The Sparks Foundation

Data Science and Business Analytics Internship | GRIPJULY'21

Task-2 :Prediction through Unsupervised ML

Author -Tanushree Gaur

Use unsupervised ML to classify data points

Importing libraries

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [12]:
```
df=pd.read_csv('C:\\Users\\tanu.Umeshchand\\Downloads\\Iris.csv')
df.head()
```

Out[12]:

	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

In [13]:
```
df.shape
```

Out[13]:
```
(150, 6)
```

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

Out[14]:
```
<bound method DataFrame.info of        Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
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 [15]:
```
df.isnull().sum()
```

Out[15]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

Creating training data set

In [20]:
```
x=df.iloc[:,[0,1,2,3]].values
x
```

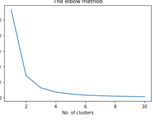
Out[20]:
```
array([[  1. ,   5.1,   3.5,   1.4],
       [  2. ,   4.9,   3. ,   1.4],
       [  3. ,   4.7,   3.2,   1.3],
       [  4. ,   4.6,   3.1,   1.5],
       [  5. ,   5. ,   3.6,   1.4],
       [  6. ,   5.4,   3.9,   1.7],
       [  7. ,   4.6,   3.4,   1.4],
       [  8. ,   5. ,   3.4,   1.5],
       [  9. ,   4.4,   2.9,   1.4],
       [ 10. ,   4.9,   3.1,   1.5],
       [ 11. ,   5.4,   3.7,   1.5],
       [ 12. ,   4.8,   3.4,   1.6],
       [ 13. ,   4.8,   3. ,   1.4],
       [ 14. ,   4.3,   3. ,   1.1],
       [ 15. ,   5.8,   4. ,   1.2],
       [ 16. ,   5.7,   4.4,   1.5],
       [ 17. ,   5.4,   3.9,   1.3],
       [ 18. ,   5.1,   3.5,   1.4],
       [ 19. ,   5.7,   3.8,   1.7],
       [ 20. ,   5.1,   3.8,   1.5],
       [ 21. ,   5.4,   3.4,   1.7],
       [ 22. ,   5.1,   3.7,   1.5],
       [ 23. ,   4.6,   3.6,   1. ],
       [ 24. ,   5.1,   3.3,   1.7],
       [ 25. ,   4.8,   3.4,   1.9],
       [ 26. ,   5. ,   3. ,   1.6],
       [ 27. ,   5. ,   3.4,   1.6],
       [ 28. ,   5.2,   3.5,   1.5],
       [ 29. ,   5.2,   3.4,   1.4],
       [ 30. ,   4.7,   3.2,   1.6],
       [ 31. ,   4.8,   3.1,   1.6],
       [ 32. ,   5.4,   3.4,   1.5],
       [ 33. ,   5.2,   4.1,   1.5],
       [ 34. ,   5.5,   4.2,   1.4],
       [ 35. ,   4.9,   3.1,   1.5],
       [ 36. ,   5. ,   3.2,   1.2],
       [ 37. ,   5.5,   3.5,   1.3],
       [ 38. ,   4.9,   3.1,   1.5],
       [ 39. ,   4.4,   3. ,   1.3],
       [ 40. ,   5.1,   3.4,   1.5],
       [ 41. ,   5. ,   3.5,   1.3],
       [ 42. ,   4.5,   2.3,   1.3],
       [ 43. ,   4.4,   3.2,   1.3],
       [ 44. ,   5. ,   3.5,   1.6],
       [ 45. ,   5.1,   3.8,   1.9],
       [ 46. ,   4.8,   3. ,   1.4],
       [ 47. ,   5.1,   3.8,   1.6],
       [ 48. ,   4.6,   3.2,   1.4],
       [ 49. ,   5.3,   3.7,   1.5],
       [ 50. ,   5. ,   3.3,   1.4],
       [ 51. ,   7. ,   3.2,   4.7],
       [ 52. ,   6.4,   3.2,   4.5],
       [ 53. ,   6.9,   3.1,   4.9],
       [ 54. ,   5.5,   2.3,   4. ],
       [ 55. ,   6.5,   2.8,   4.6],
       [ 56. ,   5.7,   2.8,   4.5],
       [ 57. ,   6.3,   3.3,   4.7],
       [ 58. ,   4.9,   2.4,   3.3],
       [ 59. ,   6.6,   2.9,   4.6],
       [ 60. ,   5.2,   2.7,   3.9],
       [ 61. ,   5. ,   2. ,   3.5],
       [ 62. ,   5.9,   3. ,   4.2],
       [ 63. ,   6. ,   2.2,   4. ],
       [ 64. ,   6.1,   2.9,   4.7],
       [ 65. ,   5.6,   2.9,   3.6],
       [ 66. ,   6.7,   3.1,   4.4],
       [ 67. ,   5.6,   3. ,   4.5],
       [ 68. ,   5.8,   2.7,   4.1],
       [ 69. ,   6.2,   2.2,   4.5],
       [ 70. ,   5.6,   2.5,   3.9],
       [ 71. ,   5.9,   3.2,   4.8],
       [ 72. ,   6.1,   2.8,   4. ],
       [ 73. ,   6.3,   2.5,   4.9],
       [ 74. ,   6.1,   2.8,   4.7],
       [ 75. ,   6.4,   2.9,   4.3],
       [ 76. ,   6.6,   3. ,   4.4],
       [ 77. ,   6.8,   2.8,   4.8],
       [ 78. ,   6.7,   3. ,   5. ],
       [ 79. ,   6. ,   2.9,   4.5],
       [ 80. ,   5.7,   2.6,   3.5],
       [ 81. ,   5.5,   2.4,   3.8],
       [ 82. ,   5.5,   2.4,   3.7],
       [ 83. ,   5.8,   2.7,   3.9],
       [ 84. ,   6. ,   2.7,   5.1],
       [ 85. ,   5.4,   3. ,   4.5],
       [ 86. ,   6. ,   3.4,   4.5],
       [ 87. ,   6.7,   3.1,   4.7],
       [ 88. ,   6.3,   2.3,   4.4],
       [ 89. ,   5.6,   3. ,   4.1],
       [ 90. ,   5.5,   2.5,   4. ],
       [ 91. ,   5.5,   2.6,   4.4],
       [ 92. ,   6.1,   3. ,   4.6],
       [ 93. ,   5.8,   2.6,   4. ],
       [ 94. ,   5. ,   2.3,   3.3],
       [ 95. ,   5.6,   2.7,   4.2],
       [ 96. ,   5.7,   3. ,   4.2],
       [ 97. ,   5.7,   2.9,   4.2],
       [ 98. ,   6.2,   2.9,   4.3],
       [ 99. ,   5.1,   2.5,   3. ],
       [100. ,   5.7,   2.8,   4.1],
       [101. ,   6.3,   3.3,   6. ],
       [102. ,   5.8,   2.7,   5.1],
       [103. ,   7.1,   3. ,   5.9],
       [104. ,   6.3,   2.9,   5.6],
       [105. ,   6.5,   3. ,   5.8],
       [106. ,   7.6,   3. ,   6.6],
       [107. ,   4.9,   2.5,   4.5],
       [108. ,   7.3,   2.9,   6.3],
       [109. ,   6.7,   2.5,   5.8],
       [110. ,   7.2,   3.6,   6.1],
       [111. ,   6.5,   3.2,   5.1],
       [112. ,   6.4,   2.7,   5.3],
       [113. ,   6.8,   3. ,   5.5],
       [114. ,   5.7,   2.5,   5. ],
       [115. ,   5.8,   2.8,   5.1],
       [116. ,   6.4,   3.2,   5.3],
       [117. ,   6.5,   3. ,   5.5],
       [118. ,   7.7,   3.8,   6.7],
       [119. ,   7.7,   2.6,   6.9],
       [120. ,   6. ,   2.2,   5. ],
       [121. ,   6.9,   3.2,   5.7],
       [122. ,   5.6,   2.8,   4.9],
       [123. ,   7.7,   2.8,   6.7],
       [124. ,   6.3,   2.7,   4.9],
       [125. ,   6.7,   3.3,   5.7],
       [126. ,   7.2,   3.2,   6. ],
       [127. ,   6.2,   2.8,   4.8],
       [128. ,   6.1,   3. ,   4.9],
       [129. ,   6.4,   2.8,   5.6],
       [130. ,   7.2,   3. ,   5.8],
       [131. ,   7.4,   2.8,   6.1],
       [132. ,   7.9,   3.8,   6.4],
       [133. ,   6.4,   2.8,   5.6],
       [134. ,   6.3,   2.8,   5.1],
       [135. ,   6.1,   2.6,   5.6],
       [136. ,   7.7,   3. ,   6.1],
       [137. ,   6.3,   3.4,   5.6],
       [138. ,   6.4,   3.1,   5.5],
       [139. ,   6. ,   3. ,   4.8],
       [140. ,   6.9,   3.1,   5.4],
       [141. ,   6.7,   3.1,   5.6],
       [142. ,   6.9,   3.1,   5.1],
       [143. ,   5.8,   2.7,   5.1],
       [144. ,   6.8,   3.2,   5.9],
       [145. ,   6.7,   3.3,   5.7],
       [146. ,   6.7,   3. ,   5.2],
       [147. ,   6.3,   2.5,   5. ],
       [148. ,   6.5,   3. ,   5.2],
       [149. ,   6.2,   3.4,   5.4],
       [150. ,   5.9,   3. ,   5.1]])
```

Finding the optimun numbers of cluster for k-means classification

In [47]:
```
from sklearn.cluster import KMeans
wcss[]
for i in range(1,11):
    kmeans=KMeans(n_clusters=i, init = 'k-means++', random_state=50 )
    kmeans.fit(x)
    wc.append(kmeans.inertia_)
```

Plotting the graphs

In [36]:
```
plt.plot(range(1,11),wc)
plt.title('The elbow method')
plt.xlabel('No. of clusters')
plt.ylabel('wc')
plt.show()
```



Applying k-means to the data with 4 as optimum no. of clusters

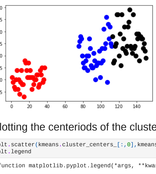
In [48]:
```
kmeans=KMeans(n_clusters=4, init='k-means++',random_state=50)
y_kmeans=kmeans.fit_predict(x)
y_kmeans
```

Out[48]:
```
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,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       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])
```

Visualising the cluster on first two columns

In [49]:
```
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 = 'black', label='Iris-virginica')
```

Out[49]:
```
<matplotlib.collections.PathCollection at 0x97d9ea3f10>
```



Plotting the centeriods of the cluster

In [50]:
```
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1] ,s = 200, c = 'yellow' , label='centriods')
plt.legend
```

Out[50]:
```
<function matplotlib.pyplot.legend(*args, **kwargs)>
```

Completed Task-2

Thankyou

Tanushree Gaur

In [ ]:
```
```