UL-Inclass-1

March 10, 2020

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from scipy.spatial import distance
from scipy.stats import zscore
```

In class Assignment Expectations/Steps -

Apply Data Cleaning to the Datasets and then apply Kmeans algorithm for find pattern and the best value of.the K for the following features.

1. Use feautes fixed acidity and volatile acidity
2. Use feautes Cirtic acidity and fixed acidity
3. Use feautes residual suger and sulphades
4. Use feautes free.sulfur.dioxide and total.sulfur.dioxide
5. Use feautes fixed acidity, citric acidity and volatile acidity
6. Use feautes density and pH

[2]:
```
df=pd.read_csv('winequality-red.csv')
df.head()
```

[2]:
```
   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
```

#we have read the data into df and we can see the first five columns of the data clearly which are all numerical

[3]:
```
df.shape
```

[3]:
```
(1599, 12)
```

#we see the shape of the data as there are 12 columns and 1599 number of rows

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed_acidity         1599 non-null   float64
 1   volatile_acidity      1599 non-null   float64
 2   citric_acid           1599 non-null   float64
 3   residual_sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free_sulfur_dioxide   1599 non-null   float64
 6   total_sulfur_dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

we can see all the columns are of the numerical data type

[5]:
```
df.isnull().sum()
```

[5]:
```
fixed_acidity           0
volatile_acidity        0
citric_acid             0
residual_sugar          0
chlorides               0
free_sulfur_dioxide     0
total_sulfur_dioxide    0
density                 0
pH                      0
```

```
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

we are here checking for the null values in the dataset and we can say that there are no null values in the data

[6]:
```
df1=df.copy()
df1.head()
```

[6]:
```
   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
```

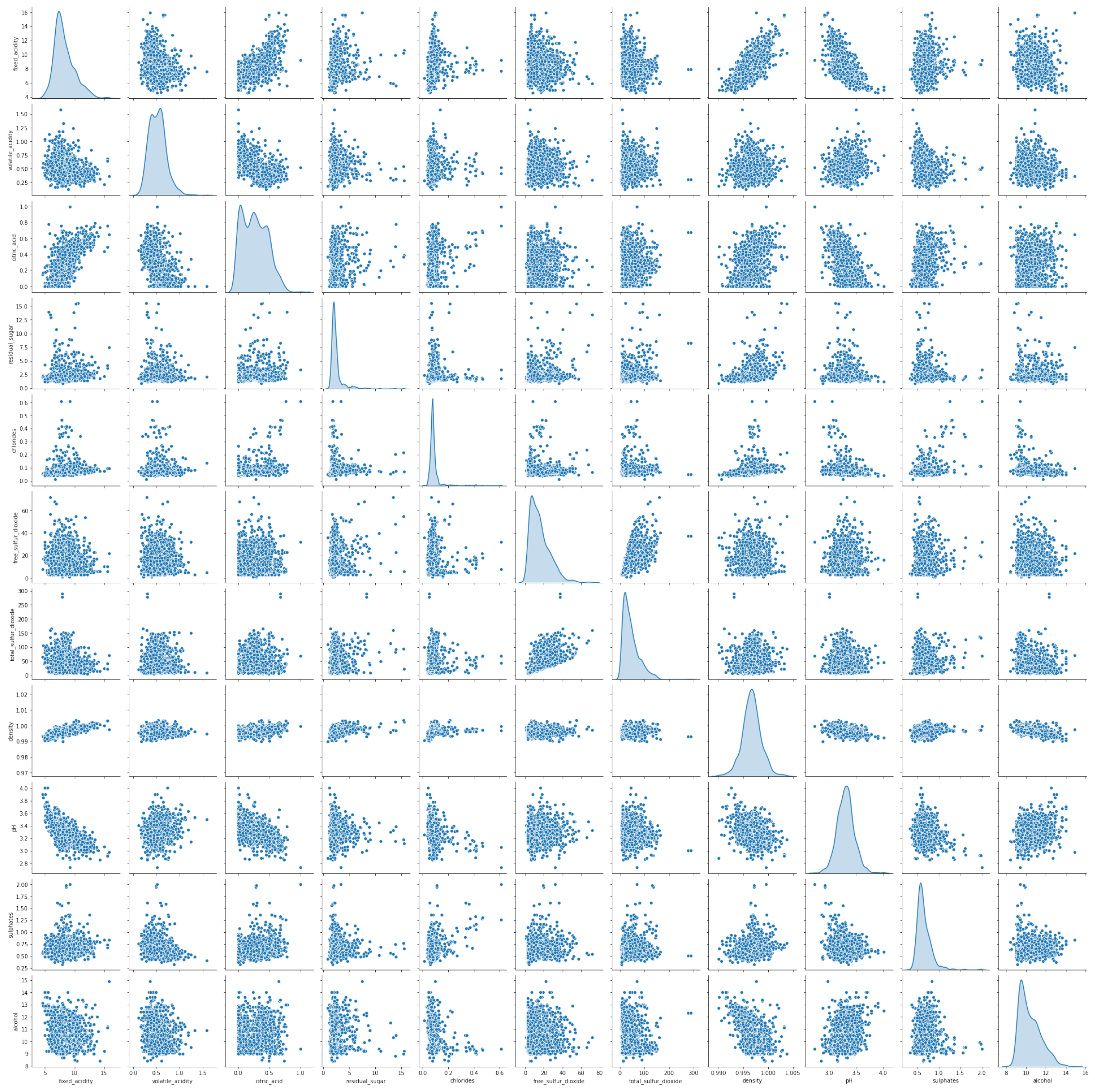
making a copy of the original data such that the operations done does not affect the original data

[7]:
```
df1.drop('quality', inplace=True, axis=1)
```

here, we are dropping the target column quality

[8]:
```
import seaborn as sns
sns.pairplot(df1,diag_kind='kde')
```

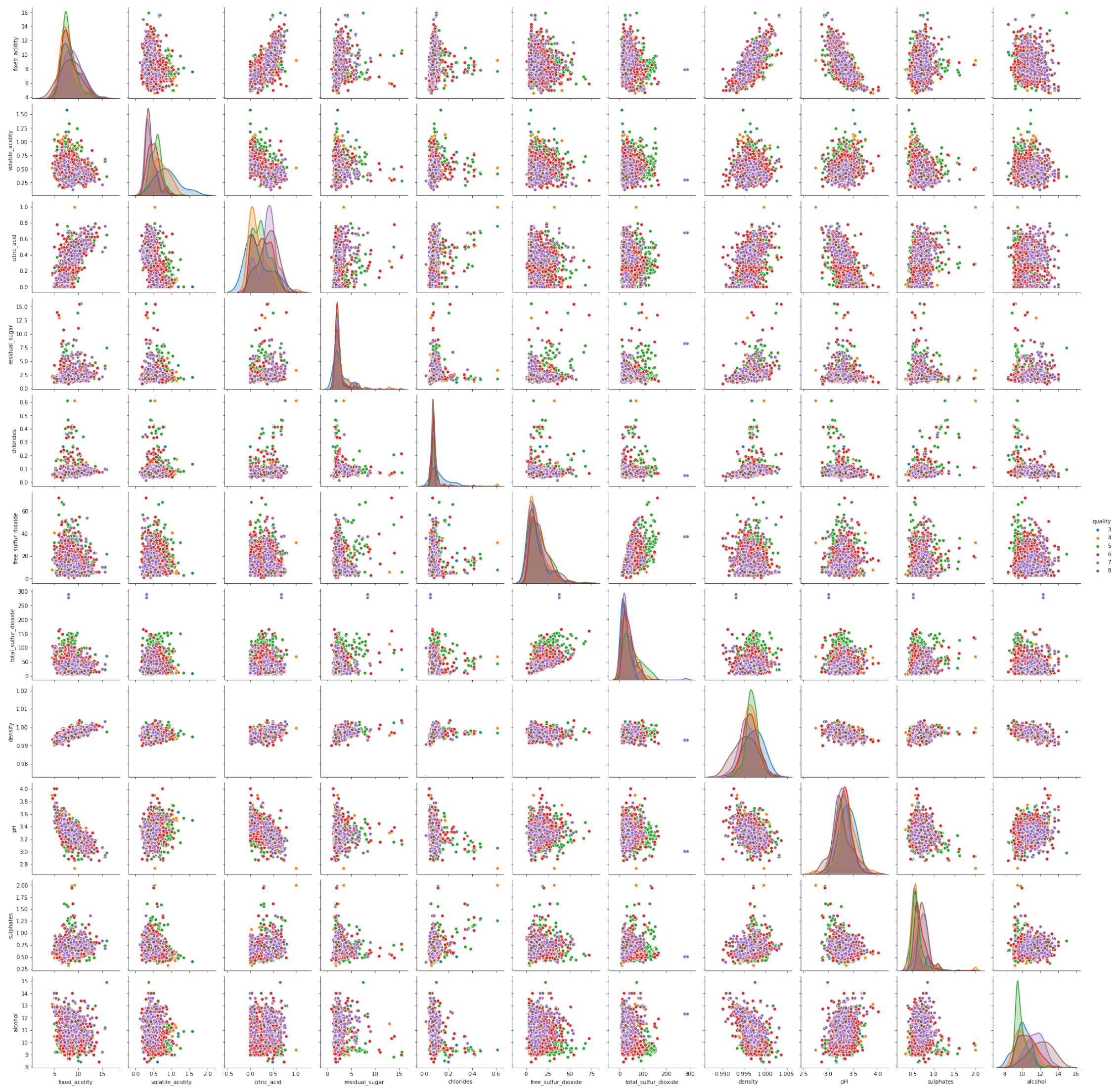
[8]:
```
<seaborn.axisgrid.PairGrid at 0x1638f678f60>
```



by seeing the pairplot we can see the kernal density plots and here if we see the small bumps on the curves which shows number of clusters

```
[41]: sns.pairplot(df,diag_kind='kde', hue='quality')
```

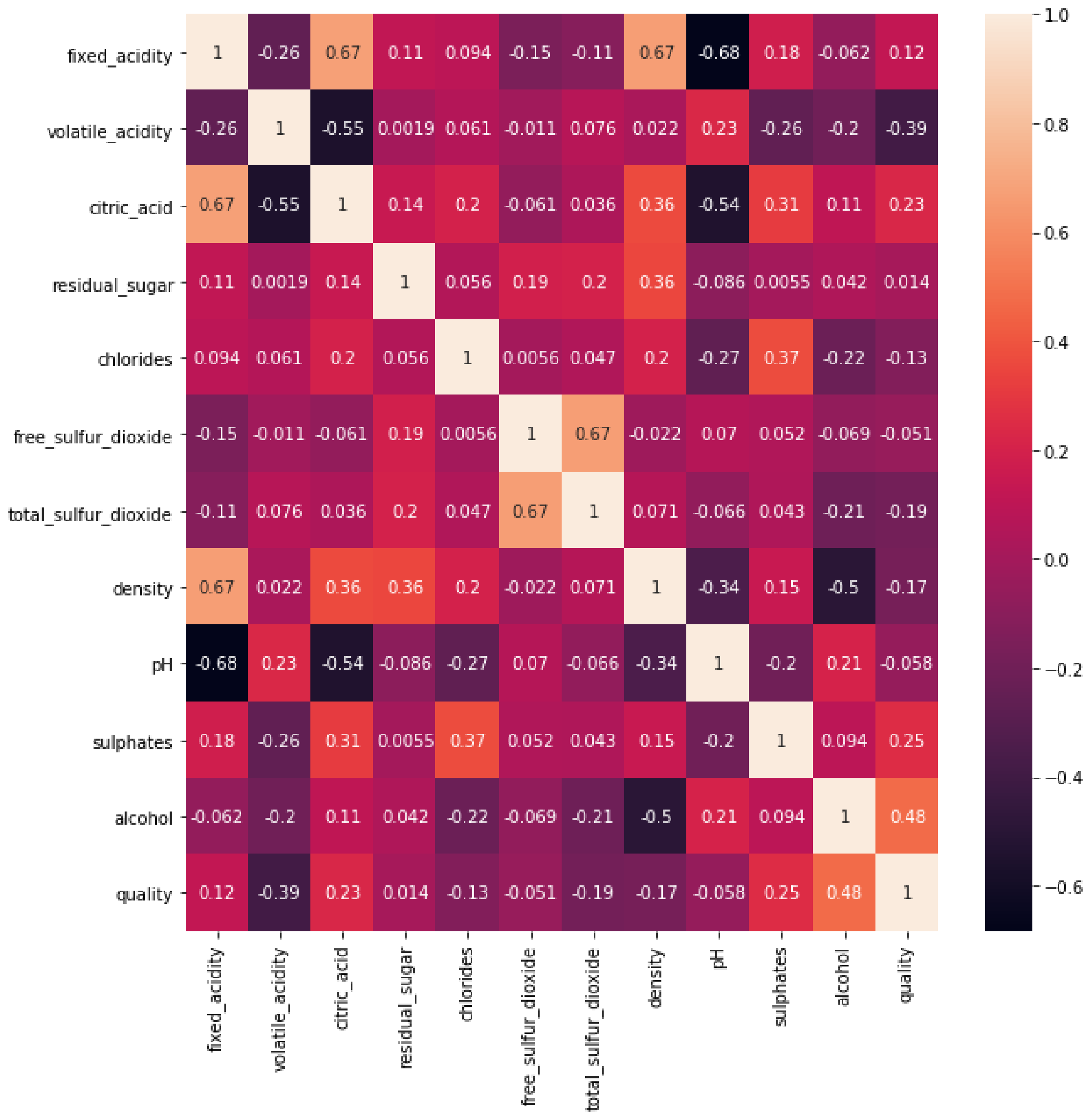
```
[41]: <seaborn.axisgrid.PairGrid at 0x16396b17550>
```



the pairplot includes the target variable where we can see the bumps on the curves as the number of clusters.

```
[15]: plt.figure(figsize=(10,10))
      sns.heatmap(df1.corr(), annot=True)
```

```
[15]: <matplotlib.axes._subplots.AxesSubplot at 0x26804021a90>
```



this is a corelation heat map of all the columns fixed acidity,citric acid and density, fixed acidity are having the highest positive corelation, citric acid, ph and fixed acidity,ph are having the strong negative corelation.

[15]:
```
df1_scaled = df1.apply(zscore)
df1_scaled.head()
```

[15]:
```
   fixed_acidity  volatile_acidity  citric_acid  residual_sugar  chlorides  \
0      -0.528360          0.961877    -1.391472       -0.453218  -0.243707
1      -0.298547          1.967442    -1.391472        0.043416   0.223875
2      -0.298547          1.297065    -1.186070       -0.169427   0.096353
3       1.654856         -1.384443     1.484154       -0.453218  -0.264960
```

```
4      -0.528360          0.961877    -1.391472       -0.453218  -0.243707

   free_sulfur_dioxide  total_sulfur_dioxide   density        pH  sulphates  \
0            -0.466193             -0.379133  0.558274  1.288643  -0.579207
1             0.872638              0.624363  0.028261 -0.719933   0.128950
2            -0.083669              0.229047  0.134264 -0.331177  -0.048089
3             0.107592              0.411500  0.664277 -0.979104  -0.461180
4            -0.466193             -0.379133  0.558274  1.288643  -0.579207

    alcohol
0 -0.960246
1 -0.584777
2 -0.584777
3 -0.584777
4 -0.960246
```

we are here standardizing the data using the z score

[16]:
```
model = KMeans(n_clusters = 3)
model
```

[16]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

we are building a simple model by using the number of clusters as 3

[25]:
```
cluster_range = range( 1, 15 )
cluster_errors = []
for num_clusters in cluster_range:
  clusters = KMeans( num_clusters, n_init = 10 )
  clusters.fit(df1_scaled)
  labels = clusters.labels_
  centroids = clusters.cluster_centers_
  cluster_errors.append( clusters.inertia_ )
clusters_df = pd.DataFrame( { "num_clusters":cluster_range, "cluster_errors":␣
 ↪cluster_errors } )
clusters_df[0:15]
```

[25]:
```
    num_clusters  cluster_errors
0              1    17589.000000
1              2    14330.119811
2              3    12629.906396
3              4    11294.409117
4              5    10155.374026
5              6     9362.868010
6              7     8644.994155
7              8     8299.254985
```

```
8              9     7966.196551
9             10     7699.666440
10            11     7451.519333
11            12     7215.884721
12            13     7064.082548
13            14     6815.347782
```

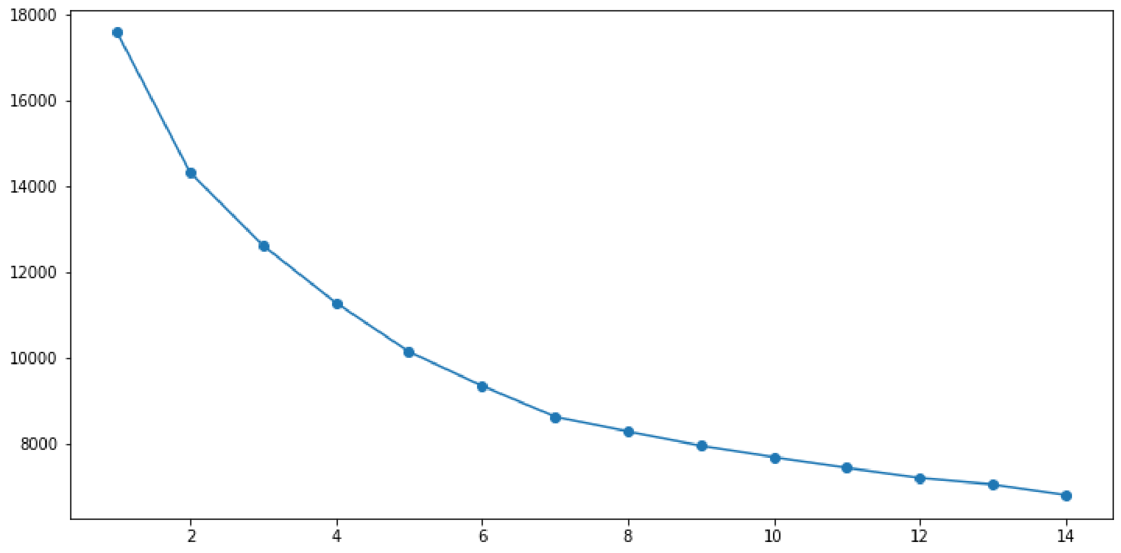
we are building model based on 3 clusters and also we are calculating the cluster errors

The total sum of squared distances of every data point from respective centroid is also called inertia. Let us print the inertia value for all K values. That K at which the inertia stop to drop significantly (elbow method) will be the best K.

[26]:
```
# Elbow plot

plt.figure(figsize=(12,6))
plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "o" )
```

[26]:
```
[<matplotlib.lines.Line2D at 0x16396b56e10>]
```



the elbow plot gives us the number of clusters to be used as we can see that the curve is slightly bent at 6 so we consider the number of clusters as 6 to build our model

[27]:
```
kmeans = KMeans(n_clusters=6, n_init = 15, random_state=2345)
```

[28]:
```
kmeans.fit(df1_scaled)
```

[28]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=6, n_init=15, n_jobs=None, precompute_distances='auto',
       random_state=2345, tol=0.0001, verbose=0)
```

[29]:
```
centroids = kmeans.cluster_centers_
centroids
```

[29]:
```
array([[-6.72451192e-02,  5.62073930e-02,  6.60225383e-02,
        -1.00502756e-02, -2.95157193e-02,  9.77087975e-01,
         1.20800508e+00,  2.48331007e-01, -1.26341795e-01,
        -1.70026513e-01, -5.62505055e-01],
       [ 1.33586541e+00, -6.74700057e-01,  1.12695753e+00,
         7.46994437e-02, -7.46339267e-03, -5.67340445e-01,
        -5.43792234e-01,  7.63188823e-01, -8.44712801e-01,
         3.49276696e-01,  1.77163635e-01],
       [-4.83625180e-01,  6.85964431e-01, -8.31615045e-01,
        -1.97792812e-01, -6.32528528e-02, -4.35623522e-01,
        -4.27241532e-01, -6.53210396e-02,  4.19133687e-01,
        -4.02054671e-01, -4.27167312e-01],
       [ 9.54162998e-02,  2.19980305e-03,  1.18155266e+00,
        -3.89872163e-01,  5.78475973e+00, -4.95156003e-02,
         5.10329601e-01,  1.80071833e-01, -1.73579154e+00,
         3.66341219e+00, -8.69731260e-01],
       [-6.92918829e-01, -4.37887177e-01, -1.47498832e-01,
        -2.57759586e-01, -4.17044161e-01,  1.24533302e-01,
        -2.29504460e-01, -1.24647482e+00,  6.33635100e-01,
         1.35008034e-01,  1.29557245e+00],
       [-8.56332101e-02, -3.46521643e-02,  4.14855742e-01,
         4.96176756e+00,  2.96387895e-01,  1.75019116e+00,
         1.69583213e+00,  1.22500051e+00, -3.25459600e-01,
        -2.37893278e-02, -3.63912996e-01]])
```

[30]:
```
centroid_df = pd.DataFrame(centroids, columns = list(df1_scaled) )
centroid_df
```

[30]:
```
   fixed_acidity  volatile_acidity  citric_acid  residual_sugar  chlorides  \
0      -0.067245          0.056207     0.066023       -0.010050  -0.029516
1       1.335865         -0.674700     1.126958        0.074699  -0.007463
2      -0.483625          0.685964    -0.831615       -0.197793  -0.063253
3       0.095416          0.002200     1.181553       -0.389872   5.784760
4      -0.692919         -0.437887    -0.147499       -0.257760  -0.417044
5      -0.085633         -0.034652     0.414856        4.961768   0.296388

   free_sulfur_dioxide  total_sulfur_dioxide   density        pH  sulphates  \
0             0.977088              1.208005  0.248331 -0.126342  -0.170027
1            -0.567340             -0.543792  0.763189 -0.844713   0.349277
2            -0.435624             -0.427242 -0.065321  0.419134  -0.402055
3            -0.049516              0.510330  0.180072 -1.735792   3.663412
4             0.124533             -0.229504 -1.246475  0.633635   0.135008
5             1.750191              1.695832  1.225001 -0.325460  -0.023789
```

```
    alcohol
0 -0.562505
1  0.177164
2 -0.427167
3 -0.869731
4  1.295572
5 -0.363913
```

[33]:
```
## creating a new dataframe only for labels and converting it into categorical␣
↪variable
df1_labels = pd.DataFrame(kmeans.labels_ , columns = list(['labels']))

df1_labels['labels'] = df1_labels['labels'].astype('category')
```

[34]:
```
# Joining the label dataframe with the Wine data frame to create␣
↪wine_df_labeled. Note: it could be appended to original dataframe
snail_df1_labeled = df1.join(df1_labels)
```

[36]:
```
df1_analysis = (snail_df1_labeled.groupby(['labels'] , axis=0)).head(4177)  #␣
↪the groupby creates a groupeddataframe that needs
# to be converted back to dataframe. I am using .head(30000) for that
df1_analysis
```

[36]:
```
      fixed_acidity  volatile_acidity  citric_acid  residual_sugar  chlorides  \
0               7.4             0.700         0.00             1.9      0.076
1               7.8             0.880         0.00             2.6      0.098
2               7.8             0.760         0.04             2.3      0.092
3              11.2             0.280         0.56             1.9      0.075
4               7.4             0.700         0.00             1.9      0.076
…               …                …           …               …         …
1594            6.2             0.600         0.08             2.0      0.090
1595            5.9             0.550         0.10             2.2      0.062
1596            6.3             0.510         0.13             2.3      0.076
1597            5.9             0.645         0.12             2.0      0.075
1598            6.0             0.310         0.47             3.6      0.067

      free_sulfur_dioxide  total_sulfur_dioxide  density    pH  sulphates  \
0                    11.0                  34.0  0.99780  3.51       0.56
1                    25.0                  67.0  0.99680  3.20       0.68
2                    15.0                  54.0  0.99700  3.26       0.65
3                    17.0                  60.0  0.99800  3.16       0.58
4                    11.0                  34.0  0.99780  3.51       0.56
…                    …                   …      …     …    …
1594                 32.0                  44.0  0.99490  3.45       0.58
1595                 39.0                  51.0  0.99512  3.52       0.76
1596                 29.0                  40.0  0.99574  3.42       0.75
1597                 32.0                  44.0  0.99547  3.57       0.71
```

```
1598                 18.0                    42.0  0.99549  3.39       0.66

      alcohol labels
0         9.4      2
1         9.8      2
2         9.8      2
3         9.8      1
4         9.4      2
…         …     …
1594     10.5      4
1595     11.2      4
1596     11.0      4
1597     10.2      2
1598     11.0      4

[1599 rows x 12 columns]
```

[37]:
```
snail_df1_labeled['labels'].value_counts()
```

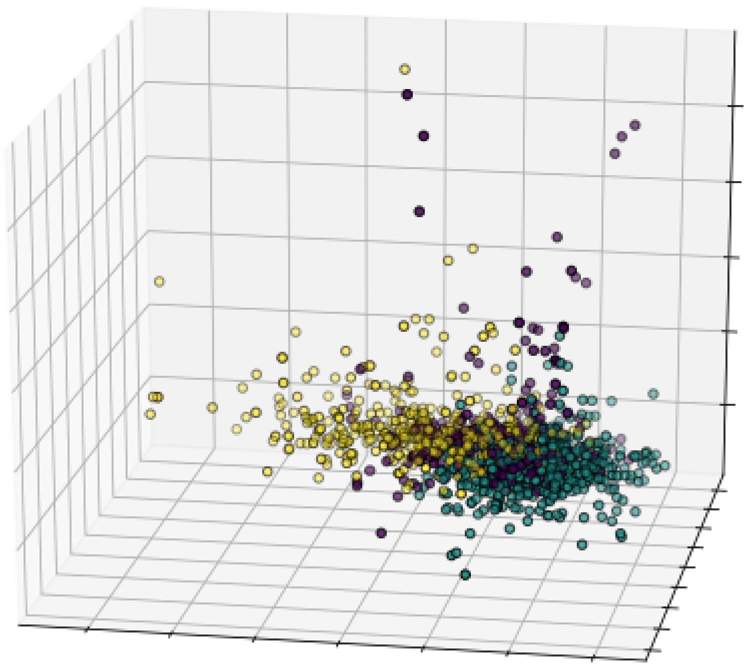
[37]:
```
2    524
1    364
0    347
4    302
5     34
3     28
Name: labels, dtype: int64
```

[38]:
```
from mpl_toolkits.mplot3d import Axes3D
```

[61]:
```
fig = plt.figure(figsize=(8, 6))
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=20, azim=100)
kmeans.fit(df1_scaled)
labels = kmeans.labels_
ax.scatter(df1_scaled.iloc[:, 0], df1_scaled.iloc[:, 1], df1_scaled.iloc[:,␣
 ↪3],c=labels.astype(np.float), edgecolor='k')
ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])
ax.set_title('3D plot of KMeans Clustering')
```

[61]:
```
Text(0.5, 0.92, '3D plot of KMeans Clustering')
```

3D plot of KMeans Clustering



Use feautes fixed acidity and volatile acidity

```
[44]: df2 = df1_scaled.loc[:, 'fixed_acidity': 'volatile_acidity']
      df2.head()
```

```
[44]:    fixed_acidity  volatile_acidity
      0      -0.528360          0.961877
      1      -0.298547          1.967442
      2      -0.298547          1.297065
      3       1.654856         -1.384443
      4      -0.528360          0.961877
```

```
[45]: model = KMeans(n_clusters = 3)
      model
```

```
[45]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
             n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
             random_state=None, tol=0.0001, verbose=0)
```

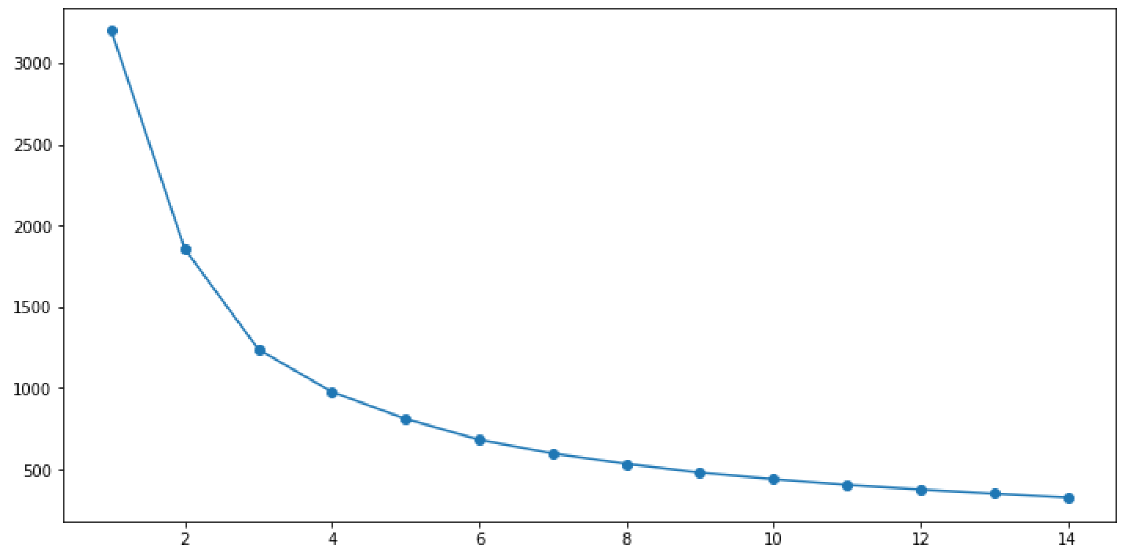
[49]:
```
cluster_range = range( 1, 15 )
cluster_errors = []
for num_clusters in cluster_range:
  clusters = KMeans( num_clusters, n_init = 10 )
  clusters.fit(df2)
  labels = clusters.labels_
  centroids = clusters.cluster_centers_
  cluster_errors.append( clusters.inertia_ )
clusters_df = pd.DataFrame( { "num_clusters":cluster_range, "cluster_errors":␣
 ↪cluster_errors } )
clusters_df[0:15]
```

[49]:
```
    num_clusters  cluster_errors
0              1     3198.000000
1              2     1855.089000
2              3     1237.886499
3              4      976.934633
4              5      810.980586
5              6      682.546130
6              7      598.547803
7              8      534.821305
8              9      480.822240
9             10      440.501462
10            11      404.879042
11            12      376.540714
12            13      350.924509
13            14      326.937199
```

[50]:
```
# Elbow plot

plt.figure(figsize=(12,6))
plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "o" )
```

[50]:
```
[<matplotlib.lines.Line2D at 0x1639cc142e8>]
```



the value of k is 3 in this case

```
[51]: kmeans = KMeans(n_clusters=3, n_init = 15, random_state=2345)
```

```
[52]: kmeans.fit(df2)
```

```
[52]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
             n_clusters=3, n_init=15, n_jobs=None, precompute_distances='auto',
             random_state=2345, tol=0.0001, verbose=0)
```

```
[53]: centroids = kmeans.cluster_centers_
      centroids
```

```
[53]: array([[-0.43343719,  0.93958281],
              [ 1.47529531, -0.60241457],
              [-0.42029776, -0.6523901 ]])
```

```
[55]: centroid_df = pd.DataFrame(centroids, columns = list(df2) )
      centroid_df
```

```
[55]:    fixed_acidity  volatile_acidity
      0      -0.433437          0.939583
      1       1.475295         -0.602415
      2      -0.420298         -0.652390
```

```
[56]: ## creating a new dataframe only for labels and converting it into categorical␣
       ↪variable
      df_labels = pd.DataFrame(kmeans.labels_ , columns = list(['labels']))
```

```
df_labels['labels'] = df_labels['labels'].astype('category')
```

[57]:
```
snail_df_labeled = df2.join(df_labels)
```

[58]:
```
df_analysis = (snail_df_labeled.groupby(['labels'] , axis=0)).head(4177)  # the␣
 ↪groupby creates a groupeddataframe that needs
# to be converted back to dataframe. I am using .head(30000) for that
df_analysis
```

[58]:
```
      fixed_acidity  volatile_acidity labels
0         -0.528360          0.961877      0
1         -0.298547          1.967442      0
2         -0.298547          1.297065      0
3          1.654856         -1.384443      1
4         -0.528360          0.961877      0
…              …                 …     …
1594      -1.217796          0.403229      0
1595      -1.390155          0.123905      2
1596      -1.160343         -0.099554      2
1597      -1.390155          0.654620      0
1598      -1.332702         -1.216849      2

[1599 rows x 3 columns]
```

[59]:
```
snail_df_labeled['labels'].value_counts()
```

[59]:
```
0    644
2    596
1    359
Name: labels, dtype: int64
```

Use feautes Cirtic acidity and fixed acidity

[72]:
```
df3 = df1_scaled[[ 'citric_acid','fixed_acidity']]
df3.head()
```

[72]:
```
   citric_acid  fixed_acidity
0    -1.391472      -0.528360
1    -1.391472      -0.298547
2    -1.186070      -0.298547
3     1.484154       1.654856
4    -1.391472      -0.528360
```

[73]:
```
model = KMeans(n_clusters = 3)
model
```

```
[73]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
             n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
             random_state=None, tol=0.0001, verbose=0)
```

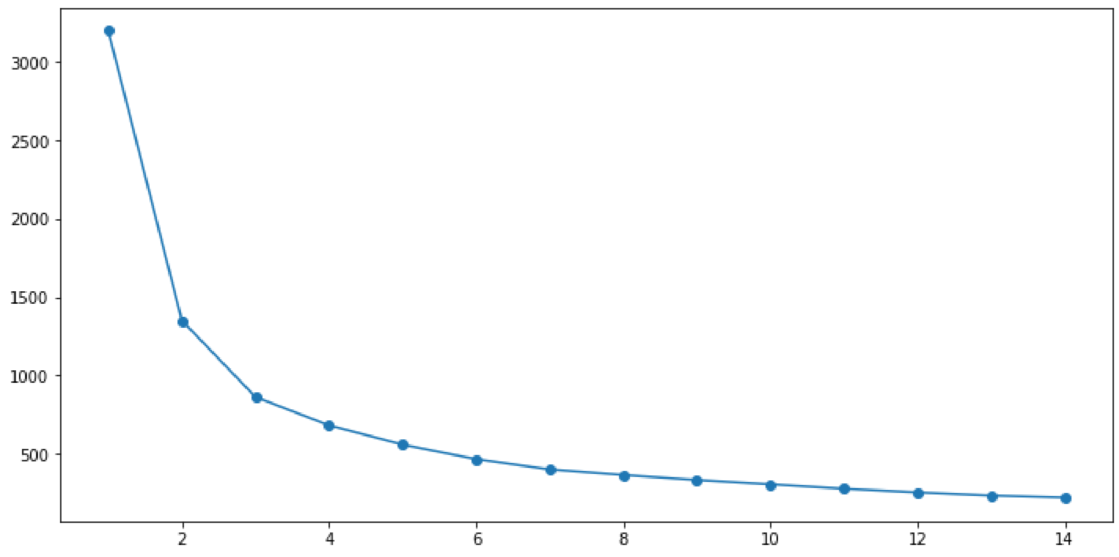
```
[74]: cluster_range = range( 1, 15 )
      cluster_errors = []
      for num_clusters in cluster_range:
        clusters = KMeans( num_clusters, n_init = 10 )
        clusters.fit(df3)
       # labels = clusters.labels_
       # centroids = clusters.cluster_centers_
        cluster_errors.append( clusters.inertia_ )
      clusters_df = pd.DataFrame( { "num_clusters":cluster_range, "cluster_errors":␣
       ↪cluster_errors } )
      clusters_df[0:15]
```

```
[74]:     num_clusters  cluster_errors
      0              1     3198.000000
      1              2     1349.582779
      2              3      861.704601
      3              4      680.668631
      4              5      558.073517
      5              6      465.071896
      6              7      399.431950
      7              8      364.434654
      8              9      331.892906
      9             10      304.718267
      10            11      276.531927
      11            12      252.743949
      12            13      232.644904
      13            14      220.003736
```

```
[75]: # Elbow plot

      plt.figure(figsize=(12,6))
      plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "o" )
```

```
[75]: [<matplotlib.lines.Line2D at 0x1639ce3e438>]
```



1 here we are taking the value of k=3

```
[76]: kmeans = KMeans(n_clusters=3, n_init = 15, random_state=2345)
```

```
[77]: kmeans.fit(df3)
```

```
[77]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
             n_clusters=3, n_init=15, n_jobs=None, precompute_distances='auto',
             random_state=2345, tol=0.0001, verbose=0)
```

```
[79]: centroids = kmeans.cluster_centers_
      centroids
```

```
[79]: array([[-0.95846298, -0.70641229],
             [ 0.38702498, -0.0232819 ],
             [ 1.30809487,  1.61190945]])
```

```
[81]: centroid_df = pd.DataFrame(centroids, columns = list(df3) )
      centroid_df
```

```
[81]:    citric_acid  fixed_acidity
      0    -0.958463      -0.706412
      1     0.387025      -0.023282
      2     1.308095       1.611909
```

```
[82]: ## creating a new dataframe only for labels and converting it into categorical␣
       ↪variable
```

```
df_labels = pd.DataFrame(kmeans.labels_ , columns = list(['labels']))

df_labels['labels'] = df_labels['labels'].astype('category')
```

[83]:
```
snail_df_labeled = df3.join(df_labels)
```

[84]:
```
df_analysis = (snail_df_labeled.groupby(['labels'] , axis=0)).head(4177)  # the␣
↪groupby creates a groupeddataframe that needs
# to be converted back to dataframe. I am using .head(30000) for that
df_analysis
```

[84]:
```
      citric_acid  fixed_acidity labels
0       -1.391472      -0.528360      0
1       -1.391472      -0.298547      0
2       -1.186070      -0.298547      0
3        1.484154       1.654856      2
4       -1.391472      -0.528360      0
…             …              …     …
1594    -0.980669      -1.217796      0
1595    -0.877968      -1.390155      0
1596    -0.723916      -1.160343      0
1597    -0.775267      -1.390155      0
1598     1.021999      -1.332702      1

[1599 rows x 3 columns]
```

[85]:
```
snail_df_labeled['labels'].value_counts()
```

[85]:
```
0    666
1    632
2    301
Name: labels, dtype: int64
```

[ ]:
```
#3 Use feautes residual suger and sulphades
```

[88]:
```
df.columns
```

[88]:
```
Index(['fixed_acidity', 'volatile_acidity', 'citric_acid', 'residual_sugar',
       'chlorides', 'free_sulfur_dioxide', 'total_sulfur_dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

[91]:
```
df4 = df1_scaled[[ 'residual_sugar','sulphates']]
df4.head()
```

[91]:
```
   residual_sugar  sulphates
0       -0.453218  -0.579207
```

```
1        0.043416   0.128950
2       -0.169427  -0.048089
3       -0.453218  -0.461180
4       -0.453218  -0.579207
```

[92]:
```
model = KMeans(n_clusters = 3)
model
```

[92]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

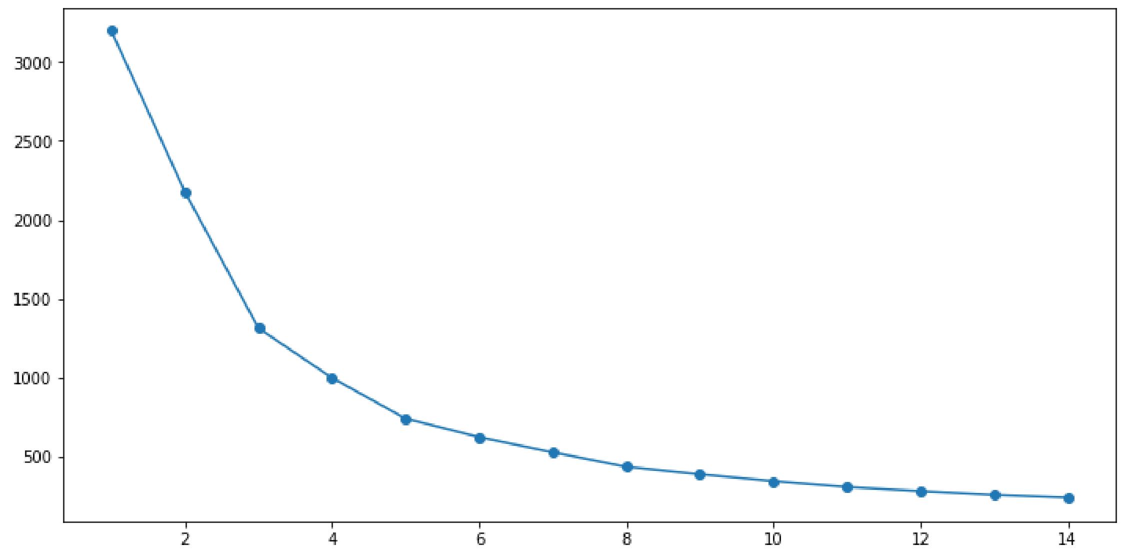
[93]:
```
cluster_range = range( 1, 15 )
cluster_errors = []
for num_clusters in cluster_range:
  clusters = KMeans( num_clusters, n_init = 10 )
  clusters.fit(df4)
 # labels = clusters.labels_
 # centroids = clusters.cluster_centers_
  cluster_errors.append( clusters.inertia_ )
clusters_df = pd.DataFrame( { "num_clusters":cluster_range, "cluster_errors":␣
 ↪cluster_errors } )
clusters_df[0:15]
```

[93]:
```
    num_clusters  cluster_errors
0              1     3198.000000
1              2     2177.114997
2              3     1312.577756
3              4      998.173046
4              5      739.725164
5              6      622.227993
6              7      526.452113
7              8      434.267238
8              9      387.224194
9             10      343.239636
10            11      306.545590
11            12      279.599023
12            13      256.186147
13            14      238.970926
```

[94]:
```
# Elbow plot

plt.figure(figsize=(12,6))
plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "o" )
```

[94]:
```
[<matplotlib.lines.Line2D at 0x1639ceab4e0>]
```



here we are taking k value as 5

[97]:
```
kmeans = KMeans(n_clusters=3, n_init = 15, random_state=2345)
```

[99]:
```
kmeans.fit(df4)
```

[99]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=15, n_jobs=None, precompute_distances='auto',
       random_state=2345, tol=0.0001, verbose=0)
```

[100]:
```
centroids = kmeans.cluster_centers_
centroids
```

[100]:
```
array([[-0.18614032, -0.40542353],
       [-0.19888324,  1.41705131],
       [ 3.36585956,  0.01717269]])
```

[102]:
```
centroid_df = pd.DataFrame(centroids, columns = list(df4) )
centroid_df
```

[102]:
```
   residual_sugar  sulphates
0       -0.186140  -0.405424
1       -0.198883   1.417051
2        3.365860   0.017173
```

[106]:
```
## creating a new dataframe only for labels and converting it into categorical␣
 ↪variable
df_labels = pd.DataFrame(kmeans.labels_ , columns = list(['labels']))
```

```
df_labels['labels'] = df_labels['labels'].astype('category')
```

[107]:
```
# Joining the label dataframe with the Wine data frame to create␣
↪wine_df_labeled. Note: it could be appended to original dataframe
snail_df_labeled = df2.join(df_labels)
```

[108]:
```
df_analysis = (snail_df_labeled.groupby(['labels'] , axis=0)).head(4177)  # the␣
↪groupby creates a groupeddataframe that needs
# to be converted back to dataframe. I am using .head(30000) for that
df_analysis
```

[108]:
```
     labels
0         0
1         0
2         0
3         0
4         0
…       …
1594      0
1595      1
1596      1
1597      0
1598      0

[1599 rows x 1 columns]
```

[109]:
```
snail_df_labeled['labels'].value_counts()
```

[109]:
```
0    1178
1     336
2      85
Name: labels, dtype: int64
```

#4 Use feautes free.sulfur.dioxide and total.sulfur.dioxide

[88]:
```
df.columns
```

[88]:
```
Index(['fixed_acidity', 'volatile_acidity', 'citric_acid', 'residual_sugar',
       'chlorides', 'free_sulfur_dioxide', 'total_sulfur_dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

[112]:
```
df5 = df1_scaled[[ 'free_sulfur_dioxide','total_sulfur_dioxide']]
df5.head()
```

[112]:
```
   free_sulfur_dioxide  total_sulfur_dioxide
0            -0.466193             -0.379133
```

```
1              0.872638              0.624363
2             -0.083669              0.229047
3              0.107592              0.411500
4             -0.466193             -0.379133
```

[113]:
```
model = KMeans(n_clusters = 3)
model
```

[113]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

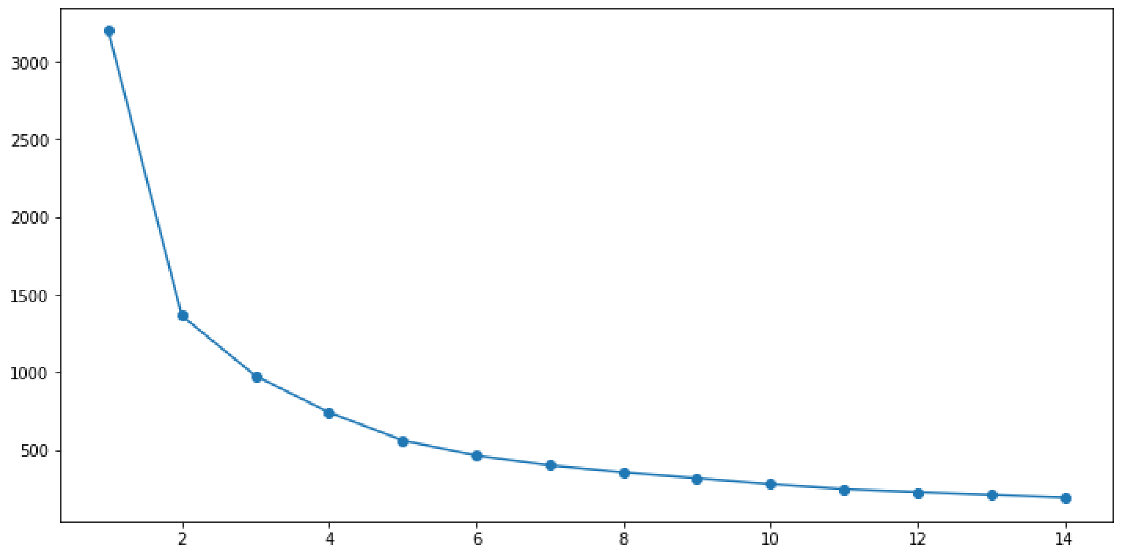
[114]:
```
cluster_range = range( 1, 15 )
cluster_errors = []
for num_clusters in cluster_range:
  clusters = KMeans( num_clusters, n_init = 10 )
  clusters.fit(df5)
 # labels = clusters.labels_
 # centroids = clusters.cluster_centers_
  cluster_errors.append( clusters.inertia_ )
clusters_df = pd.DataFrame( { "num_clusters":cluster_range, "cluster_errors":␣
 ↪cluster_errors } )
clusters_df[0:15]
```

[114]:
```
    num_clusters  cluster_errors
0              1     3198.000000
1              2     1363.149246
2              3      975.837335
3              4      740.680585
4              5      560.417140
5              6      462.895676
6              7      400.038118
7              8      354.388299
8              9      317.066485
9             10      278.740024
10            11      246.775241
11            12      227.159562
12            13      209.648403
13            14      192.081726
```

[115]:
```
# Elbow plot

plt.figure(figsize=(12,6))
plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "o" )
```

[115]:
```
[<matplotlib.lines.Line2D at 0x1639ced5e10>]
```



here we are taking k value as 2

```
[116]: kmeans = KMeans(n_clusters=3, n_init = 15, random_state=2345)
```

```
[117]: kmeans.fit(df4)
```

```
[117]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
               n_clusters=3, n_init=15, n_jobs=None, precompute_distances='auto',
               random_state=2345, tol=0.0001, verbose=0)
```

```
[118]: centroids = kmeans.cluster_centers_
       centroids
```

```
[118]: array([[-0.18614032, -0.40542353],
              [-0.19888324,  1.41705131],
              [ 3.36585956,  0.01717269]])
```

```
[119]: centroid_df = pd.DataFrame(centroids, columns = list(df4) )
       centroid_df
```

```
[119]:    residual_sugar  sulphates
       0       -0.186140  -0.405424
       1       -0.198883   1.417051
       2        3.365860   0.017173
```

```
[120]: ## creating a new dataframe only for labels and converting it into categorical␣
        ↪variable
       df_labels = pd.DataFrame(kmeans.labels_ , columns = list(['labels']))
```

```
df_labels['labels'] = df_labels['labels'].astype('category')
```

```
[121]: # Joining the label dataframe with the Wine data frame to create␣
        ↪wine_df_labeled. Note: it could be appended to original dataframe
       snail_df_labeled = df2.join(df_labels)
```

```
[122]: df_analysis = (snail_df_labeled.groupby(['labels'] , axis=0)).head(4177)  # the␣
        ↪groupby creates a groupeddataframe that needs
       # to be converted back to dataframe. I am using .head(30000) for that
       df_analysis
```

```
[122]:      labels
       0         0
       1         0
       2         0
       3         0
       4         0
       …      …
       1594      0
       1595      1
       1596      1
       1597      0
       1598      0

       [1599 rows x 1 columns]
```

```
[123]: snail_df_labeled['labels'].value_counts()
```

```
[123]: 0    1178
       1     336
       2      85
       Name: labels, dtype: int64
```

Use feautes fixed acidity, citric acidity and volatile acidity

```
[ ]:
```

```
[88]: df.columns
```

```
[88]: Index(['fixed_acidity', 'volatile_acidity', 'citric_acid', 'residual_sugar',
             'chlorides', 'free_sulfur_dioxide', 'total_sulfur_dioxide', 'density',
             'pH', 'sulphates', 'alcohol', 'quality'],
            dtype='object')
```

```
[127]: df6 = df1_scaled[[ 'fixed_acidity', 'volatile_acidity', 'citric_acid']]
       df6.head()
```

[127]:
```
   fixed_acidity  volatile_acidity  citric_acid
0      -0.528360          0.961877    -1.391472
1      -0.298547          1.967442    -1.391472
2      -0.298547          1.297065    -1.186070
3       1.654856         -1.384443     1.484154
4      -0.528360          0.961877    -1.391472
```

[128]:
```
model = KMeans(n_clusters = 3)
model
```

[128]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

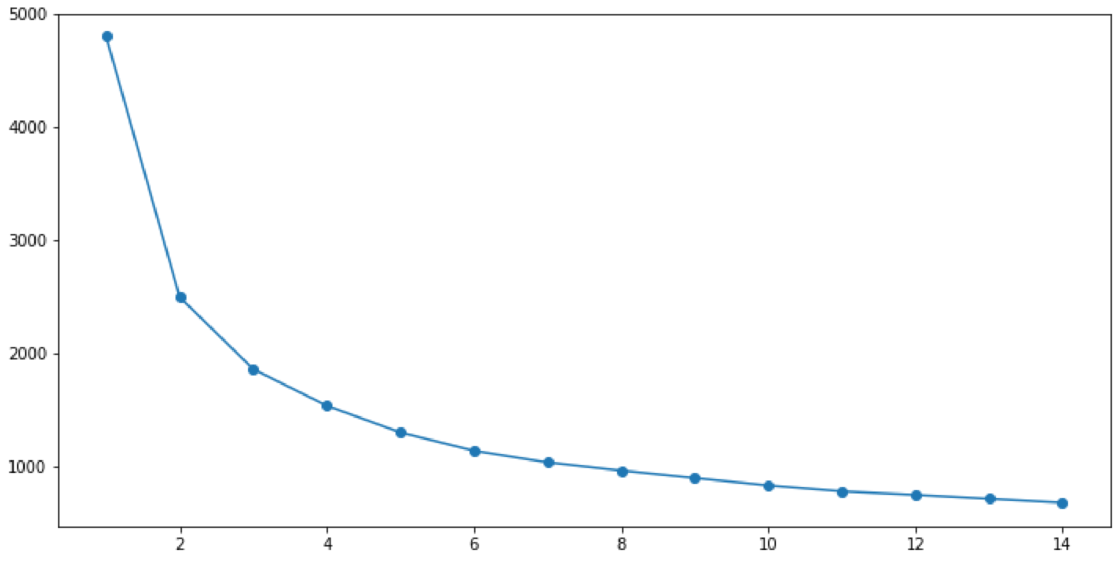
[129]:
```
cluster_range = range( 1, 15 )
cluster_errors = []
for num_clusters in cluster_range:
  clusters = KMeans( num_clusters, n_init = 10 )
  clusters.fit(df6)
 # labels = clusters.labels_
 # centroids = clusters.cluster_centers_
  cluster_errors.append( clusters.inertia_ )
clusters_df = pd.DataFrame( { "num_clusters":cluster_range, "cluster_errors":␣
 ↪cluster_errors } )
clusters_df[0:15]
```

[129]:
```
    num_clusters  cluster_errors
0              1     4797.000000
1              2     2494.155753
2              3     1857.127760
3              4     1534.077522
4              5     1297.582734
5              6     1135.794701
6              7     1033.152006
7              8      961.045212
8              9      895.923730
9             10      829.524099
10            11      777.970401
11            12      746.043602
12            13      711.805290
13            14      677.369459
```

[130]:
```
# Elbow plot

plt.figure(figsize=(12,6))
plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "o" )
```

```
[130]: [<matplotlib.lines.Line2D at 0x1639cf46390>]
```



here we are taking k value as 2

```
[131]: kmeans = KMeans(n_clusters=3, n_init = 15, random_state=2345)
```

```
[133]: kmeans.fit(df6)
```

```
[133]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
               n_clusters=3, n_init=15, n_jobs=None, precompute_distances='auto',
               random_state=2345, tol=0.0001, verbose=0)
```

```
[134]: centroids = kmeans.cluster_centers_
       centroids
```

```
[134]: array([[-0.51869461,  0.76068147, -0.83114207],
              [ 1.5514048 , -0.60938513,  1.27619223],
              [-0.20699183, -0.70725481,  0.40239642]])
```

```
[136]: centroid_df = pd.DataFrame(centroids, columns = list(df6) )
       centroid_df
```

```
[136]:    fixed_acidity  volatile_acidity  citric_acid
       0      -0.518695          0.760681    -0.831142
       1       1.551405         -0.609385     1.276192
       2      -0.206992         -0.707255     0.402396
```

```
[137]: ## creating a new dataframe only for labels and converting it into categorical␣
       ↪variable
       df_labels = pd.DataFrame(kmeans.labels_ , columns = list(['labels']))

       df_labels['labels'] = df_labels['labels'].astype('category')
```

```
[138]: # Joining the label dataframe with the Wine data frame to create␣
       ↪wine_df_labeled. Note: it could be appended to original dataframe
       snail_df_labeled = df2.join(df_labels)
```

```
[139]: df_analysis = (snail_df_labeled.groupby(['labels'] , axis=0)).head(4177)  # the␣
       ↪groupby creates a groupeddataframe that needs
       # to be converted back to dataframe. I am using .head(30000) for that
       df_analysis
```

```
[139]:      labels
       0         0
       1         0
       2         0
       3         1
       4         0
       …       …
       1594      0
       1595      0
       1596      0
       1597      0
       1598      2

       [1599 rows x 1 columns]
```

```
[140]: snail_df_labeled['labels'].value_counts()
```

```
[140]: 0    749
       2    528
       1    322
       Name: labels, dtype: int64
```

```
[ ]:
```

```
[ ]:
```

#4 Use feautes density, ph

```
[88]: df.columns
```

```
[88]: Index(['fixed_acidity', 'volatile_acidity', 'citric_acid', 'residual_sugar',
             'chlorides', 'free_sulfur_dioxide', 'total_sulfur_dioxide', 'density',
             'pH', 'sulphates', 'alcohol', 'quality'],
            dtype='object')
```

[141]:
```
df7 = df1_scaled[[ 'density', 'pH']]
df7.head()
```

[141]:
```
    density        pH
0  0.558274  1.288643
1  0.028261 -0.719933
2  0.134264 -0.331177
3  0.664277 -0.979104
4  0.558274  1.288643
```

[142]:
```
model = KMeans(n_clusters = 3)
model
```

[142]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

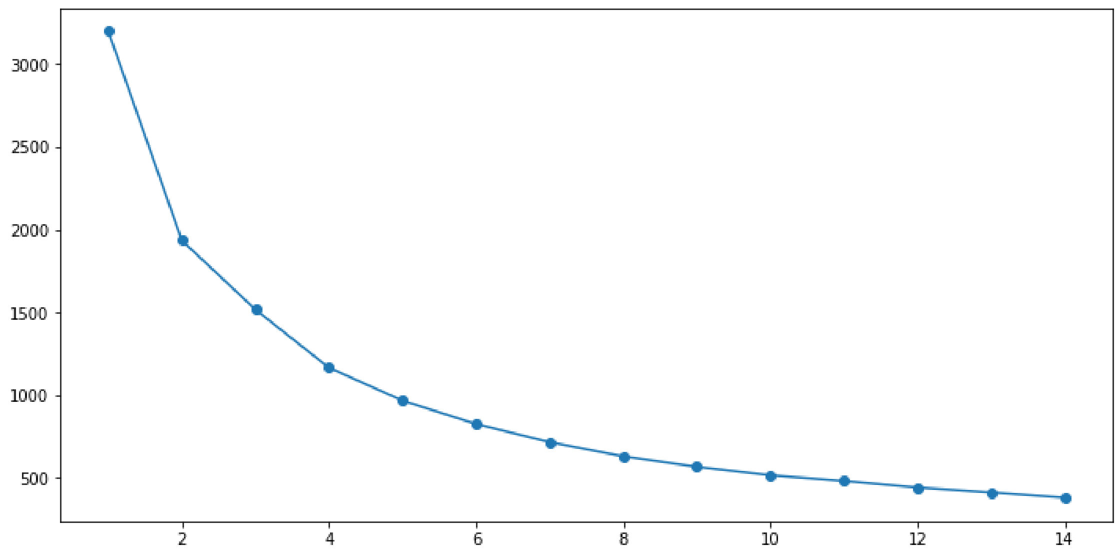
[143]:
```
cluster_range = range( 1, 15 )
cluster_errors = []
for num_clusters in cluster_range:
  clusters = KMeans( num_clusters, n_init = 10 )
  clusters.fit(df7)
 # labels = clusters.labels_
 # centroids = clusters.cluster_centers_
  cluster_errors.append( clusters.inertia_ )
clusters_df = pd.DataFrame( { "num_clusters":cluster_range, "cluster_errors":␣
 ↪cluster_errors } )
clusters_df[0:15]
```

[143]:
```
    num_clusters  cluster_errors
0              1     3198.000000
1              2     1934.185304
2              3     1516.801912
3              4     1165.674841
4              5      966.910641
5              6      825.483154
6              7      716.252814
7              8      630.578389
8              9      567.017297
9             10      516.221073
10            11      481.714845
11            12      442.195541
12            13      412.372292
13            14      381.005891
```

[144]:
```
# Elbow plot

plt.figure(figsize=(12,6))
plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "o" )
```

[144]:
```
[<matplotlib.lines.Line2D at 0x1639d015da0>]
```



here we are taking k value as 4

[145]:
```
kmeans = KMeans(n_clusters=3, n_init = 15, random_state=2345)
```

[147]:
```
kmeans.fit(df7)
```

[147]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=15, n_jobs=None, precompute_distances='auto',
       random_state=2345, tol=0.0001, verbose=0)
```

[148]:
```
centroids = kmeans.cluster_centers_
centroids
```

[148]:
```
array([[-0.56000264,  0.91708692],
       [-0.26415144, -0.57202702],
       [ 1.22728261, -0.57114987]])
```

[150]:
```
centroid_df = pd.DataFrame(centroids, columns = list(df7) )
centroid_df
```

[150]:
```
    density        pH
0 -0.560003  0.917087
```

```
1 -0.264151 -0.572027
2  1.227283 -0.571150
```

[151]:
```
## creating a new dataframe only for labels and converting it into categorical␣
↪variable
df_labels = pd.DataFrame(kmeans.labels_ , columns = list(['labels']))

df_labels['labels'] = df_labels['labels'].astype('category')
```

[152]:
```
# Joining the label dataframe with the Wine data frame to create␣
↪wine_df_labeled. Note: it could be appended to original dataframe
snail_df_labeled = df2.join(df_labels)
```

[153]:
```
df_analysis = (snail_df_labeled.groupby(['labels'] , axis=0)).head(4177)  # the␣
↪groupby creates a groupeddataframe that needs
# to be converted back to dataframe. I am using .head(30000) for that
df_analysis
```

[153]:
```
     labels
0         0
1         1
2         1
3         2
4         0
…       …
1594      0
1595      0
1596      0
1597      0
1598      0

[1599 rows x 1 columns]
```

[154]:
```
snail_df_labeled['labels'].value_counts()
```

[154]:
```
0    613
1    581
2    405
Name: labels, dtype: int64
```

[ ]:

[ ]:

[ ]: