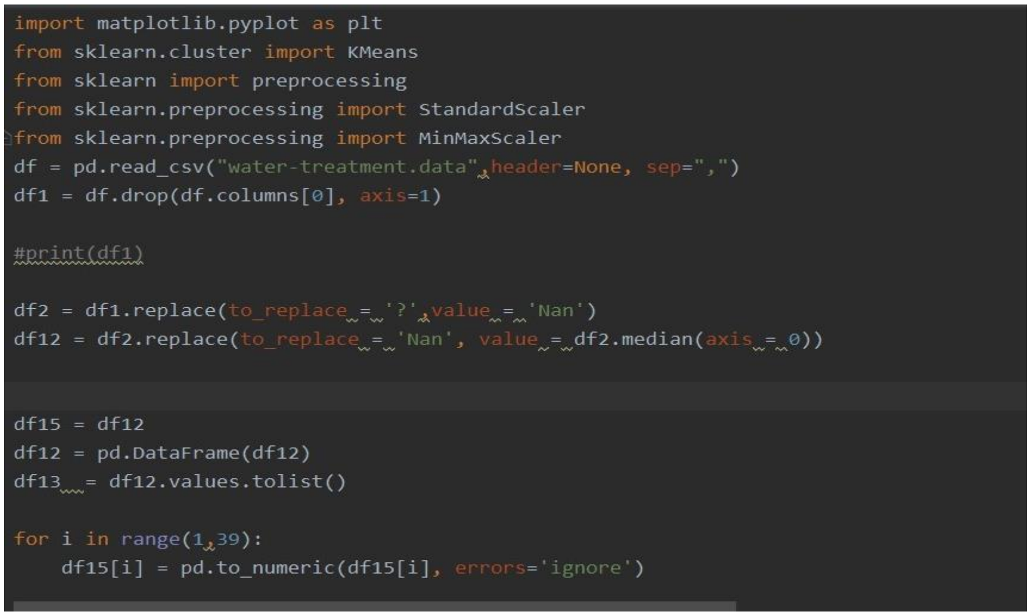
1. (20 pts.) Clean up the data set. This includes filling up the missing values and normalizing all the data items. Please state clearly the methods you use for filling up the missing values and normalizing the values in English to answer this question.
Ans – In this program I have taken following steps to fill missing values and getting normalize data

a) Firstly, I readout the data in pandas dataframe through "read_csv" function and removed header and " , ".
b) Secondly, I replaced "?" within dataset with Nan values using "replace" function of python3.
c) Then I searched for the median along with "axis= 0" then I replaced all the Nan with respective median of columns.
d) For normalization I used MinMaxScaler function from sklearn.preprocessing package.

   Which uses following formula to provide the normalized result.

   X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
   X_scaled = X_std * (max - min) + min

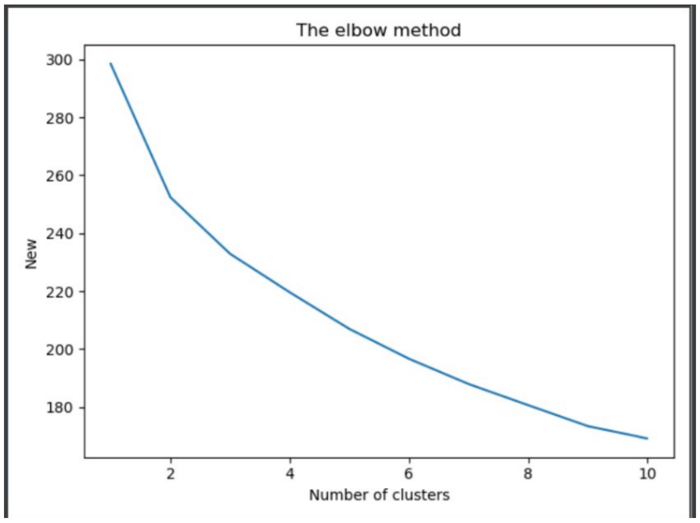


```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
df = pd.read_csv("water-treatment.data",header=None, sep=",")
df1 = df.drop(df.columns[0], axis=1)

#print(df1)

df2 = df1.replace(to_replace = '?',value = 'Nan')
df12 = df2.replace(to_replace = 'Nan', value = df2.median(axis = 0))


df15 = df12
df12 = pd.DataFrame(df12)
df13 = df12.values.tolist()

for i in range(1,39):
    df15[i] = pd.to_numeric(df15[i], errors='ignore')
```

2. (20 pts.) It is well-known that the k-means algorithm requires that the number of clusters, k, be given in advance. In this problem, we do not know the k value in advance. Propose a specific termination condition for the modified k-means when searching the true k value. State clearly your proposed condition or method in English.
Ans –

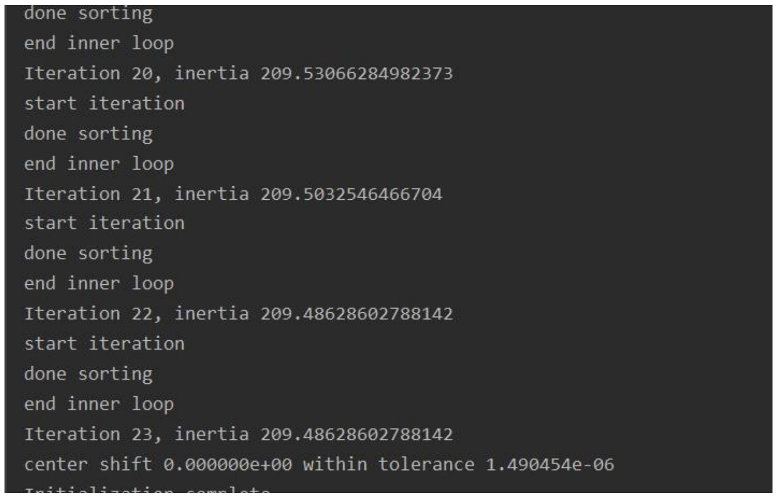
a) In the normal k-mean algorithm we required to have the number of clusters k to be known in advance. Here we have not provided with number of clusters. To find k there are multiple methonds like – Elbow method, Average silhouette method, Gap statistic method etc.

b) Over here I used the most efficient method known as Elbow method to find the optimal number of clusters. Which provided the graph as,



Here we can see the formation of elbow(like human elbow) through this I have taken 5 as the optimal number of clusters as after 5 the graph declining sharply.

c) One of the greatest challenges in k-means clustering fetching the clusters and centroids, as close to optimal as possible, also keeping the time in the consideration. Traditional K-means uses the randomize approach to centroid initialization that takes significant time and iterations.

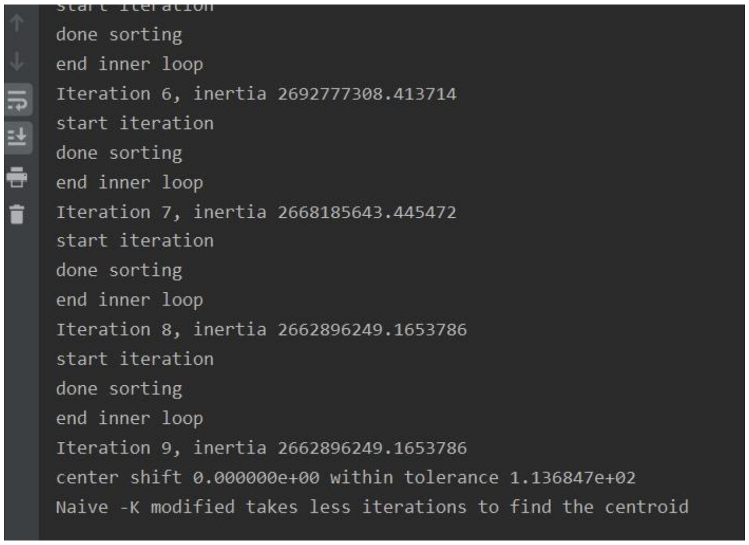


```
done sorting
end inner loop
Iteration 20, inertia 209.53066284982373
start iteration
done sorting
end inner loop
Iteration 21, inertia 209.5032546466704
start iteration
done sorting
end inner loop
Iteration 22, inertia 209.48628602788142
start iteration
done sorting
end inner loop
Iteration 23, inertia 209.48628602788142
center shift 0.000000e+00 within tolerance 1.490454e-06
```

Here we can see traditional k-means is taking 23 iterations to initialize the centroid.

d) To reduce the time by significant amount I used naïve-shard k-means algorithm as the modified algorithm. For this first step was too find the sum of all the columns (attributes). Then sort it into the ascending order after that split that list into equal part for that I have taken “42”. Then searched for the mean for centroid initialization. After that added centroid from each row to get the new set of centroids. At last used the same set for the k-means clustering.

e) Through this I was able to reduce the iterations to 9 –



```
start iteration
done sorting
end inner loop
Iteration 6, inertia 2692777308.413714
start iteration
done sorting
end inner loop
Iteration 7, inertia 2668185643.445472
start iteration
done sorting
end inner loop
Iteration 8, inertia 2662896249.1653786
start iteration
done sorting
end inner loop
Iteration 9, inertia 2662896249.1653786
center shift 0.000000e+00 within tolerance 1.136847e+02
Naive -K modified takes less iterations to find the centroid
```

3. (20 pts.) Implement the modified k-means algorithm with your proposed termination condition and run the algorithm using the water-treatment dataset. Please note that you must use the output format given in the description file. Report your output.

Ans –

a) After finding the new centroids through modified k-means using the normalized data given.

b) To view clustering of each vector(row) kindly check the output file named as KmeansOutput.txt in this folder.

4. (20 pts.) Apply the PCA method you implemented in the first assignment to this dataset. Then apply the implemented modified k-means method above to this reduced data set to report the output. Please follow the same protocol of the output format specified in the description file.

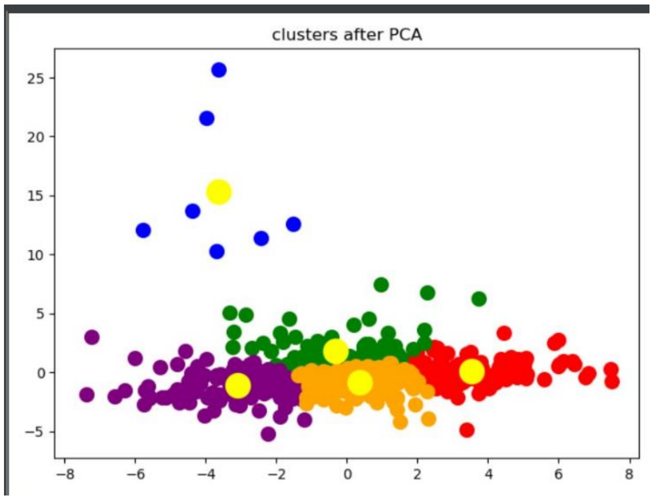
Ans –

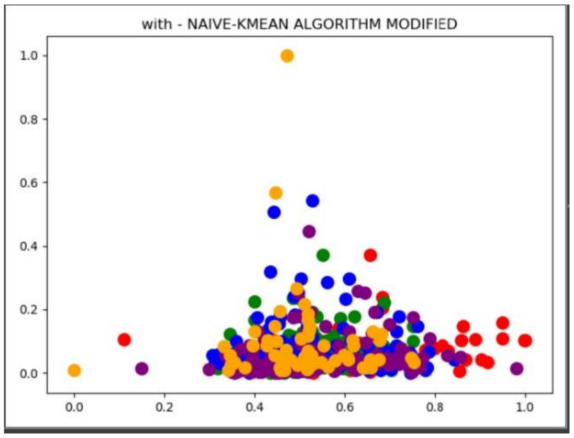
a) Then as like the first assignment I went through the all procedure to implementation of the PCA like finding covariance and correlation between the data. After that I plotted the curve of variance vs the principal components. After that I choose 4 components to get good feature with accurate data.
b) Then I applied the modified k-means on the result of PCA dataset.
c) For the output kindly check the KmeansOutputafterPCA.txt file in this folder.

5. (20 pts.) Compare the two clustering results and analyze any differences that you have observed and state why there is such difference if there is or why there is no difference if there is no.

Ans –

a) I applied modified K-means on the data which I got through PCA analysis.
b) Which unsurprisingly, gave me better results as originally we had 38 dimensional data and it's proven fact that multidimensionality is curse to the k-means algorithm.
c) Although advantage of K-means is that it has the higher speed the accuracy is also needs to be taken into the consideration.
d) After applying PCA we have reduced the dimension to 4 then obviously we can see better clustering results.





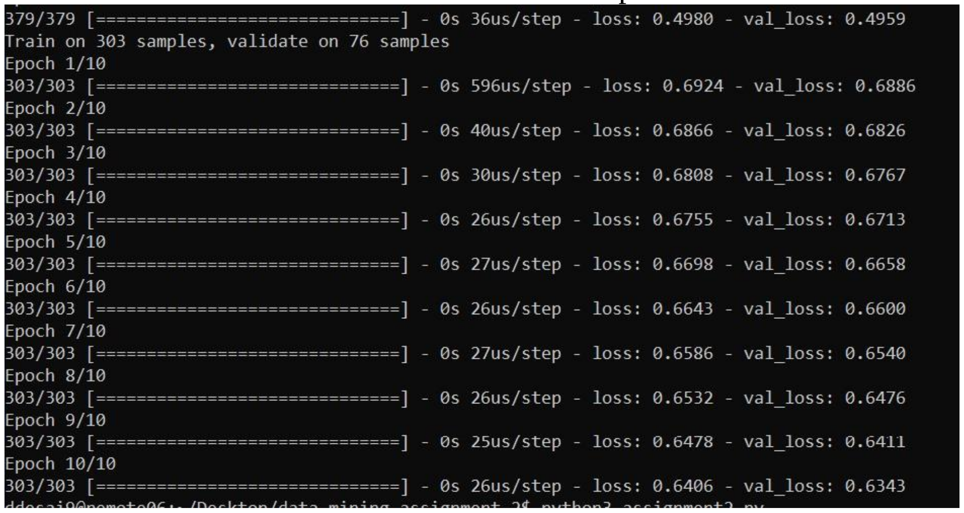
e) We can see at the right hand side we have got the distorted graph of clusters and on the left hand side we have got the better graph using the K-means after PCA analysis. Hence PCA with K-means is the good combination as it reduce the curse of dimensionality.

6. (50 pts.) Implement an autoencoder (either shallow or deep) for dimensionality reduction and apply the implemented autoencoder to the given dataset. Report the dimensionality reduction result using the autoencoder and discuss the difference between PCA and autoencoder for dimensionality reduction with this dataset.

Ans –

a) I have applied the deep autoencoder to get the feature extraction done. With deep auto encoder I was able to reduce the loss inside the data and get the better results.
b) PCA basically works better with higher number of components and Auto encoder works better with lower number of components.
c) While in PCA only 2 or 3 components are used and all the rest components are hidden hence the accuracy has to be compromised. Where as, In the auto encoders can have the full data at the 2 or 3 dimensions which contains all the information and saves the time.
d) PCA is having the memory limitation where as, we can implement auto encoder with keras easily.

e) Autoencoder has the lower loss function as compared to PCA.



```
379/379 [==============================] - 0s 36us/step - loss: 0.4980 - val_loss: 0.4959
Train on 303 samples, validate on 76 samples
Epoch 1/10
303/303 [==============================] - 0s 596us/step - loss: 0.6924 - val_loss: 0.6886
Epoch 2/10
303/303 [==============================] - 0s 40us/step - loss: 0.6866 - val_loss: 0.6826
Epoch 3/10
303/303 [==============================] - 0s 30us/step - loss: 0.6808 - val_loss: 0.6767
Epoch 4/10
303/303 [==============================] - 0s 26us/step - loss: 0.6755 - val_loss: 0.6713
Epoch 5/10
303/303 [==============================] - 0s 27us/step - loss: 0.6698 - val_loss: 0.6658
Epoch 6/10
303/303 [==============================] - 0s 26us/step - loss: 0.6643 - val_loss: 0.6600
Epoch 7/10
303/303 [==============================] - 0s 27us/step - loss: 0.6586 - val_loss: 0.6540
Epoch 8/10
303/303 [==============================] - 0s 26us/step - loss: 0.6532 - val_loss: 0.6476
Epoch 9/10
303/303 [==============================] - 0s 25us/step - loss: 0.6478 - val_loss: 0.6411
Epoch 10/10
303/303 [==============================] - 0s 26us/step - loss: 0.6406 - val_loss: 0.6343
```

Above is the result of auto encoder on the data.