iris-optimum_number_of_clusters

June 7, 2022

1 Iris dataset classification using KMeans Clustering machine learning algorithm

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

1.0.1 importing dataset

1.0.2 dataset --> DataFrame

[2]:
```
df = pd.read_csv("E:/Datasets/iris.csv")
```

[3]:
```
df.head()
```

[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

[4]:
```
df.isnull().sum()
```

[4]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

1.0.3 statistical description

[6]:
```
#droping Id column
df.drop('Id', axis=1, inplace=True)
df.describe().T
```

[6]:

	count	mean	std	min	25%	50%	75%	max
SepalLengthCm	150.0	5.843333	0.828066	4.3	5.1	5.80	6.4	7.9
SepalWidthCm	150.0	3.054000	0.433594	2.0	2.8	3.00	3.3	4.4
PetalLengthCm	150.0	3.758667	1.764420	1.0	1.6	4.35	5.1	6.9
PetalWidthCm	150.0	1.198667	0.763161	0.1	0.3	1.30	1.8	2.5

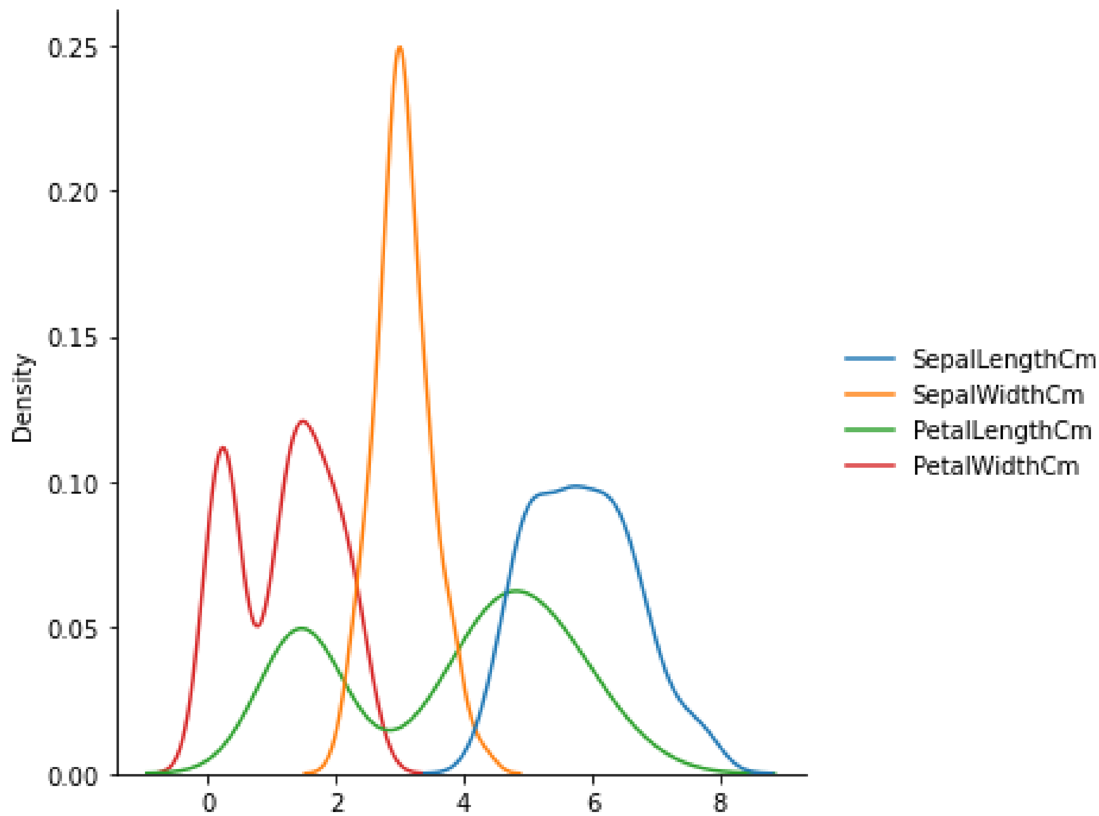
[7]:
```
#count species of iris flower
df.groupby('Species').count()
```

[7]:

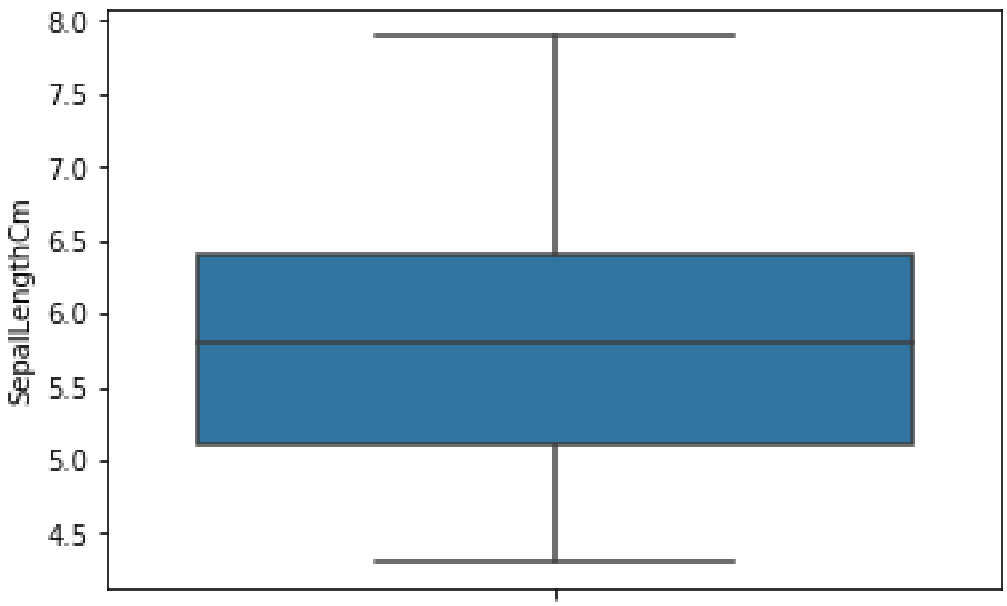
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Species				
Iris-setosa	50	50	50	50
Iris-versicolor	50	50	50	50
Iris-virginica	50	50	50	50

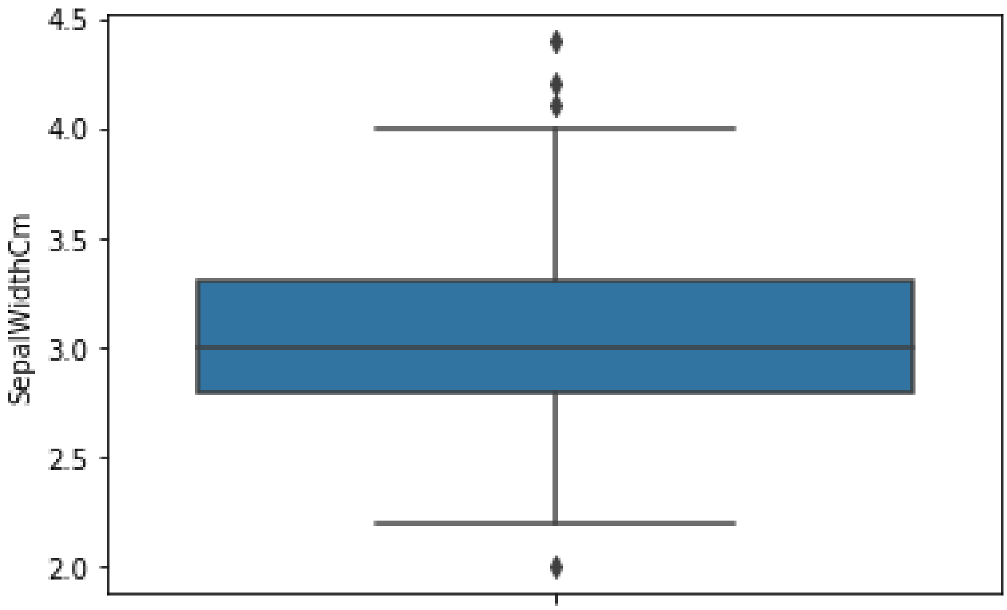
[8]:
```
# visualising data distribution
df_col = list(df.columns.drop('Species'))
sns.displot(df, kind = 'kde')
```

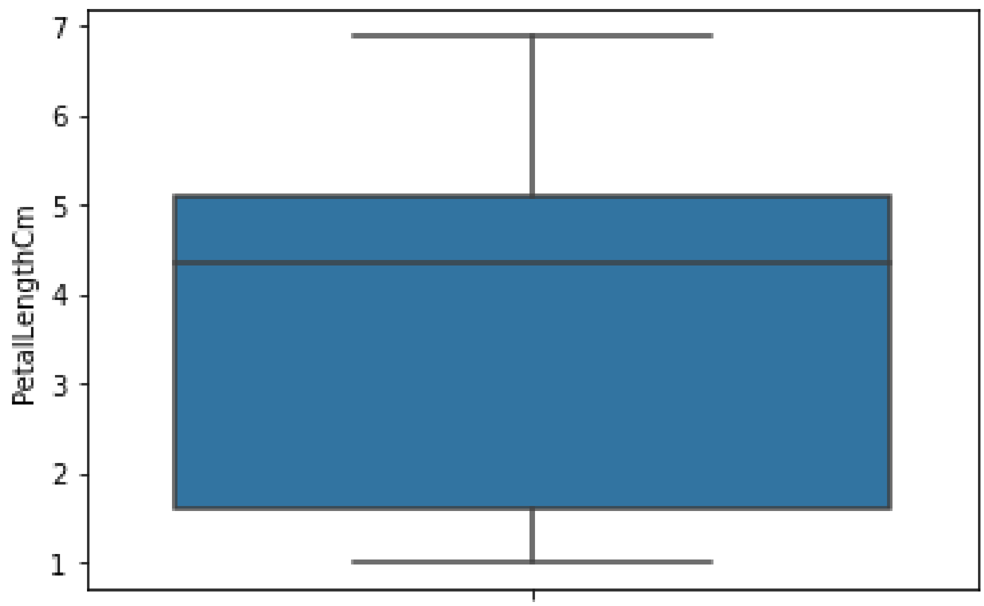
[8]: <seaborn.axisgrid.FacetGrid at 0x15e8cde66a0>

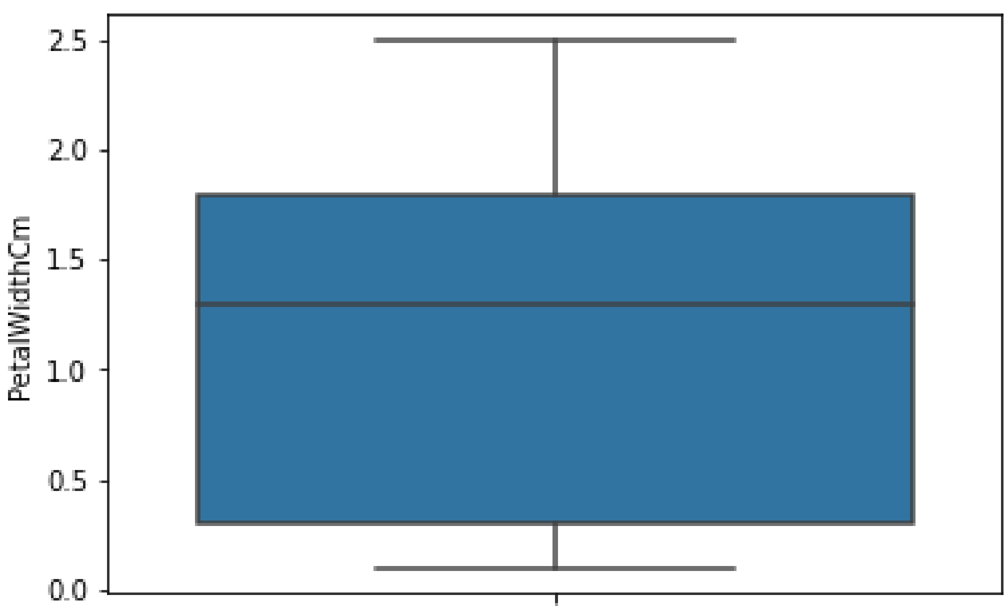


[9]:
```
#finding outliers
col = list(df.columns)
col=col[0:-1]
for i in col:
    sns.boxplot(y=df[i])
    plt.show()
```


8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
SepalLengthCm


4.5
4.0
3.5
3.0
2.5
2.0
SepalWidthCm





[10]:
```
q1=df['SepalWidthCm'].quantile(0.25)
q3=df['SepalWidthCm'].quantile(0.75)

iqr = q3-q1
df=df[(df['SepalWidthCm']>=q1-1.5*iqr) & (df['SepalWidthCm']<=q3+1.5*iqr)]
df.head()
```

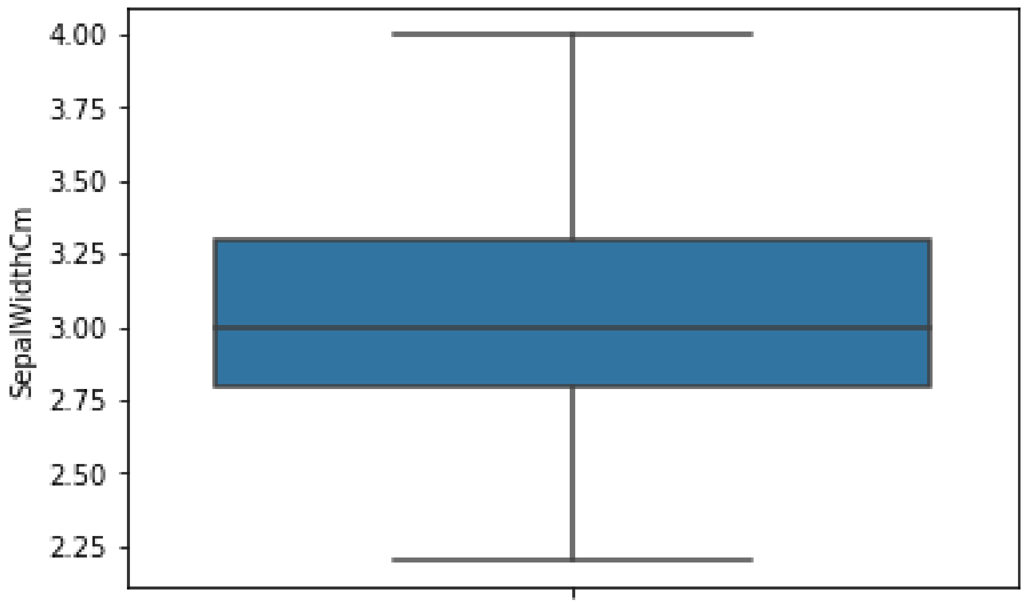
[10]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

[11]:
```
#after removing outliers from dataframe
df.shape
```

[11]: (146, 5)

[12]:
```
sns.boxplot(y=df['SepalWidthCm'])
plt.show()
```



1.0.4 training data

[13]:
```
X = df.iloc[:,0:4]
```

1.0.5 Normalising data

[14]:
```
scale = StandardScaler()
norm_df=scale.fit_transform(X)
```

1.0.6 train algorithm over k cluster_range

[15]:
```
cluster_range = range(1,20)

#( Within-Cluster Sum of Square ).
WCSS = []

for n_cluster in cluster_range:
    clusters = KMeans(n_cluster, n_init=10)
    clusters.fit(X)
    labels = clusters.labels_
    center = clusters.cluster_centers_
    WCSS.append(clusters.inertia_)
```

```
C:\DEEPAK\ANACONDA\lib\site-packages\sklearn\cluster\_kmeans.py:881:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

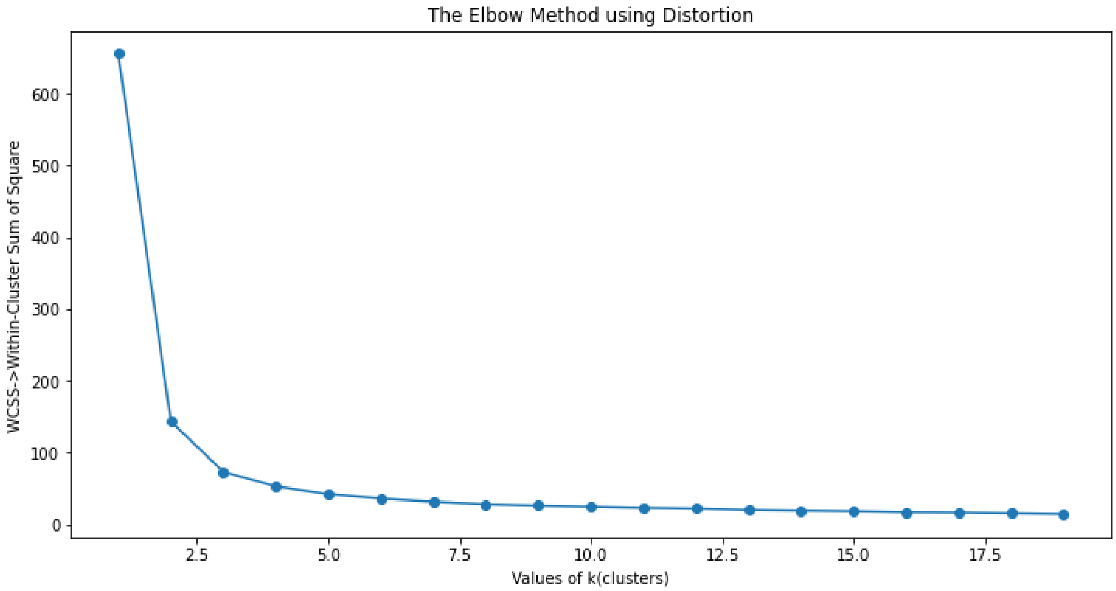
[16]:
```
df_cluster = pd.DataFrame({'Clusters':cluster_range,'WCSS':WCSS})
df_cluster
```

[16]:

	Clusters	WCSS
0	1	655.032534
1	2	143.860075
2	3	73.516565
3	4	53.304878
4	5	42.548635
5	6	36.508188
6	7	31.863827
7	8	28.487175
8	9	26.488831
9	10	25.065460
10	11	23.444732
11	12	22.366448
12	13	20.609640
13	14	19.554853
14	15	18.594544
15	16	17.193377
16	17	16.764502
17	18	15.776722
18	19	14.656394

1.0.7 Elbow plot visual

[17]:
```
plt.figure(figsize=(12,6))
plt.plot(cluster_range, WCSS, marker='o')
plt.title('The Elbow Method using Distortion')
plt.xlabel("Values of k(clusters)")
plt.ylabel("WCSS->Within-Cluster Sum of Square")
plt.show()
```



1.0.8 train model over 3 clusters

[18]:
```
model = KMeans(n_clusters=3, max_iter=300)
model.fit(X)
```

[18]: KMeans(n_clusters=3)

1.0.9 analising clusters

[24]:
```
# analising clusters
df.index = pd.RangeIndex(len(df.index))
df_km = pd.concat([X, pd.Series(model.labels_)], axis=1)
df_km.columns = ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',␣
↪'petal width (cm)', 'ClusterID']
km_clusters_Slength = pd.DataFrame(df_km.groupby(['ClusterID']).agg({'sepal␣
↪length (cm)':'mean'}))
```

```
km_clusters_Swidth = pd.DataFrame(df_km.groupby(['ClusterID']).agg({'sepal␣
↪width (cm)':'mean'}))
km_clusters_Plength = pd.DataFrame(df_km.groupby(['ClusterID']).agg({'petal␣
↪length (cm)':'mean'}))
km_clusters_Pwidth = pd.DataFrame(df_km.groupby(['ClusterID']).agg({'petal␣
↪width (cm)':'mean'}))
```

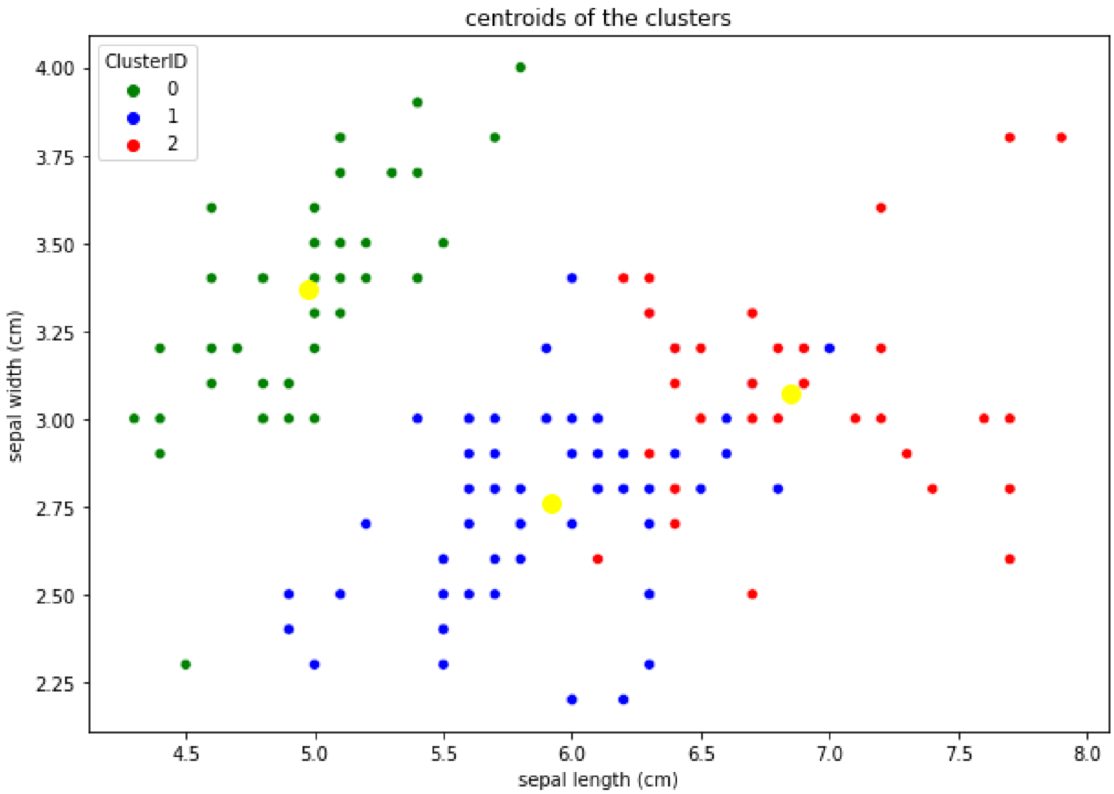
[25]:
```
df2 = pd.concat([pd.Series([0,1,2]), km_clusters_Slength, km_clusters_Swidth,␣
↪km_clusters_Plength, km_clusters_Pwidth
                ], axis=1)
df2.columns = ['ClusterID','sepal length (cm)_mean','sepal width␣
↪(cm)_mean','petal length (cm)_mean',
               'petal width (cm)_mean']
df2.head()
```

[25]:
```
   ClusterID  sepal length (cm)_mean  sepal width (cm)_mean  \
0          0                4.976596               3.365957
1          1                5.916393               2.760656
2          2                6.850000               3.073684

   petal length (cm)_mean  petal width (cm)_mean
0                1.463830               0.244681
1                4.408197               1.440984
2                5.742105               2.071053
```

[27]:
```
#Scatter plot to visualize the clusters
plt.figure(figsize=(10,7))
sns.scatterplot(x='sepal length (cm)',y='sepal width (cm)', data=df_km,␣
↪hue='ClusterID', palette=['green','blue','red'])

#centroids of the clusters
plt.scatter(model.cluster_centers_[:, 0], model.cluster_centers_[:,1],
            s = 100, c = 'yellow', label = 'Centroids')
plt.title('centroids of the clusters')
plt.show()
```



centroids of the clusters
ClusterID
0
1
2
4.00
3.75
3.50
3.25
3.00
2.75
2.50
2.25
sepal width (cm)
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
sepal length (cm)

[ ]:

[ ]: