EXPERIMENT – 9(b)

A python program using a K-Means Algorithm in a model

Aim:

To implement a python program using a K-Means Algorithm in a model.

Algorithm:

1. Import Necessary Libraries:
Import required libraries like numpy, matplotlib.pyplot, and sklearn.cluster.

2. Load and Preprocess Data:
Load the dataset.
Preprocess the data if needed (e.g., scaling).

3. Initialize Cluster Centers:
Choose the number of clusters (K).
Initialize K cluster centers randomly.

4. Assign Data Points to Clusters:
For each data point, calculate the distance to each cluster center.
Assign the data point to the cluster with the nearest center.

5. Update Cluster Centers:
Calculate the mean of the data points in each cluster.
Update the cluster centers to the calculated means.

6. Repeat Steps 4 and 5:
Repeat the assignment of data points to clusters and updating of cluster centers until
convergence (i.e., when the cluster assignments do not change much between iterations).

7. Plot the Clusters:
Plot the data points and the cluster centers to visualize the clustering result.

CODE 1:

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


data = pd.read_csv("IRIS_converted.csv")  # or "IRIS.csv" if you have it saved
data.head(5)
```

OUTPUT 1:

	sepal_ length	sepal_ width	petal_ length	Petal_ width	
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

CODE 2:

```
shuffle_index = np.random.permutation(data.shape[0])
req_data = data.iloc[shuffle_index]
req_data.head(5)
```

OUTPUT 2:

Sepal length	Sepal width	petal_ length	petal_ width	species	
7	5.0	3.4	1.5	0.2	Iris-setosa
22	6.9	3.1	4.9	1.5	Iris-versicolor
5	5.4	3.9	1.7	0.4	Iris-setosa
19	5.1	3.8	1.5	0.3	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

CODE 3:

```
train_size = int(req_data.shape[0] * 0.7)
train_df = req_data.iloc[:train_size, :]
test_df = req_data.iloc[train_size:, :]

train = train_df.values
test = test_df.values
y_true = test[:, -1]  # Actual species labels

print('Train_Shape:', train_df.shape)
print('Test_Shape:', test_df.shape)
```

OUTPUT 3:

```
Train_Shape: (21, 5)
```

Test_Shape: (10, 5)

CODE 4:

```
X = req_data.iloc[:, :-1].values  # features only


kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)

req_data['Cluster'] = kmeans.labels_
req_data.head(5)
```

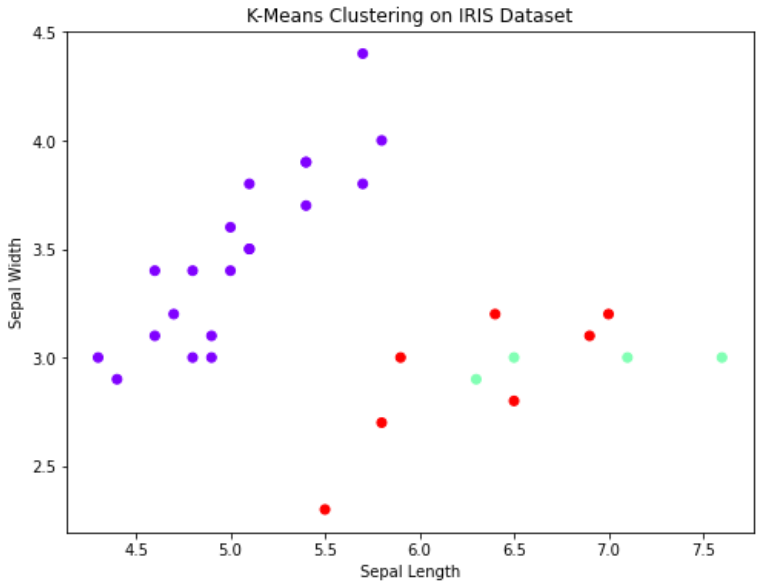
OUTPUT 4:

sepal_length	sepal_width	petal_length	petal_width	species	Cluster	
7	5.0	3.4	1.5	0.2	Iris-setosa	0
22	6.9	3.1	4.9	1.5	Iris-versicolor	2
5	5.4	3.9	1.7	0.4	Iris-setosa	0
19	5.1	3.8	1.5	0.3	Iris-setosa	0
4	5.0	3.6	1.4	0.2	Iris-setosa	0

CODE 5:

```
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_, cmap='rainbow')
plt.title("K-Means Clustering on IRIS Dataset")
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.show()
```

OUTPUT 5:



CODE 6:

```
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score

encoder = LabelEncoder()
```

true_labels = encoder.fit_transform(req_data['species'])
pred_labels = kmeans.labels_

accuracy = accuracy_score(true_labels, pred_labels)
print("Clustering Accuracy (approx):", round(accuracy, 3))

Clustering Accuracy (approx): 0.71

OUTPUT 6:

	Sepal length	sepal_ width	petal_ length	petal_ width	species
8	4.4	2.9	1.4	0.2	Iris-setosa
27	6.3	2.9	5.6	1.8	Iris-virginica
1	4.9	3.0	1.4	0.2	Iris-setosa
17	5.1	3.5	1.4	0.3	Iris-setosa
29	7.6	3.0	6.6	2.1	Iris-virginica

RESULT:

Thus a python program to use K-Means Algorithm in a model is written and the output is verified.