Experiment-5

Manan Shah

2019130059
TE Comps
Batch - C

Aim:
To train and test machine learning models using K-Means Clustering Algorithm.

Theory:

- K-Means Clustering is an unsupervised learning algorithm used in machine learning anddata science to handle clustering problems. It divides the unlabelled data into many clusters. K specifies the number of predetermined clusters that must be produced duringthe procedure; for example, if K=2, two clusters will be created, and if K=3, three clusterswill be created, and so on.
- How does the K-Means algorithm work?
    - The working of the K-Means algorithm is explained in the below steps:Step-1: Select the number K to decide the number of clusters.
      Step-2: Select random K points or centroids. (It can be different from the inputdataset).
      Step-3: Assign each data point to their closest centroid, which will form thepredefined K clusters.
      Step-4: Calculate the variance and place a new centroid of each cluster.
      Step-5: Repeat the third steps, which means assign each datapoint to the newclosest centroid of each cluster.
      Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.
      Step-7: The model is ready.

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

kMeans.ipynb

Users > manan > Documents > AI:ML > kMeans.ipynb > Loading the Dataset > Shape of Data > Data.shape

+ Code + Markdown | Outline ···

Prediction using Unsupervised ML

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

Importing required libraries.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
import seaborn as sns
from sklearn.cluster import KMeans
```

Plain Text

Loading the Dataset

```
iris = datasets.load_iris()
print("Dataset loaded successfully")
```

Plain Text

```
Dataset loaded successfully
```

Reading Dataset

```
#Creating data frame
Data = pd.DataFrame(iris.data, columns = iris.feature_names)

#Top values of Dataset
Data.head()
```

Plain Text

Restricted Mode ⊗ 0 ⚠ 0 Jupyter Server: local Cell 10 of 26

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

kMeans.ipynb

Users > manan > Documents > AI:ML > kMeans.ipynb > Loading the Dataset > Shape of Data > Data.shape

+ Code + Markdown | Outline ···

```
#Top values of Dataset
Data.head()
```

Plain Text

	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

```
#Bottom Values of Dataset
Data.tail()
```

Plain Text

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

Shape of Data

```
Data.shape
```

Plain Text

```
(150, 4)
```

Data Information

```
Data.info()
```

Plain Text

Restricted Mode ⊗ 0 ⚠ 0 Jupyter Server: local Cell 10 of 26

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

kMeans.ipynb

Users > manan > Documents > AI:ML > kMeans.ipynb > Loading the Dataset > Shape of Data > Data.shape

+ Code + Markdown | Outline ···

Data Information

```
Data.info()
```

Plain Text

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
dtypes: float64(4)
memory usage: 4.8 KB
```

From above one can clearly see that there is no null vlues.

Statistical Properties of Dataset

```
Data.describe()
```

Plain Text

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Restricted Mode ⊗ 0 ⚠ 0 Jupyter Server: local Cell 10 of 26

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

kMeans.ipynb

Users > manan > Documents > AI:ML > kMeans.ipynb > Loading the Dataset > Shape of Data > Data.shape

+ Code + Markdown | Outline ···

Data Information

```
Data.info()
```

Plain Text

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
dtypes: float64(4)
memory usage: 4.8 KB
```

From above one can clearly see that there is no null vlues.

Statistical Properties of Dataset

```
Data.describe()
```

Plain Text

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Restricted Mode ⊗ 0 ⚠ 0 Jupyter Server: local Cell 10 of 26

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

kMeans.ipynb

Users > manan > Documents > AI:ML > kMeans.ipynb > Loading the Dataset > Shape of Data > Data.shape

+ Code + Markdown Outline ···

```
Data.describe()
```

Plain Text

```
       sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
count         150.000000        150.000000         150.000000        150.000000
mean            5.843333          3.057333           3.758000          1.199333
std             0.828066          0.435866           1.765298          0.762238
min             4.300000          2.000000           1.000000          0.100000
25%             5.100000          2.800000           1.600000          0.300000
50%             5.800000          3.000000           4.350000          1.300000
75%             6.400000          3.300000           5.100000          1.800000
max             7.900000          4.400000           6.900000          2.500000
```

Data Visualization

```
sns.heatmap(Data.corr(), annot = True, linecolor='black')
```

Plain Text

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f61331362e8>
```

Restricted Mode ⊗ 0 ⚠ 0 Jupyter Server: local Cell 10 of 26

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

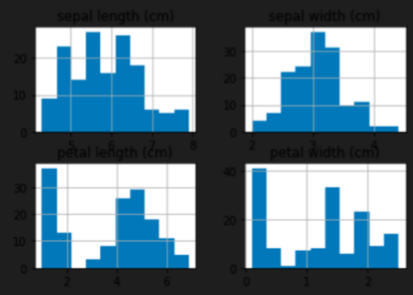
kMeans.ipynb

Users > manan > Documents > AI:ML > kMeans.ipynb > Loading the Dataset > Shape of Data > Data.shape

+ Code + Markdown Outline ···

```
Data.hist()
plt.show()
```

Plain Text



Finding numbers of Clusters for Kmeans using elbow method

```
# Settin the data
x=Data.iloc[:,0:3].values

css=[]

# Finding inertia on various k values
for i in range(1,8):
    kmeans=KMeans(n_clusters = i, init = 'k-means++',
                  max_iter = 100, n_init = 10, random_state = 0).fit(x)
    css.append(kmeans.inertia_)

plt.plot(range(1, 8), css, 'bx-', color='red')
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('CSS')
plt.show()
```

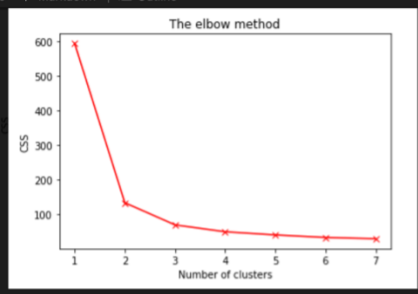
Restricted Mode ⊗ 0 ⚠ 0 Jupyter Server: local Cell 10 of 26

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

kMeans.ipynb

Users > manan > Documents > AI:ML > kMeans.ipynb > Loading the Dataset > Shape of Data > Data.shape

+ Code + Markdown | Outline ···



From one can clearly see that according to the elbow method most accurate value for number of cluseter is 3.

Applying KMeans Classifier

```
#Applying Kmeans classifier
kmeans = KMeans(n_clusters=3,init = 'k-means++', max_iter = 100, n_init = 10, random_state = 0)

y_kmeans = kmeans.fit_predict(x)
```

Plain Text

+ Code + Markdown

Visualizing the Clusters

```
kmeans.cluster_centers_
```

Plain Text

```
... array([[5.006     , 3.428     , 1.462     ],
           [5.84655172, 2.73275862, 4.3637931 ],
           [6.83571429, 3.06428571, 5.6547619 ]])
```

Restricted Mode ⊗ 0 ⚠ 0 Jupyter Server: local Cell 10 of 26

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

kMeans.ipynb

Users > manan > Documents > AI:ML > kMeans.ipynb > Loading the Dataset > Shape of Data > Data.shape

+ Code + Markdown | Outline ···

Visualizing the Clusters


```
kmeans.cluster_centers_
```

Plain Text

```
... array([[5.006     , 3.428     , 1.462     ],
           [5.84655172, 2.73275862, 4.3637931 ],
           [6.83571429, 3.06428571, 5.6547619 ]])
```

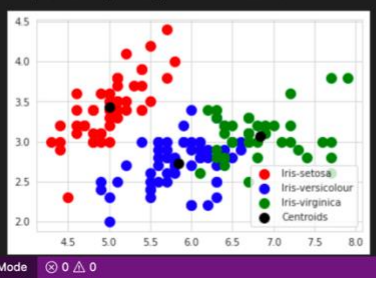

```
# Visualising the clusters - On the first two columns
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1],
            s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1],
            s = 100, c = 'blue', label = 'Iris-versicolour')
plt.scatter(x[y_kmeans == 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 = 'black', label = 'Centroids')

plt.legend()
```

Plain Text

```
... <matplotlib.legend.Legend at 0x7f612facceb8>
```



Restricted Mode ⊗ 0 ⚠ 0 Jupyter Server: local Cell 10 of 26

Conclusion:

- The experiment taught me the fundamentals of the K-Means approach. It's a centroid-based technique, thus each cluster has its own centroid.
- This technique's major purpose is to lower the sum of distances between data points and the clusters to which they belong.
- It finds the optimal value for K centre points or centroids via an iterative approach, and then assigns each data point to the closest k-centre. Data points that are close to a given K-center create a cluster.
- The algorithm starts with an unlabeled dataset, divides it into k clusters, then repeats the process until no better clusters can be found. The value of k should be predetermined in this algorithm.
- The algorithm's precision varies based on how many clusters are chosen.