Interactive exercise: Clustering

Liping Wu 300-958-061 11/18/2020

In this exercise, we will do the following:

- Explore a dataset
- Visualize the clusters using matplotlib and seaborn
- Build a clustering model using K-means clustering algorithm

Note: You need to do step 10 before you leave the lab and present the results to your professor before leaving to earn any grades for this lab.

Pre-requisites:

1- Install Anoconda
2- We will be using a lot of Public datasets these datasets are available at https://goo.gl/zjS4C6 under a folder named "Datasets for Predictive Modelling with Python", the datasets are organized in the order of the text book chapters: Python: Advanced Predictive Analytics, chapter # 7 files are required

Steps for exploring and building a logistic regression model:

1- Open your spyder IDE
2- Load the 'wine.csv' file into a dataframe name *the dataframe data_firstname_wine where first name is your first name* carry out the following activities:
   a. Display the column names
   b. Display the shape of the data frame i.e number of rows and number of columns
   c. Display the main statistics of the data
   d. Display the types of columns
   e. Display the first five records
   f. Find the unique values of the quality attribute
   g. Find the mean of the various chemical compositions across samples for the different groups of the wine quality

Following is the code, *make sure you update the path to the correct path where you placed the files and update the data frame name correctly*:

```
import pandas as pd
import os
path = "D:/CentennialWu/2020Fall/COMP309Data/Assignments/Lab08Wk10/"
filename = 'wine.csv'
fullpath = os.path.join(path,filename)
data_liping_wine = pd.read_csv(fullpath,sep=';')
```

```
# set the columns display and check the data
pd.set_option('display.max_columns',15)
print(data_liping_wine.head())
print(data_liping_wine.columns.values)
print(data_liping_wine.shape)
print(data_liping_wine.describe())
print(data_liping_wine.dtypes)
print(data_liping_wine.head(5))
print(data_liping_wine['quality'].unique())
pd.set_option('display.max_columns',15)
print(data_liping_wine.groupby('quality').mean())
```

```
In [74]: print(data_liping_wine.head())
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5

In [75]: print(data_liping_wine.columns.values)
['fixed acidity' 'volatile acidity' 'citric acid' 'residual sugar'
 'chlorides' 'free sulfur dioxide' 'total sulfur dioxide' 'density' 'pH'
 'sulphates' 'alcohol' 'quality']

In [76]: print(data_liping_wine.shape)
(1599, 12)
```

```
In [77]: print(data_liping_wine.describe())
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    1599.000000       1599.000000  1599.000000     1599.000000
mean        8.319637          0.527821     0.270976        2.538806
std         1.741096          0.179060     0.194801        1.409928
min         4.600000          0.120000     0.000000        0.900000
25%         7.100000          0.390000     0.090000        1.900000
50%         7.900000          0.520000     0.260000        2.200000
75%         9.200000          0.640000     0.420000        2.600000
max        15.900000          1.580000     1.000000       15.500000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  1599.000000          1599.000000           1599.000000  1599.000000
mean      0.087467            15.874922             46.467792     0.996747
std       0.047065            10.460157             32.895324     0.001887
min       0.012000             1.000000              6.000000     0.990070
25%       0.070000             7.000000             22.000000     0.995600
50%       0.079000            14.000000             38.000000     0.996750
75%       0.090000            21.000000             62.000000     0.997835
max       0.611000            72.000000            289.000000     1.003690

                pH    sulphates      alcohol      quality
count  1599.000000  1599.000000  1599.000000  1599.000000
mean      3.311113     0.658149    10.422983     5.636023
std       0.154386     0.169507     1.065668     0.807569
min       2.740000     0.330000     8.400000     3.000000
25%       3.210000     0.550000     9.500000     5.000000
50%       3.310000     0.620000    10.200000     6.000000
75%       3.400000     0.730000    11.100000     6.000000
max       4.010000     2.000000    14.900000     8.000000

In [78]: print(data_liping_wine.dtypes)
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide     float64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
alcohol                 float64
quality                   int64
dtype: object
```

```
In [72]: print(data_liping_wine.groupby('quality').mean())
         fixed acidity  volatile acidity  citric acid  residual sugar  \
quality
3             8.360000          0.884500     0.171000        2.635000
4             7.779245          0.693962     0.174151        2.694340
5             8.167254          0.577041     0.243686        2.528855
6             8.347179          0.497484     0.273824        2.477194
7             8.872362          0.403920     0.375176        2.720603
8             8.566667          0.423333     0.391111        2.577778

         chlorides  free sulfur dioxide  total sulfur dioxide   density  \
quality
3         0.122500            11.000000             24.900000  0.997464
4         0.090679            12.264151             36.245283  0.996542
5         0.092736            16.983847             56.513950  0.997104
6         0.084956            15.711599             40.869906  0.996615
7         0.076588            14.045226             35.020101  0.996104
8         0.068444            13.277778             33.444444  0.995212

                pH  sulphates    alcohol
quality
3         3.398000   0.570000   9.955000
4         3.381509   0.596415  10.265094
5         3.304949   0.620969   9.899706
6         3.318072   0.675329  10.629519
7         3.290754   0.741256  11.465913
8         3.267222   0.767778  12.094444
```

Some observations

- *The lesser the volatile acidity and chlorides, the higher the wine quality*
- *The more the sulphates and citric acid content, the higher the wine quality*
- *The density and pH don't vary much across the wine quality*

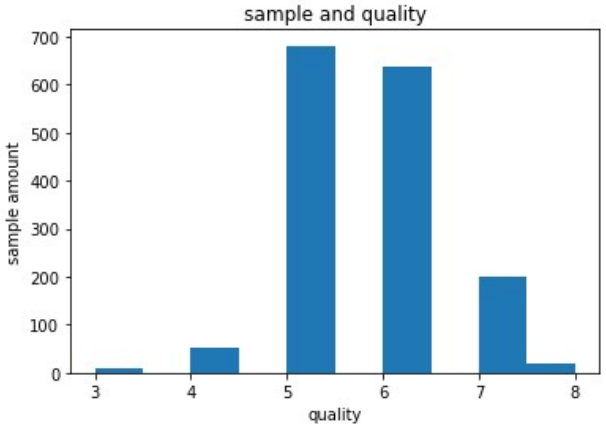
3- Plot a histogram to see the number of wine samples in each quality type

Following is the code, *make sure you update the the data frame name correctly*:

```
import matplotlib.pyplot as plt
plt.hist(data_liping_wine['quality'])
```

```
In [87]: plt.hist(data_liping_wine['quality'],bins=10)
Out[87]:
(array([ 10.,   0.,  53.,   0., 681.,   0., 638.,   0., 199.,  18.]),
 array([3. , 3.5, 4. , 4.5, 5. , 5.5, 6. , 6.5, 7. , 7.5, 8. ]),
 <a list of 10 Patch objects>)
```

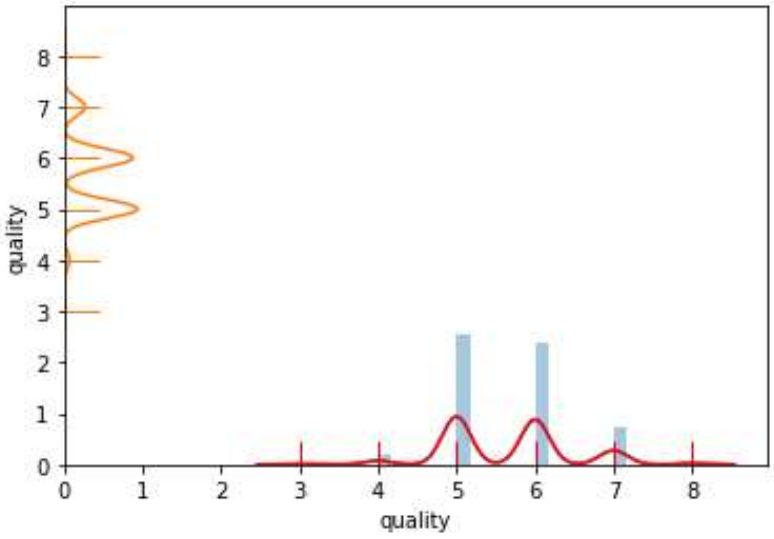
```
In [88]: plt.xlabel('quality')
    ...: plt.ylabel('sample amount')
    ...: plt.title('sample and quality')
Out[88]: Text(0.5, 1.0, 'sample and quality')
```



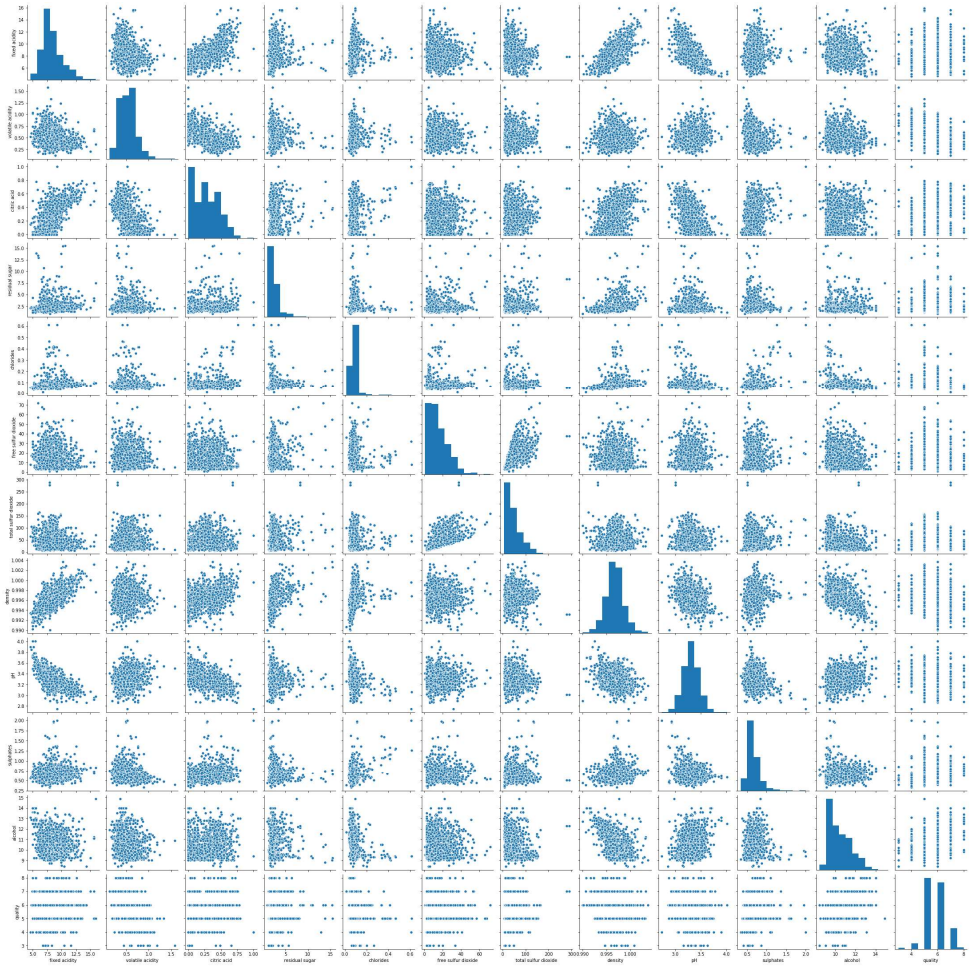
4- Use seaborn library to generate different plots: histograms, pairplots, heatmaps…etc. and investigate the correlations.

Following are the code snippets, *make sure you update the data frame name correctly*:

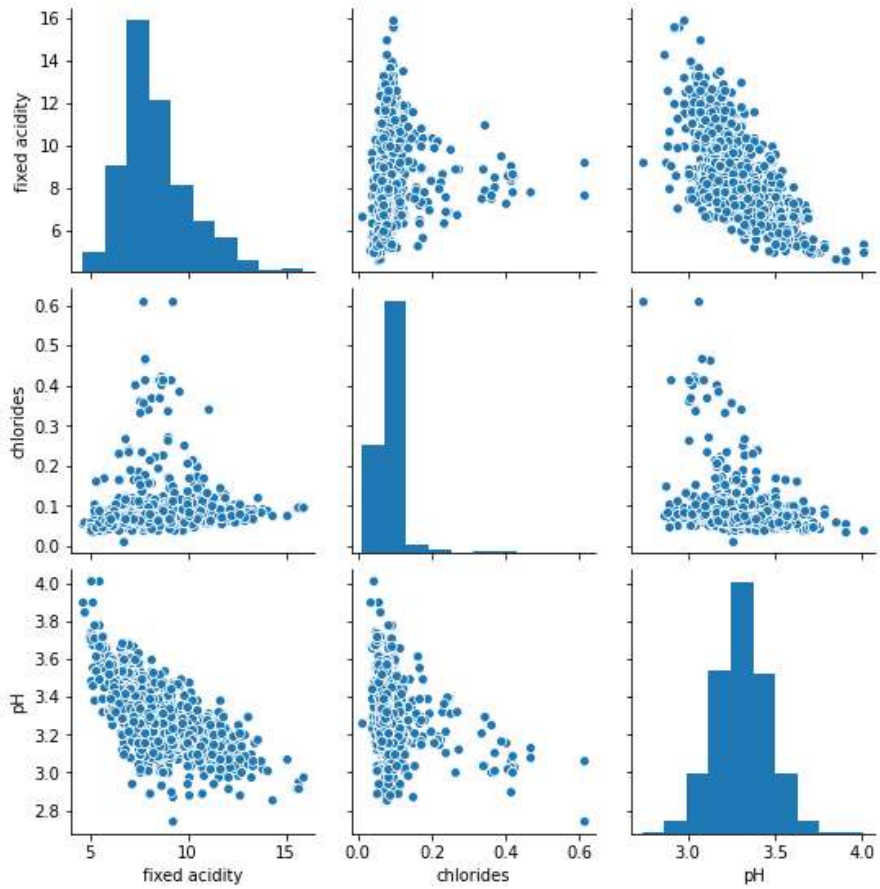
```
import seaborn as sns
sns.distplot(data_liping_wine['quality'])

# plot only the density function
sns.distplot(data_liping_wine['quality'], rug=True, hist=False, color = 'r')
# Change the direction of the plot
sns.distplot(data_liping_wine['quality'], rug=True, hist=False, vertical = True)
```



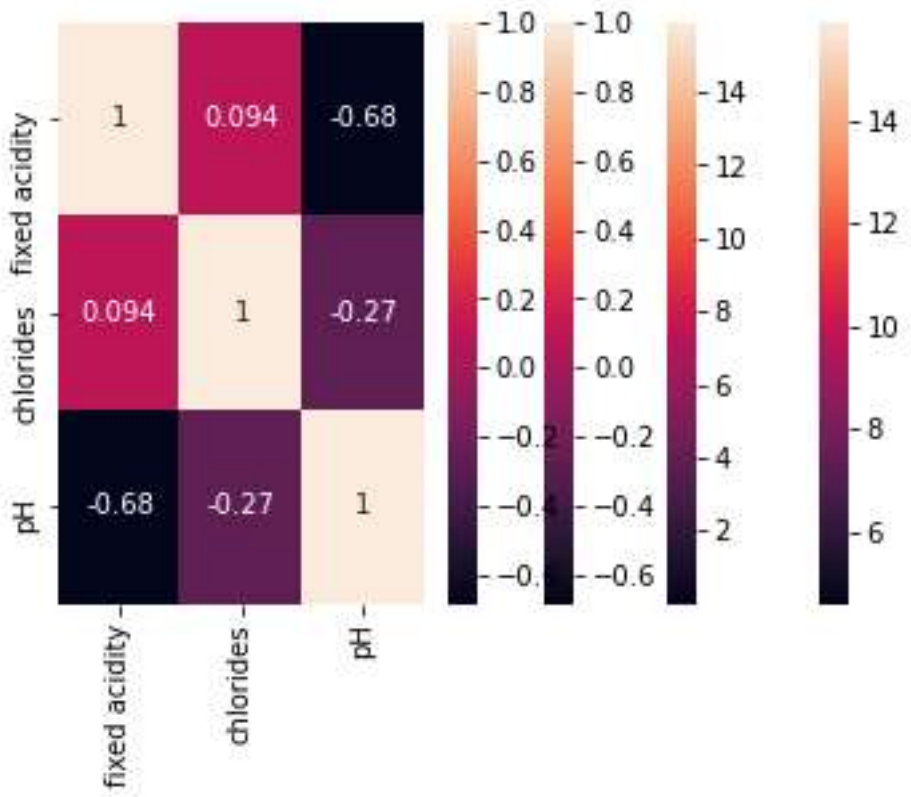
```
# Check all correlations
sns.pairplot(data_liping_wine)
```



```
# Subset three column
x=data_liping_wine[['fixed acidity','chlorides','pH']]
y=data_liping_wine[['chlorides','pH']]
# check the correlations
sns.pairplot(x)
```

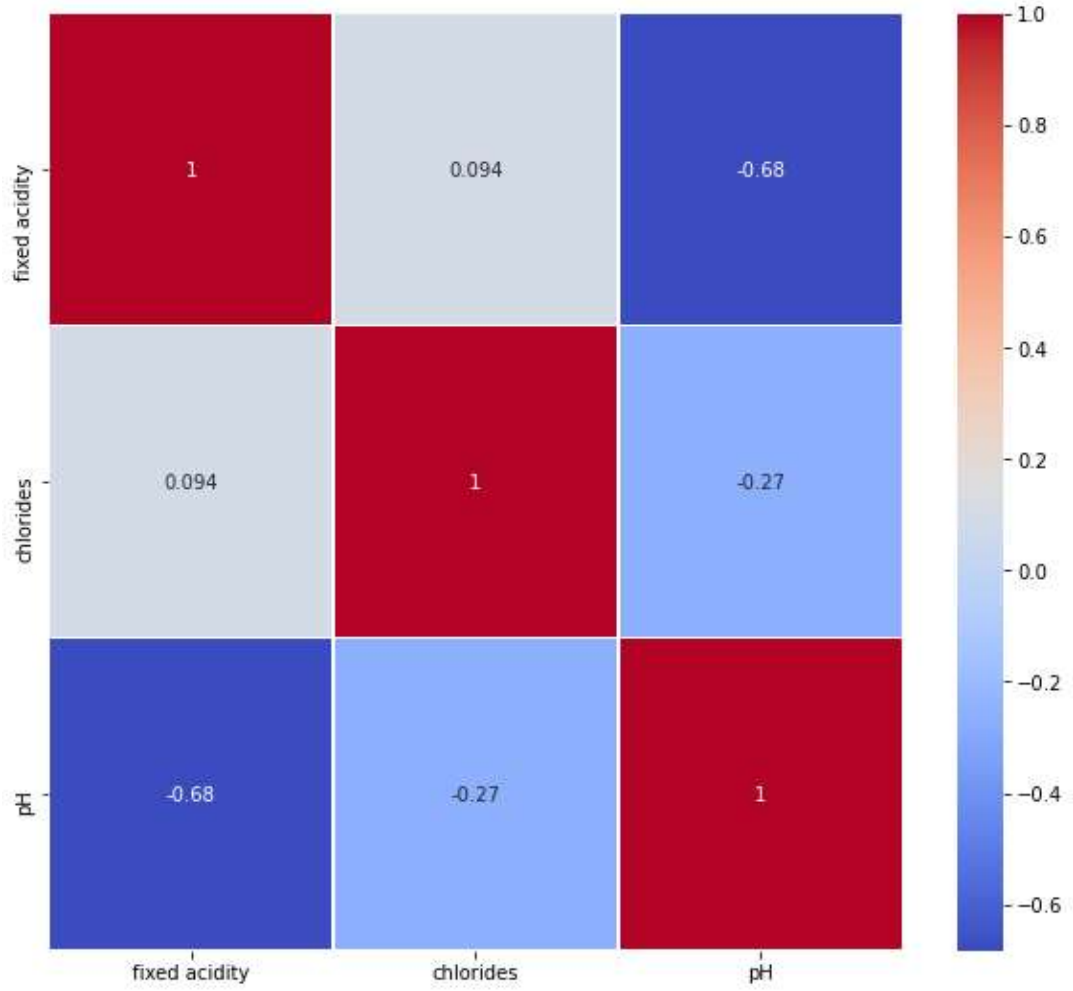



```
# Generate heatmaps
sns.heatmap(data_liping_wine[['fixed acidity']])
sns.heatmap(x)
sns.heatmap(x.corr())
sns.heatmap(x.corr(),annot=True)
```

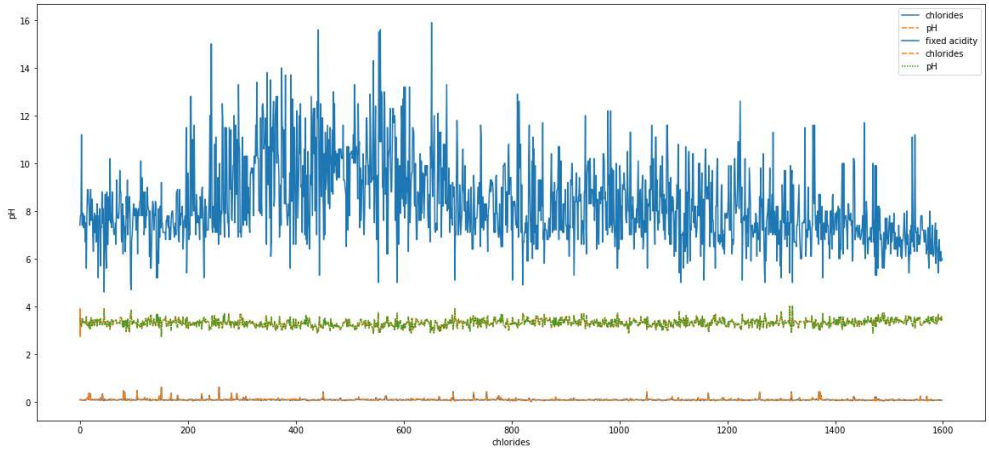


##

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,9))
sns.heatmap(x.corr(),annot=True, cmap='coolwarm',linewidth=0.5)
```




```
##line two variables
plt.figure(figsize=(20,9))
sns.lineplot(data=y)
sns.lineplot(data=y,x='chlorides',y='pH')
## line three variables
sns.lineplot(data=x)
```



5- Normalize the data in order to apply clustering, the formula is as follows:

$$Z_i = (X_i - X_{min}) / (X_{max} - X_{min})$$

Following is the code, *make sure you update model name correctly*:

```
data_liping_wine_norm = (data_liping_wine - data_liping_wine.min()) /
(data_liping_wine.max() - data_liping_wine.min())
data_liping_wine_norm.head()
```

The output should look like this

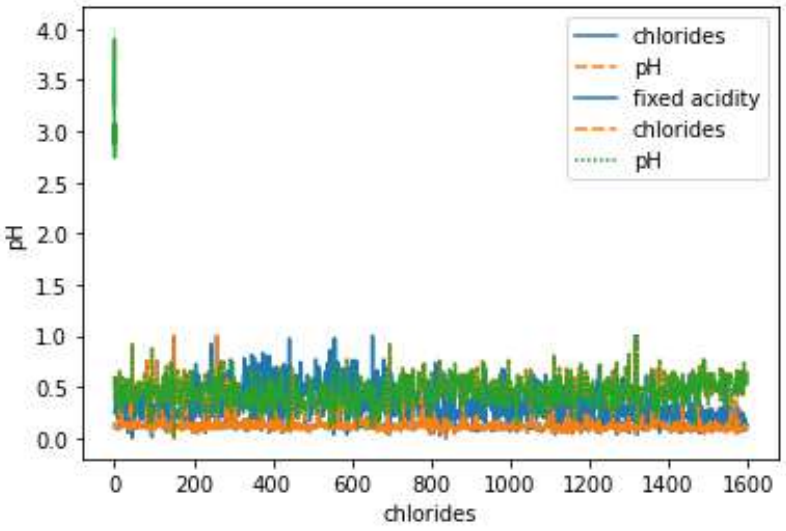
data_liping_wine_norm - DataFrame

Index	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	ee sulfur dioxic	otal sulfur dioxic	density	pH	sulphates	alcohol	quality
0	0.247788	0.39726	0	0.0684932	0.106845	0.140845	0.0989399	0.567548	0.606299	0.137725	0.153846	0.4
1	0.283186	0.520548	0	0.116438	0.143573	0.338028	0.215548	0.494126	0.362205	0.209581	0.215385	0.4
2	0.283186	0.438356	0.04	0.0958904	0.133556	0.197183	0.169611	0.508811	0.409449	0.191617	0.215385	0.4
3	0.584071	0.109589	0.56	0.0684932	0.105175	0.225352	0.190813	0.582232	0.330709	0.149701	0.215385	0.6
4	0.247788	0.39726	0	0.0684932	0.106845	0.140845	0.0989399	0.567548	0.606299	0.137725	0.153846	0.4
5	0.247788	0.369863	0	0.0616438	0.105175	0.169014	0.120141	0.567548	0.606299	0.137725	0.153846	0.4
6	0.292035	0.328767	0.06	0.0479452	0.0951586	0.197183	0.187279	0.464758	0.440945	0.0778443	0.153846	0.4
7	0.238938	0.363014	0	0.0205479	0.0884808	0.197183	0.0530035	0.332599	0.511811	0.0838323	0.246154	0.8

6- Generate some additional plots for the normalized data:

Following is the code, *make sure you update model name correctly*:

```
# check some plots after normalizing the data
x1=data_liping_wine_norm[['fixed acidity','chlorides','pH']]
y1=data_liping_wine_norm[['chlorides','pH']]
sns.lineplot(data=y1)
sns.lineplot(data=x1)
sns.lineplot(data=y,x='chlorides',y='pH')
```



7- Cluster the data (observations) into 6 clusters using k-means clustering algorithm.

8- Following is the code, *make sure you update model name correctly*:

```
from sklearn.cluster import KMeans
from sklearn import datasets
model=KMeans(n_clusters=6)
model.fit(data_liping_wine_norm)
```

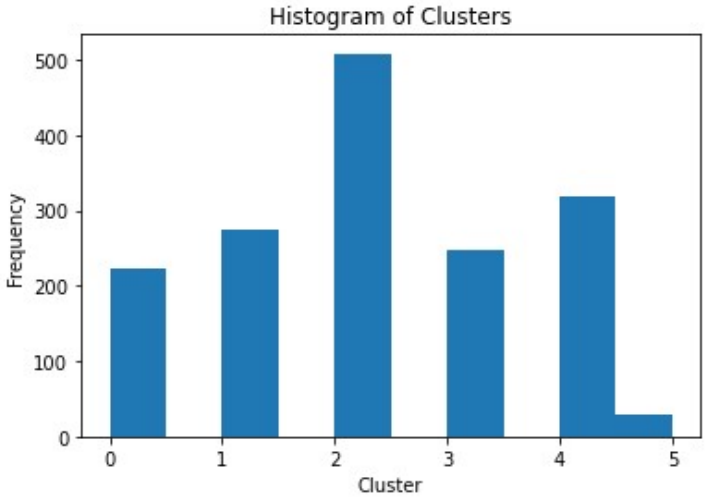
9- Check the results as follows:

a. Print the model labels
b. Append the clusters to each record on the dataframe, i.e. add a new column for clusters
c. find the final cluster's centroids for each cluster
d. Calculate the J-scores The J-score can be thought of as the sum of the squared distance between points and cluster centroid for each point and cluster. For an efficient cluster, the J-score should be as low as possible.
e. plot a histogram for the clusters variable to get an idea of the number of observations in each cluster.

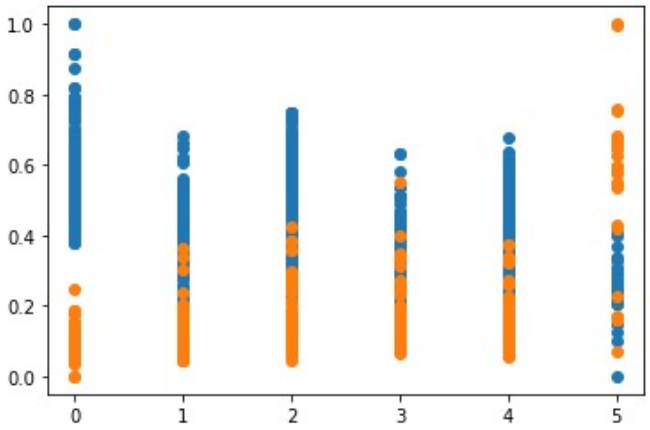
Following is the code, *make sure you update model name correctly*:

```
model.labels_
# Append the clusters to each record on the dataframe, i.e. add a new column for clusters
md=pd.Series(model.labels_)
data_liping_wine_norm['clust']=md
data_liping_wine_norm.head(10)
#find the final cluster's centroids for each cluster
model.cluster_centers_
```

#Calculate the J-scores The J-score can be thought of as the sum of the squared distance between points and cluster centroid for each point and cluster.
#For an efficient cluster, the J-score should be as low as possible.
model.inertia_



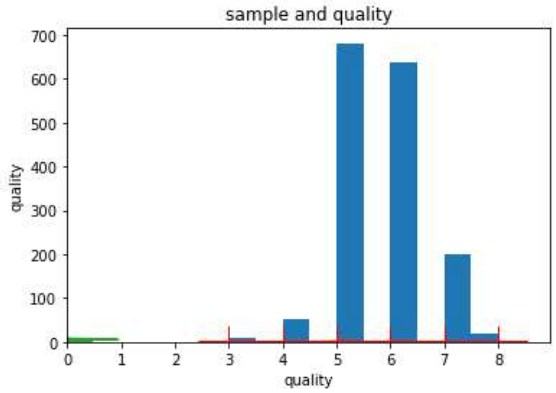
#let us plot a histogram for the clusters
import matplotlib.pyplot as plt
plt.hist(data_liping_wine_norm['clust'])
plt.title('Histogram of Clusters')
plt.xlabel('Cluster')
plt.ylabel('Frequency')
# plot a scatter
plt.scatter(data_liping_wine_norm['clust'],data_liping_wine_norm['pH'])
plt.scatter(data_liping_wine_norm['clust'],data_liping_wine_norm['chlorides'])

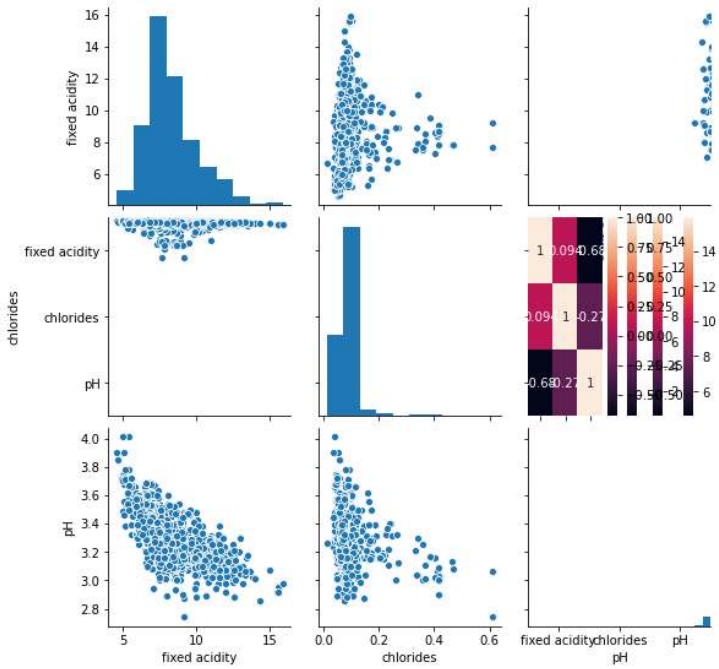


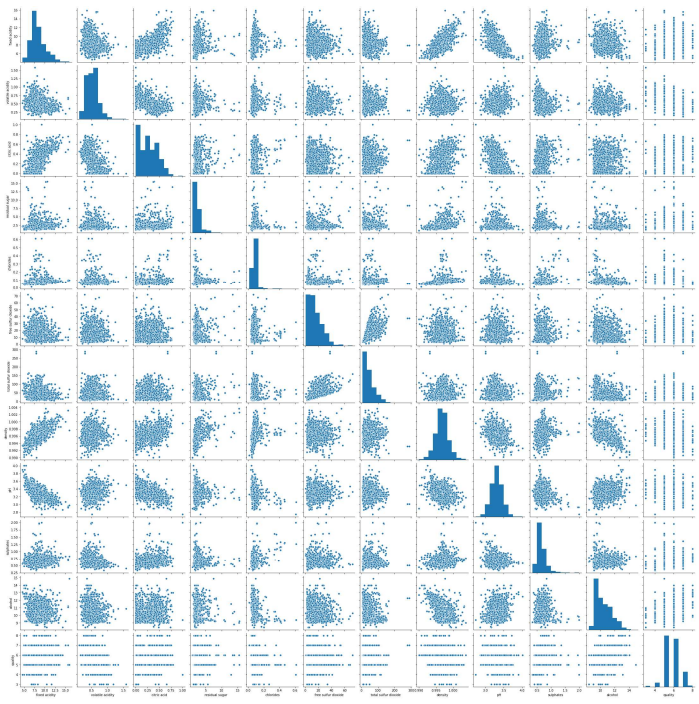
10- Re-cluster the data into three clusters and check the results. Show the results to your professor.

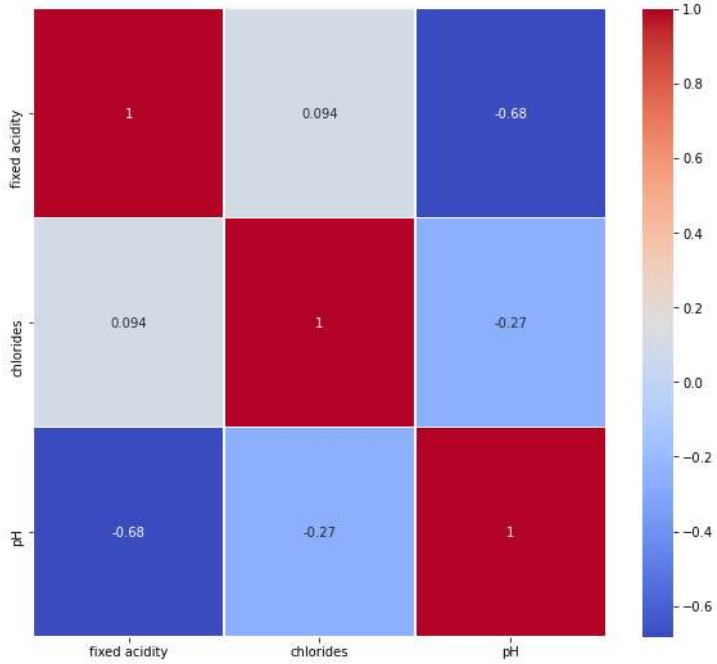
#Change n_cluster =6 for step7 and step8
from sklearn.cluster import KMeans
from sklearn import datasets

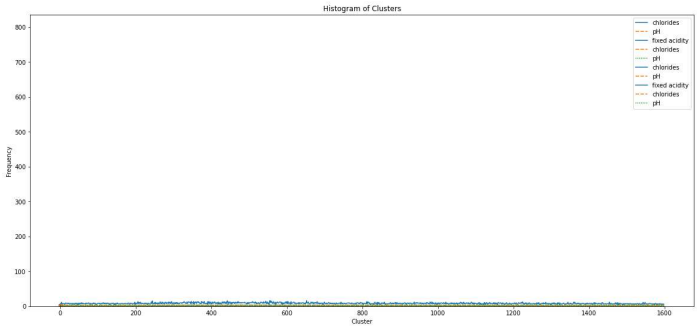
```
n_clusters=3
# n_clusters=6
model=KMeans(n_clusters)
model.fit(data_liping_wine_norm)

# Runnng results as following
```




fixed acidity
chlorides
pH
fixed acidity
chlorides
pH




1
0.094
-0.68
0.094
1
-0.27
-0.68
-0.27
1
fixed acidity
chlorides
pH
fixed acidity
chlorides
pH
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6


Histogram of Clusters
chlorides
pH
fixed acidity
chlorides
pH
chlorides
pH
fixed acidity
chlorides
pH
Frequency
Cluster
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000
1200
1400
1600