Grip @ The Sparks Foundation , JULY2021

Field : Data Science and Business Analytics

Name:SALINI S

Task 2 Description : From the given 'Iris' dataset, predict the optimum number of clusters and represent it visually.

In [6]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

In [8]:
```
df = pd.read_csv('C:\Users\HP\Downloads\Iris.csv')
df.head(10)
```

Out[8]:

	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
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

In [9]:
```
df.shape
```

Out[9]:
```
(150, 6)
```

In [10]:
```
df.info
```

Out[10]:
```
<bound method DataFrame.info of       Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWi
dthCm  \
0      1            5.1           3.5            1.4            0.2
1      2            4.9           3.0            1.4            0.2
2      3            4.7           3.2            1.3            0.2
3      4            4.6           3.1            1.5            0.2
4      5            5.0           3.6            1.4            0.2
..   ...            ...           ...            ...            ...
145  146            6.7           3.0            5.2            2.3
146  147            6.3           2.5            5.0            1.9
147  148            6.5           3.0            5.2            2.0
148  149            6.2           3.4            5.4            2.3
149  150            5.9           3.0            5.1            1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145 Iris-virginica
146 Iris-virginica
147 Iris-virginica
148 Iris-virginica
149 Iris-virginica

[150 rows x 6 columns]>
```

In [11]:
```
df.describe()
```

Out[11]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [13]:
```
# Let's now take a look at the number of instances that beleng to each class.
df.groupby('Species').size()
```

Out[13]:
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

In [14]:
```
df.columns
```

Out[14]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [15]:
```
df.corr()
```

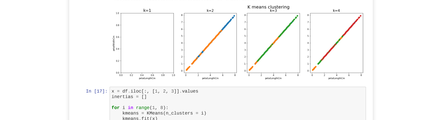
Out[15]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Id	1.000000	0.716676	-0.397729	0.882747	0.899759
SepalLengthCm	0.716676	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.397729	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.882747	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.899759	0.817954	-0.356544	0.962757	1.000000

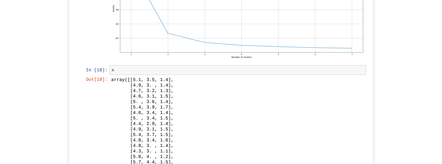
In [16]:
```
z=df.loc[:,['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm',]].values
```

In [24]:
```
plt.figure(figsize=(25,5))
plt.suptitle("K means clustering",fontsize= 20 )

plt.subplot(1,5,1)
plt.title("k=1",fontsize=16)
plt.xlabel("petallengthCm")
plt.ylabel("petalWidthCm")

plt.subplot(1,5,2)
plt.title("k=2",fontsize=16)
plt.xlabel("petallengthCm")
kmeans=KMeans(n_clusters=2)
feature=kmeans.fit_predict(z)
plt.scatter(z[feature==0],z[feature==0])
plt.scatter(z[feature==1],z[feature==1])

plt.subplot(1,5,3)
plt.title("k=3",fontsize=16)
plt.xlabel("petallengthCm")
kmeans=KMeans(n_clusters=3)
feature=kmeans.fit_predict(z)
plt.scatter(z[feature==0],z[feature==0])
plt.scatter(z[feature==1],z[feature==1])
plt.scatter(z[feature==2],z[feature==2])

plt.subplot(1,5,4)
plt.title("k=4",fontsize=16)
plt.xlabel("petallengthCm")
kmeans=KMeans(n_clusters=4)
feature=kmeans.fit_predict(z)
plt.scatter(z[feature==0],z[feature==0])
plt.scatter(z[feature==1],z[feature==1])
plt.scatter(z[feature==2],z[feature==2])
plt.scatter(z[feature==3],z[feature==3])
```

Out[24]:
```
<matplotlib.collections.PathCollection at 0x13cf1a09260>
```



In [17]:
```
x = df.iloc[:, [1, 2, 3]].values
inertias = []

for i in range(1, 8):
    kmeans = KMeans(n_clusters = i)
    kmeans.fit(x)
    inertias.append(kmeans.inertia_)

plt.figure(figsize=(20,5))
plt.title("WCSS / k chart", fontsize=18)
plt.grid(True)
plt.plot(range(1, 8), inertias)
plt.title('Elbow Graph',fontsize=16)
plt.xlabel('Number of clusters',fontsize="12")
plt.ylabel('Inertia', fontsize="12 )
plt.tight_layout()
plt.show()
```



In [18]:
```
x
```

Out[18]:
```
array([[5.1, 3.5, 1.4],
       [4.9, 3. , 1.4],
       [4.7, 3.2, 1.3],
       [4.6, 3.1, 1.5],
       [5. , 3.6, 1.4],
       [5.4, 3.9, 1.7],
       [4.6, 3.4, 1.4],
       [5. , 3.4, 1.5],
       [4.4, 2.9, 1.4],
       [4.9, 3.1, 1.5],
       [5.4, 3.7, 1.5],
       [4.8, 3.4, 1.6],
       [4.8, 3. , 1.4],
       [4.3, 3. , 1.1],
       [5.8, 4. , 1.2],
       [5.7, 4.4, 1.5],
       [5.4, 3.9, 1.3],
       [5.1, 3.5, 1.4],
       [5.7, 3.8, 1.7],
       [5.1, 3.8, 1.5],
       [5.4, 3.4, 1.7],
       [5.1, 3.7, 1.5],
       [4.6, 3.6, 1. ],
       [5.1, 3.3, 1.7],
       [4.8, 3.4, 1.9],
       [5. , 3. , 1.6],
       [5. , 3.4, 1.6],
       [5.2, 3.5, 1.5],
       [5.2, 3.4, 1.4],
       [4.7, 3.2, 1.6],
       [4.8, 3.1, 1.6],
       [5.4, 3.4, 1.5],
       [5.2, 4.1, 1.5],
       [5.5, 4.2, 1.4],
       [4.9, 3.1, 1.5],
       [5. , 3.2, 1.2],
       [5.5, 3.5, 1.3],
       [4.9, 3.1, 1.5],
       [4.4, 3. , 1.3],
       [5.1, 3.4, 1.5],
       [5. , 3.5, 1.3],
       [4.5, 2.3, 1.3],
       [4.4, 3.2, 1.3],
       [5. , 3.5, 1.6],
       [5.1, 3.8, 1.9],
       [4.8, 3. , 1.4],
       [5.1, 3.8, 1.6],
       [4.6, 3.2, 1.4],
       [5.3, 3.7, 1.5],
       [5. , 3.3, 1.4],
       [7. , 3.2, 4.7],
       [6.4, 3.2, 4.5],
       [6.9, 3.1, 4.9],
       [5.5, 2.3, 4. ],
       [6.5, 2.8, 4.6],
       [5.7, 2.8, 4.5],
       [6.3, 3.3, 4.7],
       [4.9, 2.4, 3.3],
       [6.6, 2.9, 4.6],
       [5.2, 2.7, 3.9],
       [5. , 2. , 3.5],
       [5.9, 3. , 4.2],
       [6. , 2.2, 4. ],
       [6.1, 2.9, 4.7],
       [5.6, 2.9, 3.6],
       [6.7, 3.1, 4.4],
       [5.6, 3. , 4.5],
       [5.8, 2.7, 4.1],
       [6.2, 2.2, 4.5],
       [5.6, 2.5, 3.9],
       [5.9, 3.2, 4.8],
       [6.1, 2.8, 4. ],
       [6.3, 2.5, 4.9],
       [6.1, 2.8, 4.7],
       [6.4, 2.9, 4.3],
       [6.6, 3. , 4.4],
       [6.8, 2.8, 4.8],
       [6.7, 3. , 5. ],
       [6. , 2.9, 4.5],
       [5.7, 2.6, 3.5],
       [5.5, 2.4, 3.8],
       [5.5, 2.4, 3.7],
       [5.8, 2.7, 3.9],
       [6. , 2.7, 5.1],
       [5.4, 3. , 4.5],
       [6. , 3.4, 4.5],
       [6.7, 3.1, 4.7],
       [6.3, 2.3, 4.4],
       [5.6, 3. , 4.1],
       [5.5, 2.5, 4. ],
       [5.5, 2.6, 4.4],
       [6.1, 3. , 4.6],
       [5.8, 2.6, 4. ],
       [5. , 2.3, 3.3],
       [5.6, 2.7, 4.2],
       [5.7, 3. , 4.2],
       [5.7, 2.9, 4.2],
       [6.2, 2.9, 4.3],
       [5.1, 2.5, 3. ],
       [5.7, 2.8, 4.1],
       [6.3, 3.3, 6. ],
       [5.8, 2.7, 5.1],
       [7.1, 3. , 5.9],
       [6.3, 2.9, 5.6],
       [6.5, 3. , 5.8],
       [7.6, 3. , 6.6],
       [4.9, 2.5, 4.5],
       [7.3, 2.9, 6.3],
       [6.7, 2.5, 5.8],
       [7.2, 3.6, 6.1],
       [6.5, 3.2, 5.1],
       [6.4, 2.7, 5.3],
       [6.8, 3. , 5.5],
       [5.7, 2.5, 5. ],
       [5.8, 2.8, 5.1],
       [6.4, 3.2, 5.3],
       [6.5, 3. , 5.5],
       [7.7, 3.8, 6.7],
       [7.7, 2.6, 6.9],
       [6. , 2.2, 5. ],
       [6.9, 3.2, 5.7],
       [5.6, 2.8, 4.9],
       [7.7, 2.8, 6.7],
       [6.3, 2.7, 4.9],
       [6.7, 3.3, 5.7],
       [7.2, 3.2, 6. ],
       [6.2, 2.8, 4.8],
       [6.1, 3. , 4.9],
       [6.4, 2.8, 5.6],
       [7.2, 3. , 5.8],
       [7.4, 2.8, 6.1],
       [7.9, 3.8, 6.4],
       [6.4, 2.8, 5.6],
       [6.3, 2.8, 5.1],
       [6.1, 2.6, 5.6],
       [7.7, 3. , 6.1],
       [6.3, 3.4, 5.6],
       [6.4, 3.1, 5.5],
       [6. , 3. , 4.8],
       [6.9, 3.1, 5.4],
       [6.7, 3.1, 5.6],
       [6.9, 3.1, 5.1],
       [5.8, 2.7, 5.1],
       [6.8, 3.2, 5.9],
       [6.7, 3.3, 5.7],
       [6.7, 3. , 5.2],
       [6.3, 2.5, 5. ],
       [6.5, 3. , 5.2],
       [6.2, 3.4, 5.4],
       [5.9, 3. , 5.1]])
```

In [19]:
```
kmeans = KMeans(n_clusters = 3, init = 'k-means++',
                max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(x)
```

In [20]:
```
# Predict the cluster labels: labels
labels = kmeans.predict(x)
```

In [21]:
```
labels
```

Out[21]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [23]:
```
plt.scatter(x[labels == 0, 0], x[labels == 0, 1],
            s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(x[labels == 1, 0], x[labels == 1, 1],
            s = 100, c = 'orange', label = 'Iris versicolour')
plt.scatter(x[labels == 2, 0], x[labels == 2, 1],
            s = 100, c = 'pink', label = 'Iris virginica')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1],
            s = 100, c = 'brown', label = 'Centroids')

plt.legend()
```

Out[23]:
```
<matplotlib.legend.Legend at 0x13cf1fdab70>
```

In [ ]:
```
```