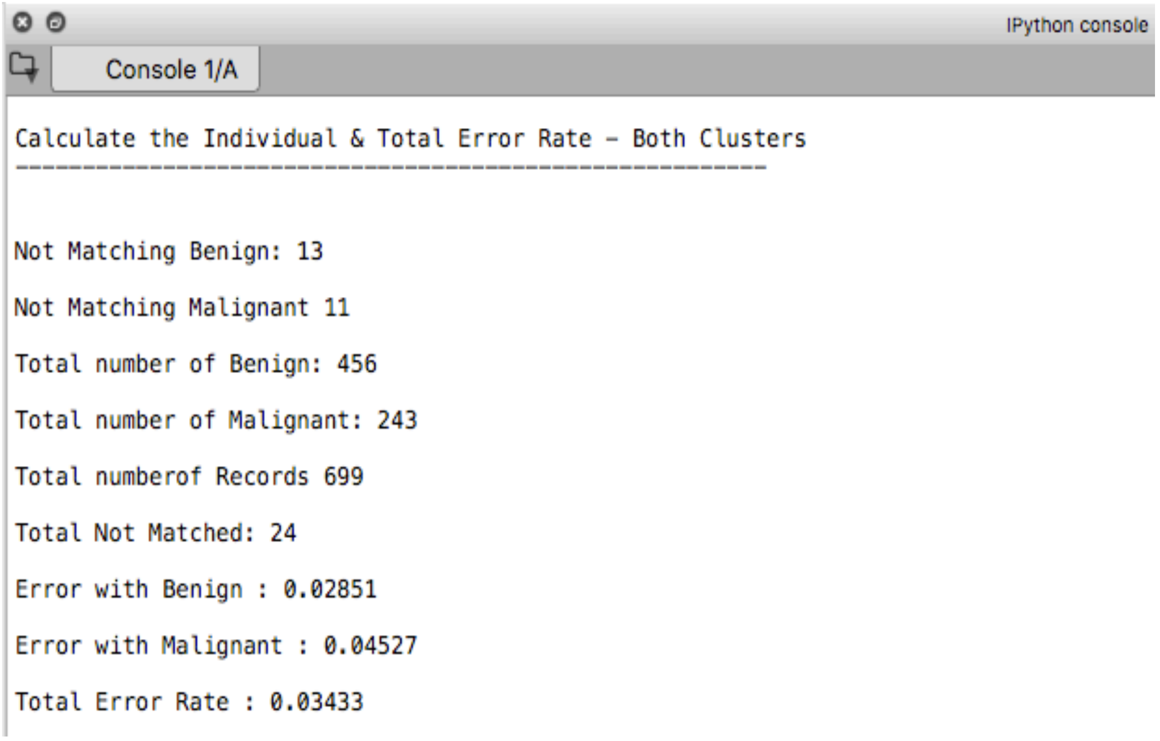
There are different functions has been created picktwocor(x), assigndf(x, u1, u2), updatemean(x, pc), recalmeanfifty(x, col, pc) for picking the two random means and then creating and assigning the prediction for cluster 2 and 4. Once we have initial clustering 2 and 4 with initial random means, update the mean with from above cluster2 and 4, repeat the same for 1500 iterations or if matched with previous iterations

Detail code is provided in below Overall Code section.

Phase 3 – Error rate calculation

1. Benign, Malignant and Total error rate of your 2 clusters.
   a. We will calculate the error rate of Cluster 2 assignment
   b. We will calculate the error rate of Cluster 4 assignment
   c. We will calculate the error rate of Total data point's assignment.

The error for the predicted values which are actually Benign (Cluster with 2) and has been predicted as Malignant (Cluster with 4) and the Overall Error Rate compared to actual classification on each datapoints.



```
Calculate the Individual & Total Error Rate - Both Clusters
--------------------------------------------------------


Not Matching Benign: 13

Not Matching Malignant 11

Total number of Benign: 456

Total number of Malignant: 243

Total numberof Records 699

Total Not Matched: 24

Error with Benign : 0.02851

Error with Malignant : 0.04527

Total Error Rate : 0.03433
```

Overall code –

```
#!/usr/bin/env python3

# -*- coding: utf-8 -*-

"""

Created on Mon Jul 23 01:55:37 2018

@author: ArijitSinha


Question 1 - During the third week, you will analyze the quality of the clustering.

• Write a code to calculate the individual and total error rate of your 2 clusters.

• Submit final report
```

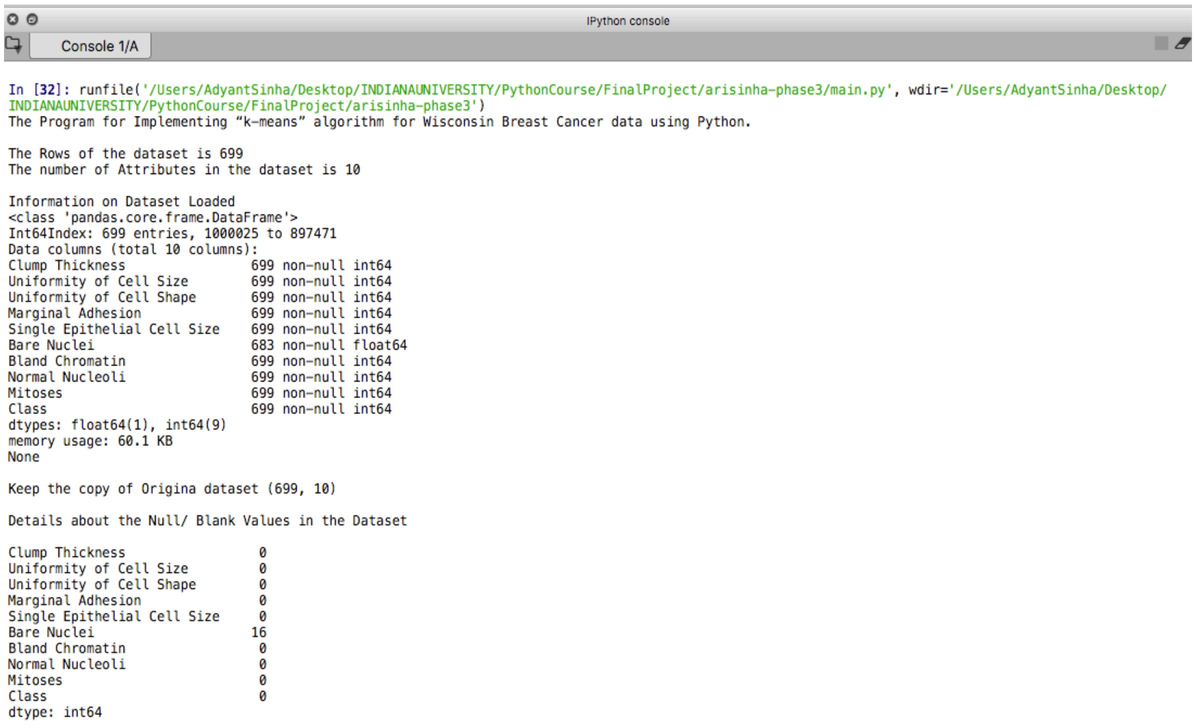
```
Name - Arijit Sinha

"""

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

def contblk(x):

   print('Details about the Null/ Blank Values in the Dataset')

   print()

   print(np.sum(x.isna()))

   print()

   return(x)

def repbnk(x):

   npval = np.around(np.nanmean(x['Bare Nuclei'], dtype = float), 2)

   print('The NaN Value will be replaced with', npval)

   #npval = np.around((x['Bare Nuclei'].mean()), 2)

   x['Bare Nuclei'].replace(np.nan, npval, inplace=True)

   print()

   print('Validate if the NaN has been replaced with or still exist')

   print()

   print(x.loc[x['Bare Nuclei'].isna()])

   return (x)


def statdf(x):

   print()

   print('Statistics from the Dataset')

   print()

   std = np.std(x)

   mea = np.mean(x)

   var = np.var(x)

   med = (np.median(x, axis = 0))
```

```
    return (std, mea, var, med)

def histplt(x):

    fig = plt.figure(figsize=(14, 32))

    ax1 = fig.add_subplot(9, 2, 1)

    ax2 = fig.add_subplot(9, 2, 2)

    ax3 = fig.add_subplot(9, 2, 3)

    ax4 = fig.add_subplot(9, 2, 4)

    ax5 = fig.add_subplot(9, 2, 5)

    ax6 = fig.add_subplot(9, 2, 6)

    ax7 = fig.add_subplot(9, 2, 7)

    ax8 = fig.add_subplot(9, 2, 8)

    ax9 = fig.add_subplot(9, 2, 9)

    ax1.hist(x["Clump Thickness"], bins=10, color = "b", alpha = 0.5)

    ax2.hist(x["Uniformity of Cell Size"], bins=9, color = "b", alpha = 0.5)

    ax3.hist(x["Uniformity of Cell Shape"], bins=10, color = "b", alpha = 0.5)

    ax4.hist(x["Marginal Adhesion"], bins=10, color = "b", alpha = 0.5)

    ax5.hist(x["Single Epithelial Cell Size"], bins=9, color = "b", alpha = 0.5)

    ax6.hist(x["Bare Nuclei"], bins= 8, color = "b", alpha = 0.5)

    ax7.hist(x["Bland Chromatin"], bins=10, color = "b", alpha = 0.5)

    ax8.hist(x["Normal Nucleoli"], bins=10, color = "b", alpha = 0.5)

    ax9.hist(x["Mitoses"],bins=5, color = "b", alpha = 0.5)


    ax1.set_ylabel("Frequency")

    ax2.set_ylabel("Frequency")

    ax3.set_ylabel("Frequency")

    ax4.set_ylabel("Frequency")

    ax5.set_ylabel("Frequency")

    ax6.set_ylabel("Frequency")

    ax7.set_ylabel("Frequency")

    ax8.set_ylabel("Frequency")

    ax9.set_ylabel("Frequency")
    ax1.set_xlabel("Clump Thickness")
    ax2.set_xlabel("Uniformity Cell Size")
    ax3.set_xlabel("Uniformity Cell Shape")
    ax4.set_xlabel("Marginal Adhesion")
    ax5.set_xlabel("Single Epithelial Cell Size")
    ax6.set_xlabel("Bare Nuclei")
    ax7.set_xlabel("Bland Chromatin")
    ax8.set_xlabel("Normal Nucleoli")
    ax9.set_xlabel("Mitoses")
    plt.tight_layout(pad=1.0, w_pad=0.5, h_pad=2.5)
    plt.show()
def corrmatrix(x):
    corr = x.corr()
    # plot the heatmap
    sns.heatmap(corr, xticklabels=corr.columns,yticklabels=corr.columns)
def picktwocor(x):
    k = 2
    c = x.sample(k)
    #d = [c.iloc[0].values, c.iloc[1].values]
    randomu1 = c.iloc[0].values
    randomu2 = c.iloc[1].values
    #print("Initial Random Mean selected are :")
    #print('randomu1 :', randomu1)
    #print('randomu2 :', randomu2)
    #plt.scatter(u1, u2, color = 'green')
    return (randomu1, randomu2)
    # Alternate code -
    #cn = x.iloc[np.random.choice(np.arange(len(x)), k, False)]
    #dn = [cn.iloc[0].values, cn.iloc[1].values]
    #un1 = cn.iloc[0].values
    #un2 = cn.iloc[1].values
    #plt.scatter(un1, un2, color = 'red')
def assigndf(x, u1, u2):
    PredictedClass = []
    for i in range(len(x)):
        distu1 = np.around((np.sqrt(sum(((x.iloc[i].values)-(u1))**2))),2)
        distu2 = np.around((np.sqrt(sum(((x.iloc[i].values)-(u2))**2))),2)
        if distu1<distu2:
            PredictedClass.append(2)
        else:
            PredictedClass.append(4)
    return(PredictedClass)
def updatemean(x, pc):
    x['PreCls'] = pd.Series(pc, index=x.index)
    dfwith2 = x[x['PreCls']==2]
    dfwith4 = x[x['PreCls']==4]
    #print(len(dfwith2), len(dfwith4))

    utwoar = np.mean(dfwith2[:9], axis = 1).values
    ufourar = np.mean(dfwith4[:9], axis = 1).values
#print(len(utwoar), len(ufourar))
    x.drop('PreCls', axis = 1, inplace  = True)
    return(x, utwoar, ufourar)
def recalmeanfifty(x, col, pc):
    i = 0
    while (i <= 1499):
        x1, a, b = updatemean(x, pc)
        #print(len(a), len(b))
        newpc = assigndf(x1, a, b)
        if pc == newpc:
            return(newpc)
            break
        else:
            pc = newpc
            x1, a, b = updatemean(x1, newpc)
            newpc = assigndf(x1, a, b)
    i = i+1
        print()
        print("-----Final Means-------")
        print("mu2:", a)
        print("mu4:", b)
    return(newpc)
def ErrorRate(actclass,predclass):
    counttwo= 0
    countfour = 0
    Notmatch = 0
    cbenign = 0
    cmalignant = 0
    TotalDatapoint = len(actclass)
    for i in range(len(actclass)):
        if (actclass[i] ==2 and predclass[i]==4):
            counttwo +=1
        elif (actclass[i]==4 and predclass[i]==2):
            countfour +=1
   for i in range(len(actclass)):
        if (predclass[i]==2):
            cbenign +=1
        else:
            cmalignant +=1
    for i in range(len(actclass)):
        if (actclass[i]!=predclass[i]):
            Notmatch +=1
    print()
    print("Not Matching Benign:", counttwo)

    print()

    print("Not Matching Malignant", countfour)

    print()

    print("Total number of Benign:", cbenign)

    print()

    print("Total number of Malignant:", cmalignant)

    print()

    print("Total numberof Records", TotalDatapoint)

    print()

    print("Total Not Matched:", Notmatch)

    errorB = counttwo/cbenign


    errorM = countfour/cmalignant

    TotalErrorRate = Notmatch/TotalDatapoint

    print()

    print("Error with Benign :", np.around(errorB,5))

    print()

    print("Error with Malignant :",np.around(errorM, 5))

    print()

    print("Total Error Rate :", np.around(TotalErrorRate,5))

def main():

    print('The Program for Implementing “k-means” algorithm for Wisconsin Breast
Cancer data using Python.')

    url = ('https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-
wisconsin/'

         'breast-cancer-wisconsin.data')

    df = pd.read_csv(url,

                 header=None, names=['ID', 'Clump Thickness', 'Uniformity of Cell Size',

                              'Uniformity of Cell Shape', 'Marginal Adhesion',

                              'Single Epithelial Cell Size', 'Bare Nuclei',
                              'Bland Chromatin', 'Normal Nucleoli', 'Mitoses',
                              'Class'], na_values="?")
    dfcol = df['Class'].copy()
    df = df.set_index('ID')
    m,n = df.shape
    print()
    print('The Rows of the dataset is', m)
    print('The number of Attributes in the dataset is', n)
    print()
    # Information on Dataset loaded
    print('Information on Dataset Loaded')
    print(df.info())
    print()
    dforiginal = df
    print('Keep the copy of Origina dataset', dforiginal.shape)
    print()
    # Data Cleaning activties
    # Count the details on Null and ? in the data columns
    df = contblk(df)
    print('Datasection has the null Values')
    print()
    print(df.loc[df['Bare Nuclei'].isna()])
    # replace Nan and ? in the Data column
    dfwb = repbnk(df)
    dfwb.drop('Class',axis=1, inplace = True)
    # Draw 9 Historgram
    histplt(dfwb)
    # Data Statistics Details
    # Calcuate the Statistics from dataset columns
    stdf, meaf, varf, medf = statdf(dfwb)
    print('Standard Deviation of each of the attributes A2 to A10')
    print()

    print(stdf)

    print()

    print('Mean of each of the attributes A2 to A10')

    print()

    print(meaf)

    print()

    print('Variance of each of the attributes A2 to A10')

    print()

    print(varf)

    print()

    print('Median of each of the attributes A2 to A10')

    print()

    col_names = df.columns.tolist()

    for i in range(0,9):

        print(col_names[i].ljust(33) + str(medf[i]).ljust(2))

    print()

    # Draw Correlational Matrix

    #print("Correlational Matrix")

    #corrmatrix(dfwb)

    #print()

    picktwocor(dfwb)

    arr1, arr2 = picktwocor(dfwb)

    predclassint = assigndf(dfwb, arr1, arr2)

    #predclass = recalmeanfirst(dfwb, predclassint)

    predclassfifty = recalmeanfifty(dfwb, dfcol, predclassint)

    print()

    print("----------------Cluster Assignments------------")

    print("ID     ", "Class ", "Predicted Class")

    for i in range(0,21):

        print(dfwb.index[i],'  ', dfcol[i],'        ', predclassfifty[i])
```

```
        print()

        print("Calculate the Individual & Total Error Rate - Both Clusters")

        print("-----------------------------------------------------")

        print()

        ErrorRate(dfcol,predclassfifty)

    main()
```

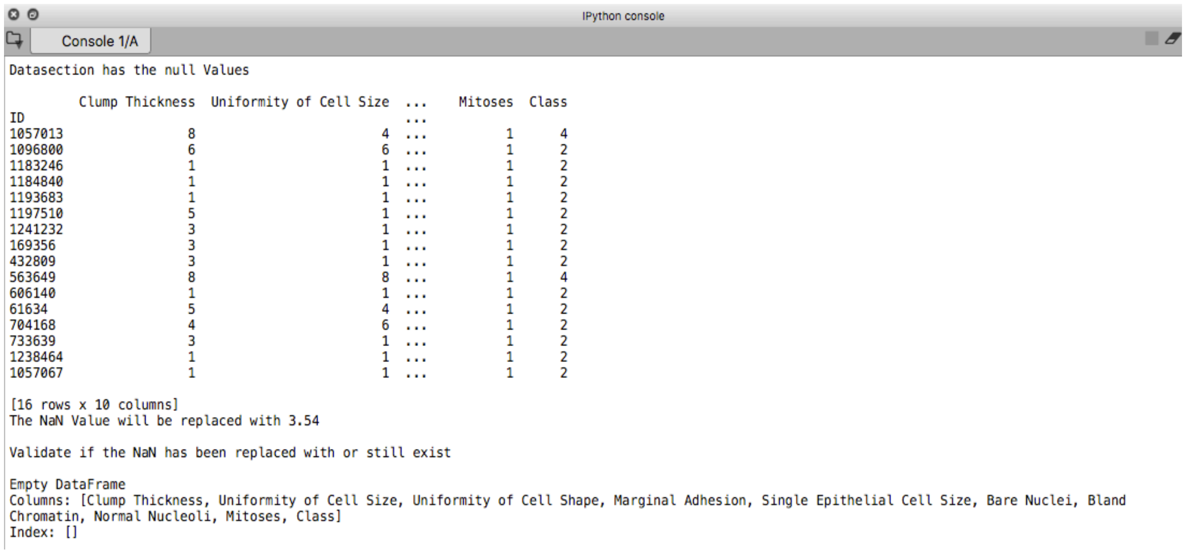
<u>Overall Script run and Outcome –</u>

Load Dataset – Wisconsin Breast Cancer



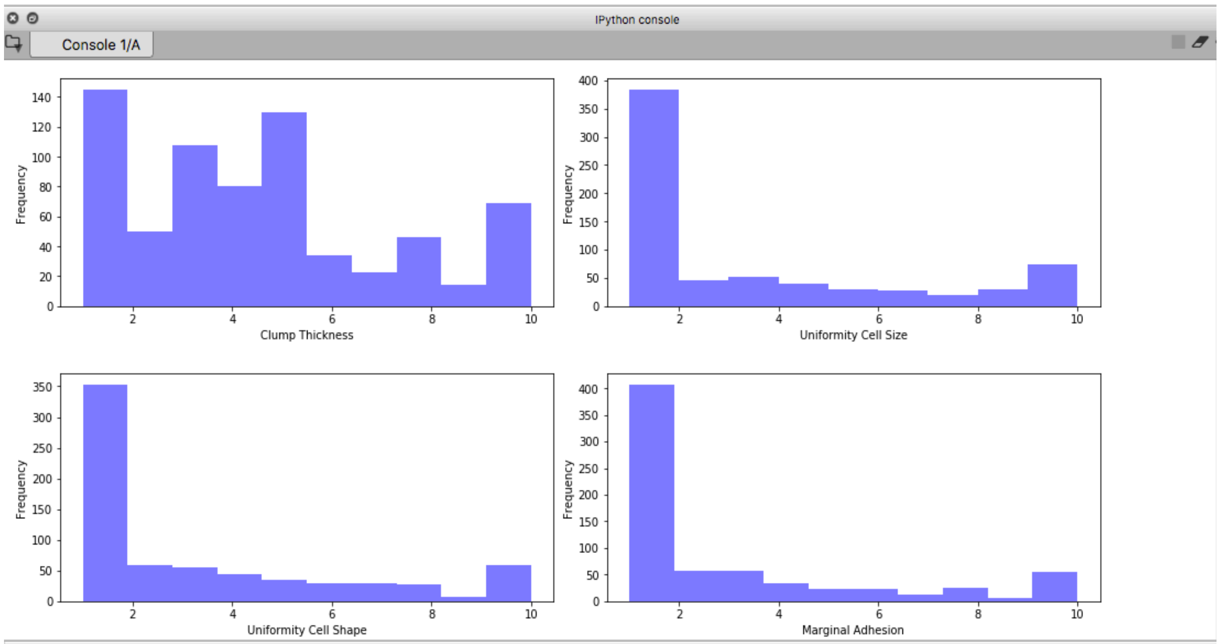
```
In [32]: runfile('/Users/AdyantSinha/Desktop/INDIANAUNIVERSITY/PythonCourse/FinalProject/arisinha-phase3/main.py', wdir='/Users/AdyantSinha/Desktop/
INDIANAUNIVERSITY/PythonCourse/FinalProject/arisinha-phase3')
The Program for Implementing "k-means" algorithm for Wisconsin Breast Cancer data using Python.

The Rows of the dataset is 699
The number of Attributes in the dataset is 10

Information on Dataset Loaded
<class 'pandas.core.frame.DataFrame'>
Int64Index: 699 entries, 1000025 to 897471
Data columns (total 10 columns):
Clump Thickness                699 non-null int64
Uniformity of Cell Size        699 non-null int64
Uniformity of Cell Shape       699 non-null int64
Marginal Adhesion              699 non-null int64
Single Epithelial Cell Size    699 non-null int64
Bare Nuclei                    683 non-null float64
Bland Chromatin                699 non-null int64
Normal Nucleoli                699 non-null int64
Mitoses                        699 non-null int64
Class                          699 non-null int64
dtypes: float64(1), int64(9)
memory usage: 60.1 KB
None

Keep the copy of Origina dataset (699, 10)

Details about the Null/ Blank Values in the Dataset

Clump Thickness                 0
Uniformity of Cell Size         0
Uniformity of Cell Shape        0
Marginal Adhesion               0
Single Epithelial Cell Size     0
Bare Nuclei                    16
Bland Chromatin                 0
Normal Nucleoli                 0
Mitoses                         0
Class                           0
dtype: int64
```

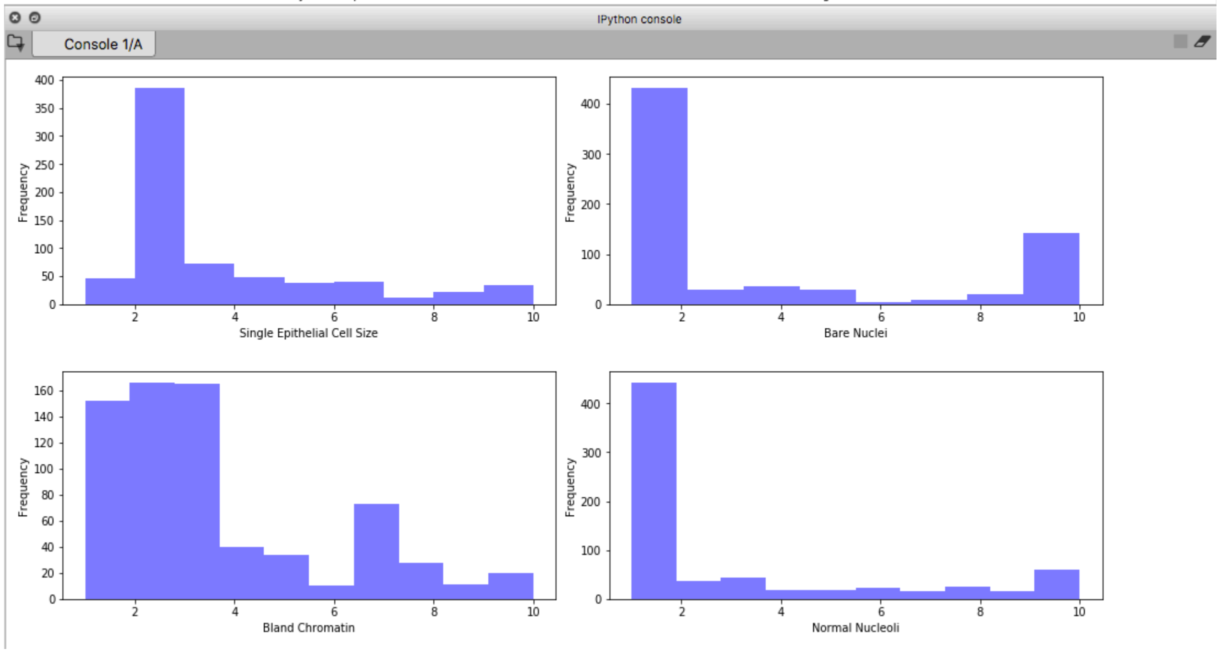
Handling Missing Values Bare Nuclei – With "mean" value

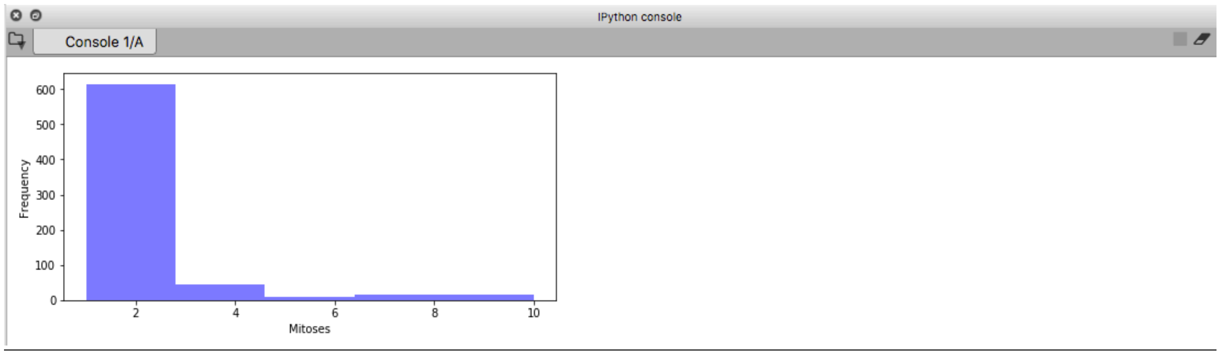


```
Datasection has the null Values

         Clump Thickness  Uniformity of Cell Size  ...    Mitoses  Class
ID                                                 ...
1057013                8                        4  ...          1      4
1096800                6                        6  ...          1      2
1183246                1                        1  ...          1      2
1184840                1                        1  ...          1      2
1193683                1                        1  ...          1      2
1197510                5                        1  ...          1      2
1241232                3                        1  ...          1      2
169356                 3                        1  ...          1      2
432809                 3                        1  ...          1      2
563649                 8                        8  ...          1      4
606140                 1                        1  ...          1      2
61634                  5                        4  ...          1      2
704168                 4                        6  ...          1      2
733639                 3                        1  ...          1      2
1238464                1                        1  ...          1      2
1057067                1                        1  ...          1      2

[16 rows x 10 columns]
The NaN Value will be replaced with 3.54

Validate if the NaN has been replaced with or still exist

Empty DataFrame
Columns: [Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland
Chromatin, Normal Nucleoli, Mitoses, Class]
Index: []
```

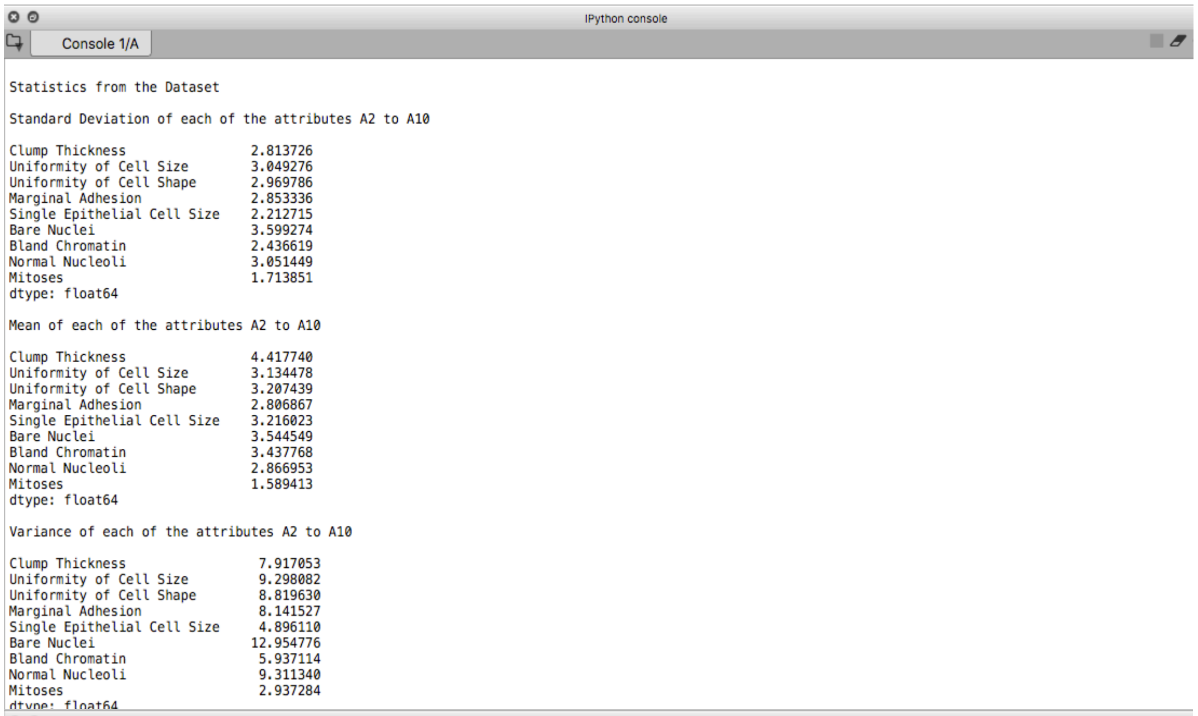
Visualization of each Attribute - Histrogram



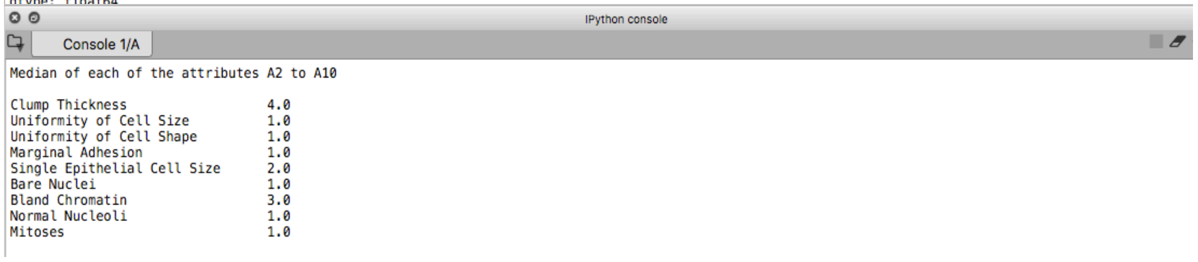




Statistics from each Attribute

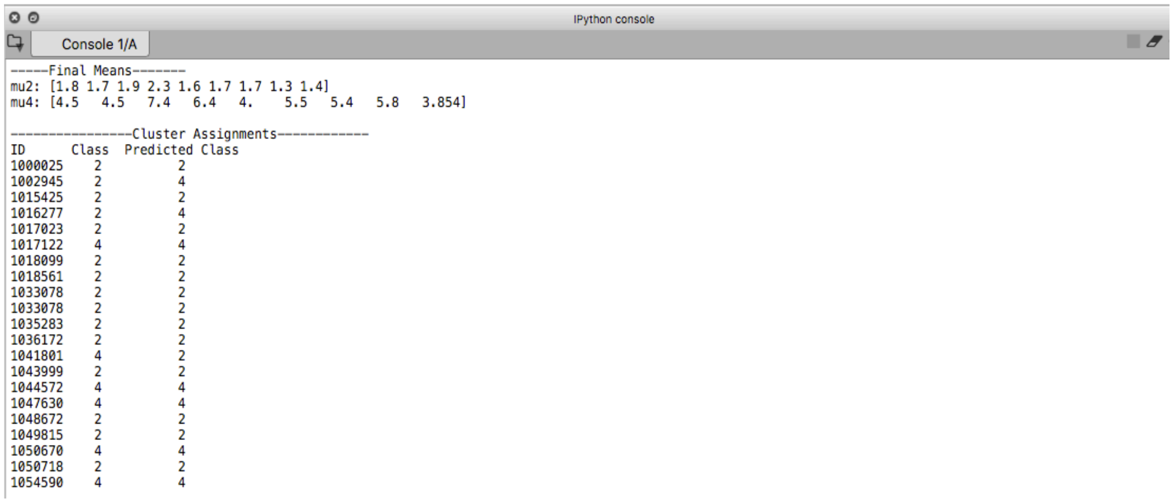


```
Statistics from the Dataset

Standard Deviation of each of the attributes A2 to A10

Clump Thickness                2.813726
Uniformity of Cell Size        3.049276
Uniformity of Cell Shape       2.969786
Marginal Adhesion              2.853336
Single Epithelial Cell Size    2.212715
Bare Nuclei                    3.599274
Bland Chromatin                2.436619
Normal Nucleoli                3.051449
Mitoses                        1.713851
dtype: float64

Mean of each of the attributes A2 to A10

Clump Thickness                4.417740
Uniformity of Cell Size        3.134478
Uniformity of Cell Shape       3.207439
Marginal Adhesion              2.806867
Single Epithelial Cell Size    3.216023
Bare Nuclei                    3.544549
Bland Chromatin                3.437768
Normal Nucleoli                2.866953
Mitoses                        1.589413
dtype: float64

Variance of each of the attributes A2 to A10

Clump Thickness                 7.917053
Uniformity of Cell Size         9.298082
Uniformity of Cell Shape        8.819630
Marginal Adhesion               8.141527
Single Epithelial Cell Size     4.896110
Bare Nuclei                    12.954776
Bland Chromatin                 5.937114
Normal Nucleoli                 9.311340
Mitoses                         2.937284
dtype: float64
```



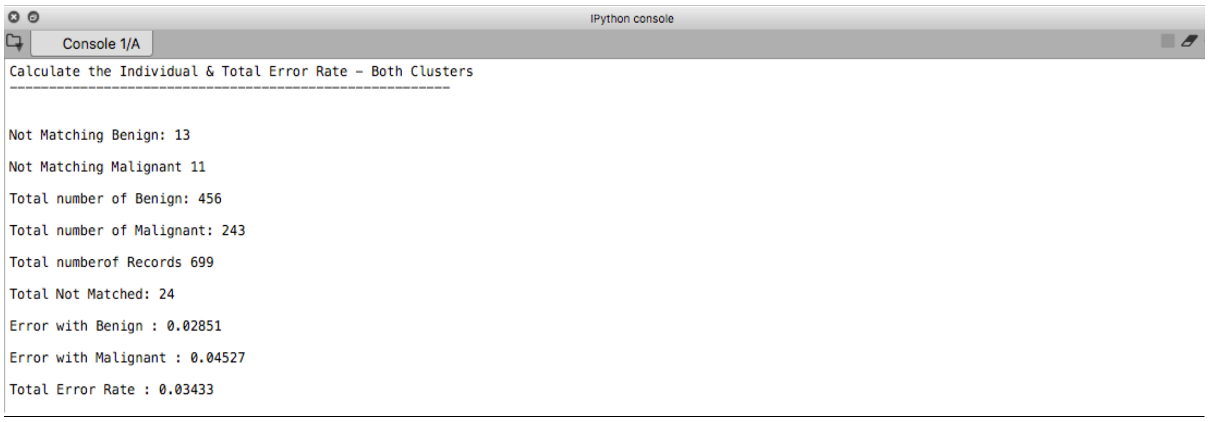
```
Median of each of the attributes A2 to A10

Clump Thickness                   4.0
Uniformity of Cell Size           1.0
Uniformity of Cell Shape          1.0
Marginal Adhesion                 1.0
Single Epithelial Cell Size       2.0
Bare Nuclei                       1.0
Bland Chromatin                   3.0
Normal Nucleoli                   1.0
Mitoses                           1.0
```

First 20 Prediction and Actuals



```
-----Final Means-------
mu2: [1.8 1.7 1.9 2.3 1.6 1.7 1.7 1.3 1.4]
mu4: [4.5   4.5   7.4   6.4   4.    5.5   5.4   5.8   3.854]

---------------Cluster Assignments------------
ID      Class  Predicted Class
1000025    2          2
1002945    2          4
1015425    2          2
1016277    2          4
1017023    2          2
1017122    4          4
1018099    2          2
1018561    2          2
1033078    2          2
1033078    2          2
1035283    2          2
1036172    2          2
1041801    4          2
1043999    2          2
1044572    4          4
1047630    4          4
1048672    2          2
1049815    2          2
1050670    4          4
1050718    2          2
1054590    4          4
```

ErrorRate


IPython console

Console 1/A

```
Calculate the Individual & Total Error Rate - Both Clusters
---------------------------------------------------------

Not Matching Benign: 13

Not Matching Malignant 11

Total number of Benign: 456

Total number of Malignant: 243

Total numberof Records 699

Total Not Matched: 24

Error with Benign : 0.02851

Error with Malignant : 0.04527

Total Error Rate : 0.03433
```

Summary/conclusions –

With the above implementation, we observe that the actual we have approximately '0.03433' error rate between the prediction and actual class has been provided in the dataset. Error rate with Benign is '0.02851' and Error rate with Malignant is '0.04527'.

References –

http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

https://pdfs.semanticscholar.org/a92f/718fe24a60a9731a5be0280171d52a6c6e52.pdf