AIM: IMPLEMENTATION AND ANALYSIS OF CLUSTERING ALGORITHMS LIKE K-MEANS

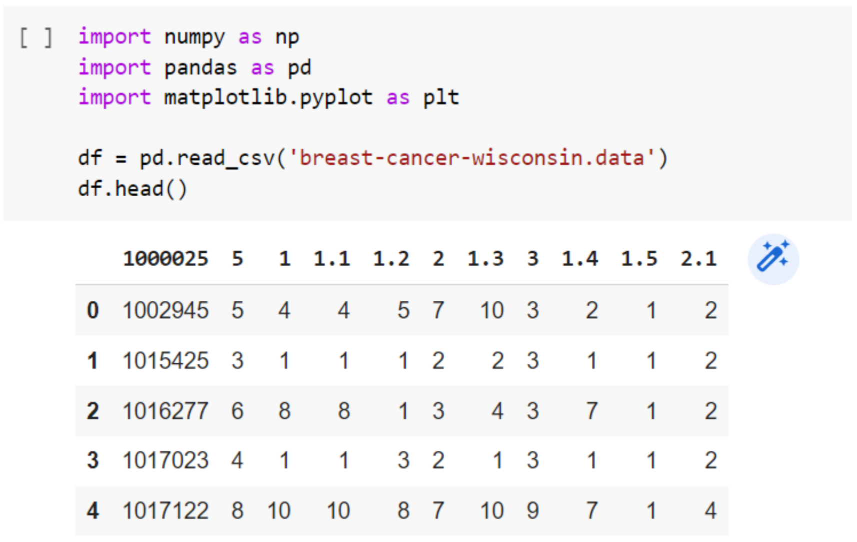
THEORY:

K-medoids clustering is a variant of K-means that is more robust to noises and outliers. Instead of using the mean point as the center of a cluster, K medoids uses an actual point in the cluster to represent it. Medoid is the most centrally located object of the cluster, with minimum sum of distances to other points.

The group of points in the right form a cluster, while the rightmost point is an outlier. Mean is greatly influenced by the outlier and thus cannot represent the correct cluster center, while medoid is robust to the outlier and correctly represents the cluster center. It is a clustering algorithm resembling the K-Means clustering technique. It falls under the category of unsupervised machine learning. It majorly differs from the K-Means algorithm in terms of the way it selects the clusters' centres. The former selects the average of a cluster's points as its centre (which may or may not be one of the data points) while the latter always picks the actual data points from the clusters as their centres (also known as 'exemplars' or 'medoids'). K-Medoids also differ in this respect from the K-Medians algorithm which is the same as K-means, except that it chooses the medians (instead of means) of the clusters as centres

SOURCE CODE:

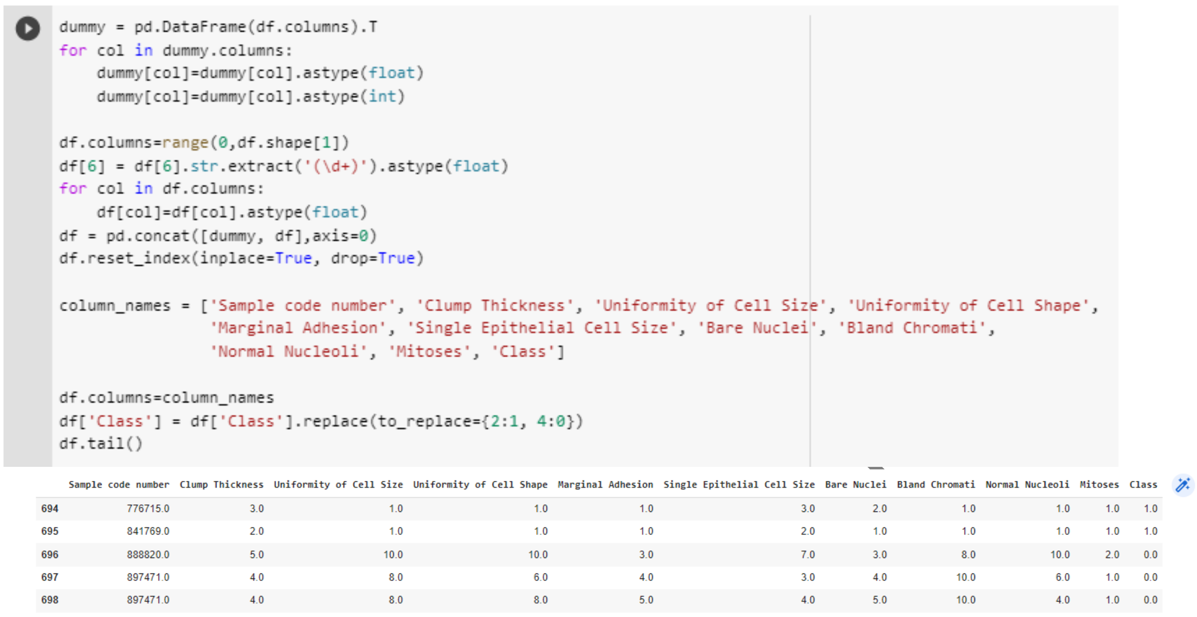
1) IMPORTING LIBRARIES & READING DATASET:



```
[ ] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt

    df = pd.read_csv('breast-cancer-wisconsin.data')
    df.head()
```

```
   1000025  5   1  1.1  1.2  2  1.3  3  1.4  1.5  2.1
0  1002945  5   4    4    5  7   10  3    2    1    2
1  1015425  3   1    1    1  2    2  3    1    1    2
2  1016277  6   8    8    1  3    4  3    7    1    2
3  1017023  4   1    1    3  2    1  3    1    1    2
4  1017122  8  10   10    8  7   10  9    7    1    4
```

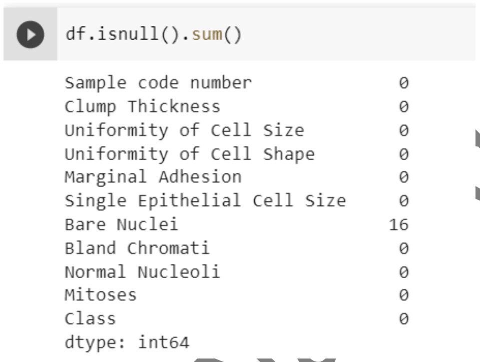
2) DATA PREPROCESSING:



```
dummy = pd.DataFrame(df.columns).T
for col in dummy.columns:
    dummy[col]=dummy[col].astype(float)
    dummy[col]=dummy[col].astype(int)

df.columns=range(0,df.shape[1])
df[6] = df[6].str.extract('(\d+)').astype(float)
for col in df.columns:
    df[col]=df[col].astype(float)
df = pd.concat([dummy, df],axis=0)
df.reset_index(inplace=True, drop=True)

column_names = ['Sample code number', 'Clump Thickness', 'Uniformity of Cell Size', 'Uniformity of Cell Shape',
                'Marginal Adhesion', 'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromati',
                'Normal Nucleoli', 'Mitoses', 'Class']

df.columns=column_names
df['Class'] = df['Class'].replace(to_replace={2:1, 4:0})
df.tail()
```

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromati	Normal Nucleoli	Mitoses	Class
694	776715.0	3.0	1.0	1.0	1.0	3.0	2.0	1.0	1.0	1.0	1.0
695	841769.0	2.0	1.0	1.0	1.0	2.0	1.0	1.0	1.0	1.0	1.0
696	888820.0	5.0	10.0	10.0	3.0	7.0	3.0	8.0	10.0	2.0	0.0
697	897471.0	4.0	8.0	6.0	4.0	3.0	4.0	10.0	6.0	1.0	0.0
698	897471.0	4.0	8.0	8.0	5.0	4.0	5.0	10.0	4.0	1.0	0.0

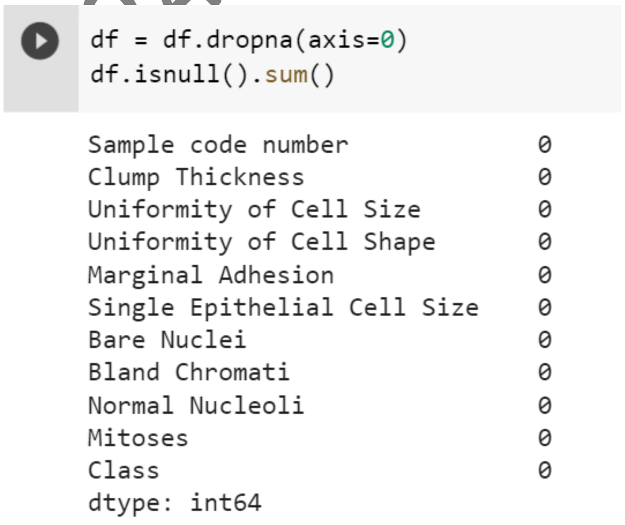
3) COUNT OF NULL VALUES:



```
df.isnull().sum()
```

```
Sample code number             0
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 Chromati                 0
Normal Nucleoli                0
Mitoses                        0
Class                          0
dtype: int64
```

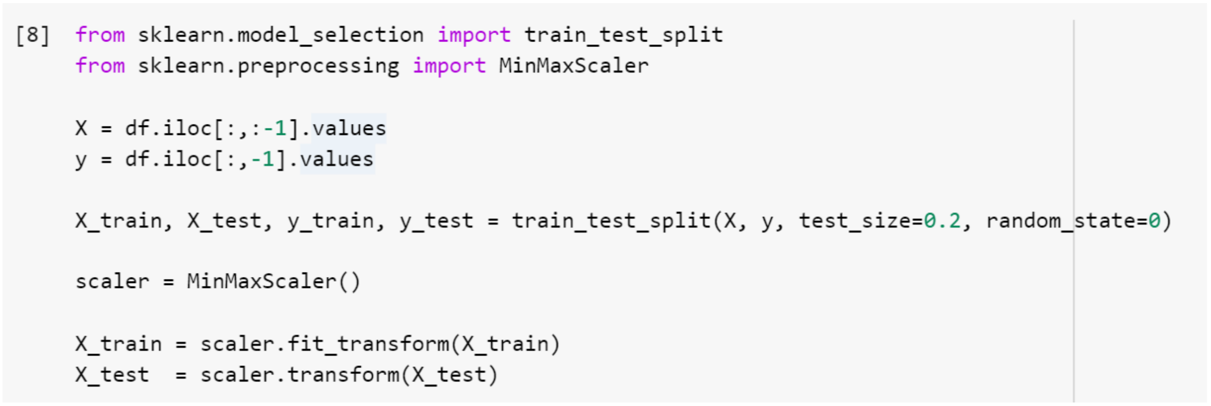
4) DROPPING NULL VALUES:



```
df = df.dropna(axis=0)
df.isnull().sum()
```

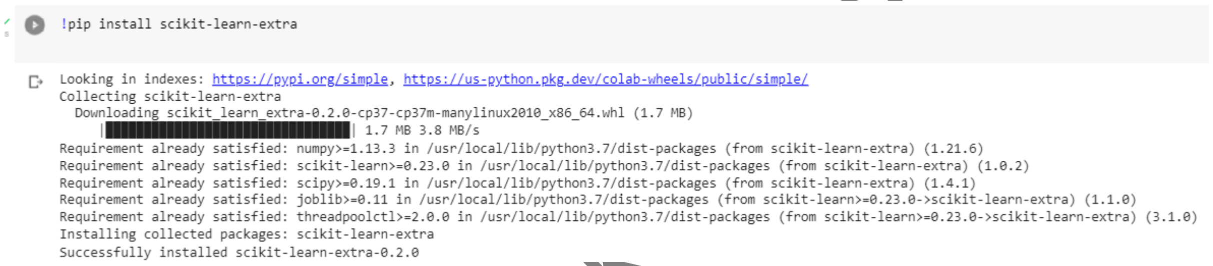
```
Sample code number             0
Clump Thickness                0
Uniformity of Cell Size        0
Uniformity of Cell Shape       0
Marginal Adhesion              0
Single Epithelial Cell Size    0
Bare Nuclei                    0
Bland Chromati                 0
Normal Nucleoli                0
Mitoses                        0
Class                          0
dtype: int64
```

5) NORMALIZATION:



```
[8] from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import MinMaxScaler

    X = df.iloc[:,:-1].values
    y = df.iloc[:,-1].values

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

    scaler = MinMaxScaler()

    X_train = scaler.fit_transform(X_train)
    X_test  = scaler.transform(X_test)
```

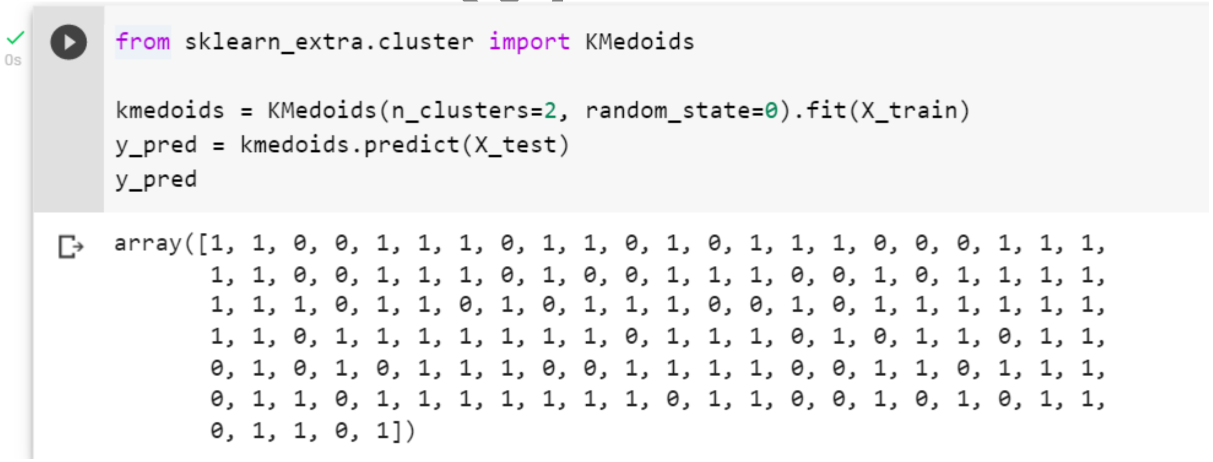
6) INSTALLING & IMPORTING LIBRARIES FOR KMEDIOD:



```
!pip install scikit-learn-extra
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting scikit-learn-extra
  Downloading scikit_learn_extra-0.2.0-cp37-cp37m-manylinux2010_x86_64.whl (1.7 MB)
     |████████████████████████████████| 1.7 MB 3.8 MB/s
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-packages (from scikit-learn-extra) (1.21.6)
Requirement already satisfied: scikit-learn>=0.23.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn-extra) (1.0.2)
Requirement already satisfied: scipy>=0.19.1 in /usr/local/lib/python3.7/dist-packages (from scikit-learn-extra) (1.4.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.23.0->scikit-learn-extra) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.23.0->scikit-learn-extra) (3.1.0)
Installing collected packages: scikit-learn-extra
Successfully installed scikit-learn-extra-0.2.0
```

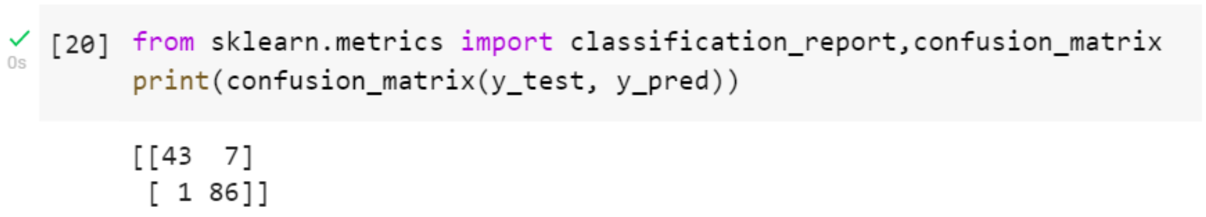
7) BUILDING MODEL OF K-MEDIOD & PREDICTION OF BREAST CANCER:



```
from sklearn_extra.cluster import KMedoids

kmedoids = KMedoids(n_clusters=2, random_state=0).fit(X_train)
y_pred = kmedoids.predict(X_test)
y_pred
```

```
array([1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1,
       1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
       0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       0, 1, 1, 0, 1])
```

8) CONFUSION MATRIX:



```
[20] from sklearn.metrics import classification_report,confusion_matrix
     print(confusion_matrix(y_test, y_pred))
```

```
[[43  7]
 [ 1 86]]
```

CONCLUSION:

From this practical, I have learned the implementation of k-medoid in python.