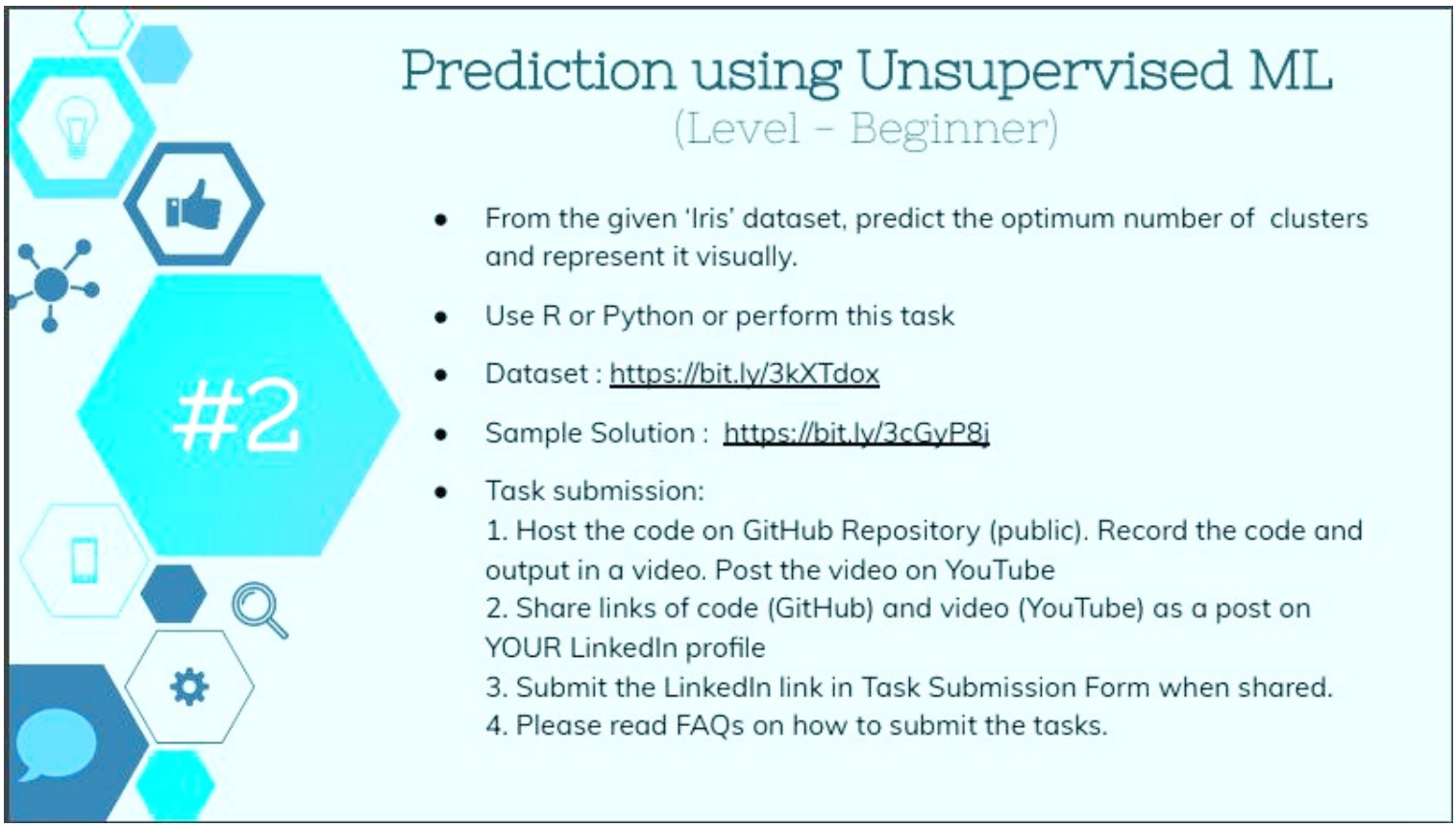
Graduate Rotational Internship Program

The Sparks Foundation

Author: Kartikey Gupta

TASK GIVEN:


#2
Prediction using Unsupervised ML
(Level - Beginner)
From the given 'Iris' dataset, predict the optimum number of clusters and represent it visually.
Use R or Python or perform this task
Dataset : https://bit.ly/3kXTdox
Sample Solution : https://bit.ly/3cGyP8j
Task submission:
1. Host the code on GitHub Repository (public). Record the code and output in a video. Post the video on YouTube
2. Share links of code (GitHub) and video (YouTube) as a post on YOUR LinkedIn profile
3. Submit the LinkedIn link in Task Submission Form when shared.
4. Please read FAQs on how to submit the tasks.

THE SPARKS FOUNDATION

DATA SCIENCE AND BUSINESS ANALYTICS INTERNSHIP

SUBMITTED BY– KARTIKEY GUPTA

Task Name: Prediction using Unsupervised ML

1. From the given 'Iris' dataset, predict the optimum number of clusters and represent it visually.
2. Use R or Python or perform this task.

Solution

Step1:Import Libraries

We'll import necessary libraries like pandas for making data , sklearn for loading dataset and for implementing K-Means algorithm, and matplotlib for plotting scatter plots.

Step2:Load Iris Dataset

We'll load iris dataset from sklearn default datasets.

Step3: Define Target and Predictors

Step4:Explore Dataset

We'll explore dataset by just plotting a scatter plot of dataset and will see all the data points.

Step 5: Finding Optimum No.of Clusters

We'll use elbow method to determine optimal number of clusters. The basic idea behind k-means clustering, is to define clusters such that the total within-cluster sum of square (WSS) is minimized. The total WSS measures the compactness of the clustering and we want it to be as small as possible.

The Elbow method looks at the total WSS as a function of the number of clusters: One should choose a number of clusters so that adding another cluster doesn't improve much better the total WSS.

The optimal number of clusters can be defined as follow:

1. Compute clustering algorithm (e.g., k-means clustering) for different values of k. For instance, by varying k from 1 to 10 clusters.
2. For each k, calculate the total within-cluster sum of square (wss).
3. Plot the curve of wss according to the number of clusters k.
4. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.

Step6: Applying K -Means

After finding optimal no. of clusters we'll apply K-Means algorithm on dataset.

K means works through the following iterative process:

1. Pick a value for k (the number of clusters to create).
2. Initialize k 'centroids' (starting points) in your data.
3. Create your clusters. Assign each point to the nearest centroid.
4. Make your clusters better. Move each centroid to the center of its cluster.
5. Repeat steps 3–4 until your centroids converge.

Step7: Visualizing Clusters

In the end we'll visualize the clusters.

ALLOTTED DATA:

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5.0	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3.0	1.4	0.1	Iris-setosa
14	4.3	3.0	1.1	0.1	Iris-setosa
15	5.8	4.0	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
18	5.1	3.5	1.4	0.3	Iris-setosa
19	5.7	3.8	1.7	0.3	Iris-setosa
20	5.1	3.8	1.5	0.3	Iris-setosa
21	5.4	3.4	1.7	0.2	Iris-setosa
22	5.1	3.7	1.5	0.4	Iris-setosa
23	4.6	3.6	1.0	0.2	Iris-setosa
24	5.1	3.3	1.7	0.5	Iris-setosa
25	4.8	3.4	1.9	0.2	Iris-setosa
26	5.0	3.0	1.6	0.2	Iris-setosa
27	5.0	3.4	1.6	0.4	Iris-setosa
28	5.2	3.5	1.5	0.2	Iris-setosa
29	5.2	3.4	1.4	0.2	Iris-setosa
30	4.7	3.2	1.6	0.2	Iris-setosa
31	4.8	3.1	1.6	0.2	Iris-setosa
32	5.4	3.4	1.5	0.4	Iris-setosa
33	5.2	4.1	1.5	0.1	Iris-setosa
34	5.5	4.2	1.4	0.2	Iris-setosa
35	4.9	3.1	1.5	0.1	Iris-setosa
36	5.0	3.2	1.2	0.2	Iris-setosa
37	5.5	3.5	1.3	0.2	Iris-setosa
38	4.9	3.1	1.5	0.1	Iris-setosa
39	4.4	3.0	1.3	0.2	Iris-setosa
40	5.1	3.4	1.5	0.2	Iris-setosa
41	5.0	3.5	1.3	0.3	Iris-setosa
42	4.5	2.3	1.3	0.3	Iris-setosa
43	4.4	3.2	1.3	0.2	Iris-setosa
44	5.0	3.5	1.6	0.6	Iris-setosa
45	5.1	3.8	1.9	0.4	Iris-setosa
46	4.8	3.0	1.4	0.3	Iris-setosa
47	5.1	3.8	1.6	0.2	Iris-setosa
48	4.6	3.2	1.4	0.2	Iris-setosa
49	5.3	3.7	1.5	0.2	Iris-setosa
50	5.0	3.3	1.4	0.2	Iris-setosa
51	7.0	3.2	4.7	1.4	Iris-versicolor
52	6.4	3.2	4.5	1.5	Iris-versicolor
53	6.9	3.1	4.9	1.5	Iris-versicolor
54	5.5	2.3	4.0	1.3	Iris-versicolor
55	6.5	2.8	4.6	1.5	Iris-versicolor

56	5.7	2.8	4.5	1.3	Iris-versicolor
57	6.3	3.3	4.7	1.6	Iris-versicolor
58	4.9	2.4	3.3	1.0	Iris-versicolor
59	6.6	2.9	4.6	1.3	Iris-versicolor
60	5.2	2.7	3.9	1.4	Iris-versicolor
61	5.0	2.0	3.5	1.0	Iris-versicolor
62	5.9	3.0	4.2	1.5	Iris-versicolor
63	6.0	2.2	4.0	1.0	Iris-versicolor
64	6.1	2.9	4.7	1.4	Iris-versicolor
65	5.6	2.9	3.6	1.3	Iris-versicolor
66	6.7	3.1	4.4	1.4	Iris-versicolor
67	5.6	3.0	4.5	1.5	Iris-versicolor
68	5.8	2.7	4.1	1.0	Iris-versicolor
69	6.2	2.2	4.5	1.5	Iris-versicolor
70	5.6	2.5	3.9	1.1	Iris-versicolor
71	5.9	3.2	4.8	1.8	Iris-versicolor
72	6.1	2.8	4.0	1.3	Iris-versicolor
73	6.3	2.5	4.9	1.5	Iris-versicolor
74	6.1	2.8	4.7	1.2	Iris-versicolor
75	6.4	2.9	4.3	1.3	Iris-versicolor
76	6.6	3.0	4.4	1.4	Iris-versicolor
77	6.8	2.8	4.8	1.4	Iris-versicolor
78	6.7	3.0	5.0	1.7	Iris-versicolor
79	6.0	2.9	4.5	1.5	Iris-versicolor
80	5.7	2.6	3.5	1.0	Iris-versicolor
81	5.5	2.4	3.8	1.1	Iris-versicolor
82	5.5	2.4	3.7	1.0	Iris-versicolor
83	5.8	2.7	3.9	1.2	Iris-versicolor
84	6.0	2.7	5.1	1.6	Iris-versicolor
85	5.4	3.0	4.5	1.5	Iris-versicolor
86	6.0	3.4	4.5	1.6	Iris-versicolor
87	6.7	3.1	4.7	1.5	Iris-versicolor
88	6.3	2.3	4.4	1.3	Iris-versicolor
89	5.6	3.0	4.1	1.3	Iris-versicolor
90	5.5	2.5	4.0	1.3	Iris-versicolor
91	5.5	2.6	4.4	1.2	Iris-versicolor
92	6.1	3.0	4.6	1.4	Iris-versicolor
93	5.8	2.6	4.0	1.2	Iris-versicolor
94	5.0	2.3	3.3	1.0	Iris-versicolor
95	5.6	2.7	4.2	1.3	Iris-versicolor
96	5.7	3.0	4.2	1.2	Iris-versicolor
97	5.7	2.9	4.2	1.3	Iris-versicolor
98	6.2	2.9	4.3	1.3	Iris-versicolor
99	5.1	2.5	3.0	1.1	Iris-versicolor
100	5.7	2.8	4.1	1.3	Iris-versicolor
101	6.3	3.3	6.0	2.5	Iris-virginica
102	5.8	2.7	5.1	1.9	Iris-virginica
103	7.1	3.0	5.9	2.1	Iris-virginica
104	6.3	2.9	5.6	1.8	Iris-virginica
105	6.5	3.0	5.8	2.2	Iris-virginica
106	7.6	3.0	6.6	2.1	Iris-virginica
107	4.9	2.5	4.5	1.7	Iris-virginica
108	7.3	2.9	6.3	1.8	Iris-virginica
109	6.7	2.5	5.8	1.8	Iris-virginica
110	7.2	3.6	6.1	2.5	Iris-virginica
111	6.5	3.2	5.1	2.0	Iris-virginica

112	6.4	2.7	5.3	1.9	Iris-virginica
113	6.8	3.0	5.5	2.1	Iris-virginica
114	5.7	2.5	5.0	2.0	Iris-virginica
115	5.8	2.8	5.1	2.4	Iris-virginica
116	6.4	3.2	5.3	2.3	Iris-virginica
117	6.5	3.0	5.5	1.8	Iris-virginica
118	7.7	3.8	6.7	2.2	Iris-virginica
119	7.7	2.6	6.9	2.3	Iris-virginica
120	6.0	2.2	5.0	1.5	Iris-virginica
121	6.9	3.2	5.7	2.3	Iris-virginica
122	5.6	2.8	4.9	2.0	Iris-virginica
123	7.7	2.8	6.7	2.0	Iris-virginica
124	6.3	2.7	4.9	1.8	Iris-virginica
125	6.7	3.3	5.7	2.1	Iris-virginica
126	7.2	3.2	6.0	1.8	Iris-virginica
127	6.2	2.8	4.8	1.8	Iris-virginica
128	6.1	3.0	4.9	1.8	Iris-virginica
129	6.4	2.8	5.6	2.1	Iris-virginica
130	7.2	3.0	5.8	1.6	Iris-virginica
131	7.4	2.8	6.1	1.9	Iris-virginica
132	7.9	3.8	6.4	2.0	Iris-virginica
133	6.4	2.8	5.6	2.2	Iris-virginica
134	6.3	2.8	5.1	1.5	Iris-virginica
135	6.1	2.6	5.6	1.4	Iris-virginica
136	7.7	3.0	6.1	2.3	Iris-virginica
137	6.3	3.4	5.6	2.4	Iris-virginica
138	6.4	3.1	5.5	1.8	Iris-virginica
139	6.0	3.0	4.8	1.8	Iris-virginica
140	6.9	3.1	5.4	2.1	Iris-virginica
141	6.7	3.1	5.6	2.4	Iris-virginica
142	6.9	3.1	5.1	2.3	Iris-virginica
143	5.8	2.7	5.1	1.9	Iris-virginica
144	6.8	3.2	5.9	2.3	Iris-virginica
145	6.7	3.3	5.7	2.5	Iris-virginica
146	6.7	3.0	5.2	2.3	Iris-virginica
147	6.3	2.5	5.0	1.9	Iris-virginica
148	6.5	3.0	5.2	2.0	Iris-virginica
149	6.2	3.4	5.4	2.3	Iris-virginica
150	5.9	3.0	5.1	1.8	Iris-virginica

ANALYSIS:

K-Means Clustering with Python Scikit Learn

In this section we will see how the Python Scikit-Learn library can be used to implement simple K-means Clutering algorithm in Machine Learning.

Task 2 : Predict the optimum number of clusters and represent it visually

For a give dataset "IRIS", predict the optimum number of clusters and reprenst it visually.

Author - Kartikey Gupta

Step 1: Importing Libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
```

Step 2: Loading Dataset

In [2]:
```
iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data, columns = iris.feature_names)
iris_df.head(10) # See the first 10 rows
```

Out[2]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1

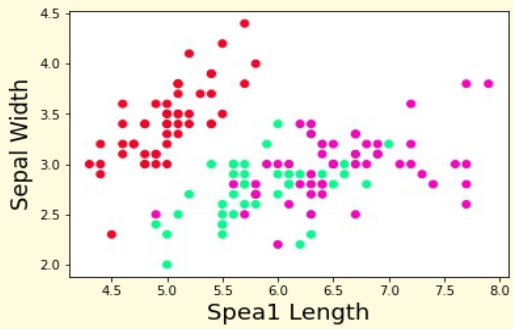
Step 3: Defining Target and Predictors

In [3]:
```
X = iris.data[:, :2]
y = iris.target
```

Step 4: Exploring Dataset

In [4]:
```
plt.scatter(X[:,0], X[:,1], c=y, cmap='gist_rainbow')
plt.xlabel('Spea1 Length', fontsize=18)
plt.ylabel('Sepal Width', fontsize=18)
```

Out[4]:
```
Text(0, 0.5, 'Sepal Width')
```

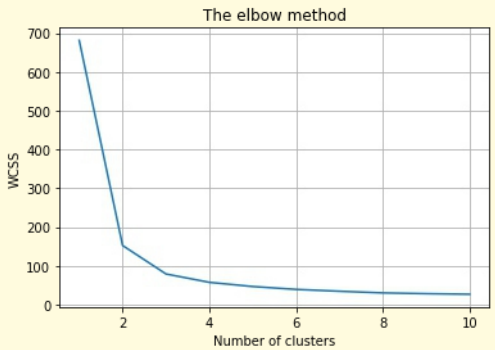


Step 5: Finding Optimum No. of Clusters

In [5]:
```
x = iris_df.iloc[ :, [0,1,2,3]].values
wcss = []

for i in range(1,11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit (x)
    wcss.append(kmeans.inertia_)
```

In [6]:
```
# Plotting results on graph to observe the elbow
plt.plot(range(1,11),wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') #Within cluster sum of squares
plt.grid(True)
plt.show()
```



Step 6: Applying K-Means

In [7]:
```
# Applying kmeans to the dataset / Creating the kmeans classifier
kmeans = KMeans(n_clusters = 3, init = 'k-means++',
                max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict (x)
```

Step 7: Visualizing Clusters

In [8]:
```
# Visualising the clusters - On the first two columns
plt.figure (figsize =(10,10))
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1],
            s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1],
            s = 100, c = 'blue', label = 'Iris-versicolour')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1],
            s = 100, c = 'green', label = 'Iris-virginica')

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

plt.legend()
```

Out[8]: <matplotlib.legend.Legend at 0x12ccc3c6700>