**Batch: B2 Experiment Number:4**

**Roll Number: 16010421073 Name:Keyur Patel**

## **Aim of the Experiment:** To implement any four K-value selection algorithms along with K means clustering algorithm.

**Program/ Steps:**

**K-mean**

import pandas as pd

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

# Load the dataset

data = pd.read\_csv("samplecalifornia\_housing\_test.csv")

# Select the relevant features for clustering (modify this as needed) selected\_features = data[['GDP ($ per capita)', 'Literacy (%)']]

selected\_features = data[['median\_income', 'median\_house\_value']]

# Perform K-Means clustering with K=3

kmeans = KMeans(n\_clusters=3, random\_state=0)

data['Cluster'] = kmeans.fit\_predict(selected\_features)

# Visualize the clusters plt.figure(figsize=(10, 8))

plt.figure(figsize=(10, 8))

plt.scatter(data['median\_income'], data['median\_house\_value'], c=data['Cluster'], cmap='viridis')

plt.title('K-Means Clustering (K=3)')

plt.xlabel('median\_income')

plt.ylabel('median\_house\_value')

plt.show()

K NEAREST NEIGHBOURS:

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import mean\_squared\_error

# Load your data (replace 'data.csv' with the actual file path)

data = pd.read\_csv('sample\_data/california\_housing\_test.csv')

# Define the features for clustering

features\_for\_clustering = data[['housing\_median\_age', 'total\_rooms', 'total\_bedrooms', 'population', 'households', 'median\_income']]

# Perform K-Means clustering

n\_clusters = 3  # You can choose the number of clusters

kmeans = KMeans(n\_clusters=n\_clusters, random\_state=42)

data['Cluster'] = kmeans.fit\_predict(features\_for\_clustering)

# Define the features for regression

features\_for\_regression = data[['housing\_median\_age', 'total\_rooms', 'total\_bedrooms', 'population', 'households', 'median\_income']]

# Define the target variable

target = data['median\_house\_value']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features\_for\_regression, target, test\_size=0.2, random\_state=42)

# Standardize the feature data

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Choose the value of k (number of neighbors)

k = 5  # You can experiment with different values

# Create a KNN model

knn\_model = KNeighborsRegressor(n\_neighbors=k)

# Fit the model to the training data

knn\_model.fit(X\_train, y\_train)

# Make predictions on the test data

y\_pred = knn\_model.predict(X\_test)

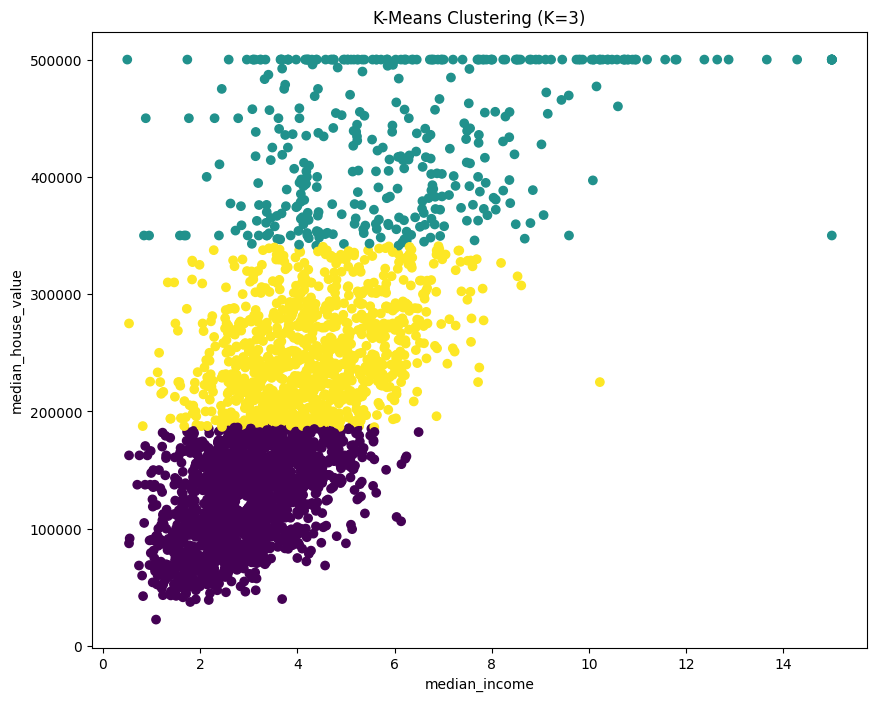
# Calculate the mean squared error to evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred)

print(f"Mean Squared Error: {mse}")

**Output/Result:**

**K-mean**



K NEAREST NEIGHBOURS:



**Post Lab Question-Answers:**

**1. What is the main difference between K-means and K-nearest neighbours?**

K-Means and K-Nearest Neighbors (KNN) are both machine learning algorithms, but they serve different purposes and have distinct characteristics:

1. **Type of Algorithm:**

* K-Means is an unsupervised clustering algorithm used for grouping similar data points into clusters

based on their feature similarity. It's used for exploratory data analysis and finding hidden patterns in the data.

* KNN, on the other hand, is a supervised classification algorithm used for assigning a category label to a data point based on the majority category of its K-nearest neighbors in the feature space. It's used for classification tasks.

1. **Purpose:**

* K-Means aims to partition data points into distinct, non-overlapping clusters, making it useful for segmenting data into natural groups based on similarity.
* KNN is primarily used for classifying data points into predefined classes or categories based on the class labels of their nearest neighbors.

1. **Training:**

* K-Means is an unsupervised learning algorithm and doesn't require training on labeled data. It identifies clusters by minimizing the distance between data points and cluster centroids.
* KNN is a supervised learning algorithm, meaning it requires a labeled dataset for training. It uses the labeled data to find the class labels of the nearest neighbors when making predictions.

1. **Number of Clusters or Neighbors:**

* In K-Means, you need to specify the number of clusters (K) a priori. The algorithm aims to partition the data into K clusters.
* In KNN, you specify the number of nearest neighbors (K) to consider when making predictions. The choice of K is a hyperparameter and depends on the problem and the dataset.

1. **Output:**

* K-Means assigns each data point to one and only one cluster, and the output is the cluster labels.
* KNN assigns a data point to a class label based on the majority class among its K-nearest neighbors, and the output is a class label or category.

1. **Algorithm Complexity:**

* K-Means involves iterative optimization to minimize the sum of squared distances between data points and cluster centroids. The algorithm can be computationally intensive, especially for large datasets.
* KNN has a simple and direct classification process where it computes distances between data points and their neighbors. It can be computationally expensive during prediction when there's a need to calculate distances to all training data points.

In summary, K-Means is used for unsupervised clustering and identifying natural groupings in data, while KNN is used for supervised classification based on the majority class of its nearest neighbors. The choice between these two algorithms depends on the nature of your data and the specific problem you are trying to solve.

**2. What are some stopping criteria for k-means clustering?**

K-Means clustering is an iterative algorithm that seeks to minimize the sum of squared distances between data points and cluster centroids. To prevent the algorithm from running indefinitely and to determine when to stop the iteration, various stopping criteria can be used. Here are some common stopping criteria for K-Means clustering:

1. **Convergence:** The most common stopping criterion for K-Means is to check for convergence. You can monitor the change in cluster assignments or the centroids between iterations. When the change falls below a certain threshold or the algorithm reaches a predefined number of iterations, you can stop.
2. **Maximum Number of Iterations:** You can set a predetermined maximum number of iterations. If the algorithm reaches this limit and hasn't converged, it stops. This is often used as a safeguard to prevent long-running or infinite loops.
3. **No Change in Cluster Assignments:** If, during an iteration, no data points change their cluster assignment, the algorithm has likely converged. You can stop the algorithm in such cases to save computational resources.
4. **Minimum Cluster Size:** You can set a minimum cluster size threshold. If a cluster has fewer data points than this threshold, it may be too small to be meaningful, and you can stop the algorithm.
5. **Percentage Change in Inertia:** Monitor the percentage change in the sum of squared distances (inertia) between iterations. If the change is very small, the algorithm has likely converged.
6. **Silhouette Score:** Calculate the Silhouette Score for the clusters at each iteration. If the score stops improving significantly, it can be an indication that the algorithm is close to convergence.

The choice of stopping criteria may depend on the specific problem, the nature of your data, and your computational resources. It's common to use a combination of these criteria to ensure that the K-Means algorithm stops efficiently and effectively.

**Outcomes: CO3**Comprehend radial-basis-function (RBF) networks and Kernel learning method

**Conclusion (based on the Results and outcomes achieved):**

**Thus Successfully implemented K value selection algorithms along with K mean clustering alg.**

**References:**

## Books/ Journals/ Websites: