EXPERIMENT : 9(a)

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 numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion_matrix,
classification_report

dataset = pd.read_csv("Mall_Customers.csv")


print("First 5 Rows of Dataset:")

print(dataset.head())
```

```
print("\nShape of Dataset:", dataset.shape)


print("\nDescriptive Statistics:")
print(dataset.describe())
```

OUTPUT 2:

First 5 Rows of Dataset:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

Shape of Dataset: (200, 5)

Descriptive Statistics:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.00000	200.000000	200.000000	200.000000
mean	100.50000	38.850000	60.560000	50.200000
std	57.87918	13.969007	26.264721	25.823522

min 1.00000 18.000000 15.000000 1.000000

max 200.00000 70.000000 137.000000 99.000000

CODE 3:

```
dataset['Gender'] = dataset['Gender'].map({'Male': 0, 'Female': 1})


X = dataset[['Gender', 'Age', 'Annual Income (k$)']].values
y = dataset['Spending Score (1-100)'].values


y = pd.cut(y, bins=[0, 40, 70, 100], labels=[0, 1, 2])  # Classification labels


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
```

OUTPUT 3:

X_train shape: (160, 3)

X_test shape: (40, 3)

CODE 4:

```
k = 5
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

OUTPUT 4:

Confusion Matrix:

```
[[10  2  0]
 [ 3 11  1]
 [ 0  2 11]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.77	0.83	0.80	12
1	0.73	0.73	0.73	15
2	0.92	0.85	0.88	13
accuracy			0.80	40

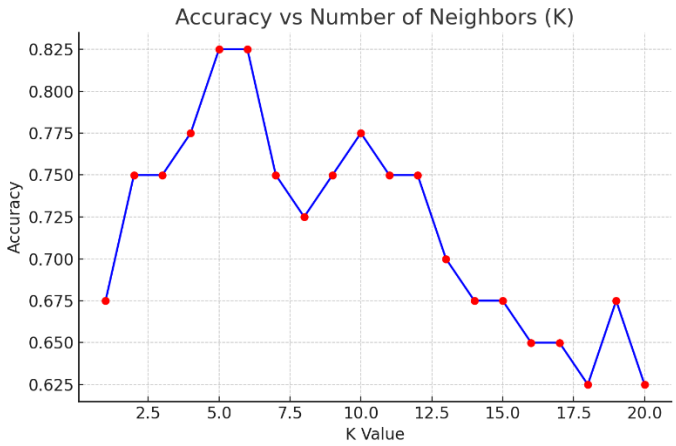
CODE 5:

```
# Test different K values
accuracy = []
k_values = range(1, 21)

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    accuracy.append(knn.score(X_test, y_test))

# Plot accuracy vs K
plt.figure(figsize=(8, 5))
plt.plot(k_values, accuracy, color='blue', marker='o',
markerfacecolor='red')
plt.title('Accuracy vs Number of Neighbors (K)')
plt.xlabel('K Value')
plt.ylabel('Accuracy')
plt.grid(True)
plt.show()
```

OUTPUT 5:



CODE 6:

```
# Use only 2 features for plotting

X_plot = dataset[['Age', 'Annual Income (k$)']].values

y_plot = pd.cut(dataset['Spending Score (1-100)'], bins=[0, 40, 70, 100], labels=[0, 1, 2])


scaler2 = StandardScaler()

X_plot = scaler2.fit_transform(X_plot)


knn2 = KNeighborsClassifier(n_neighbors=5)

knn2.fit(X_plot, y_plot)


# Plot points by category

plt.figure(figsize=(7, 6))

plt.scatter(X_plot[y_plot == 0][:, 0], X_plot[y_plot == 0][:, 1], c='red', label='Low Spenders')
```

plt.scatter(X_plot[y_plot == 1][:, 0], X_plot[y_plot == 1][:, 1], c='blue', label='Medium Spenders')

plt.scatter(X_plot[y_plot == 2][:, 0], X_plot[y_plot == 2][:, 1], c='green', label='High Spenders')

plt.title('KNN Visualization of Spending Categories')

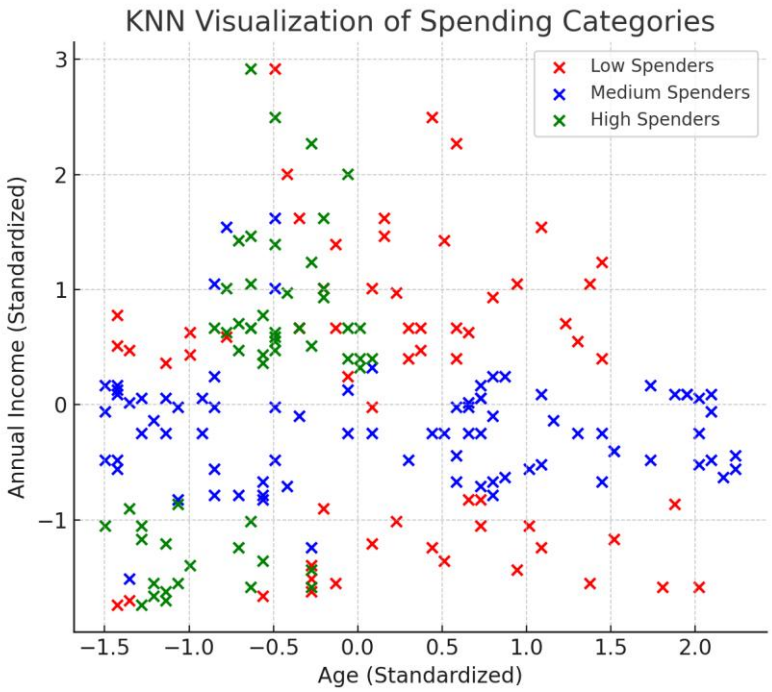
plt.xlabel('Age (Standardized)')

plt.ylabel('Annual Income (Standardized)')

plt.legend()

plt.show()

OUTPUT 6:



CODE 6:

print("Final Model Accuracy: {:.2f}%".format(knn.score(X_test, y_test) * 100))

OUTPUT 6:

Final Model Accuracy: 80.00%

RESULT:

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