**Department of Electrical Engineering and   
Computer Science**

**Faculty Member:** Dr. Ahmad Salman **Dated:** 11/10/2023

**Semester:** 7th **Section:** BEE 12C

**CS-471 Machine Learning**

Lab 5: K-Means Clustering

**Group Members**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **PLO4 - CLO4** | | **PLO5 -CLO5** | **PLO8 -CLO6** | **PLO9 -CLO7** |
| **Name** | **Reg. No** | **Viva / Quiz / Lab Performance** | **Analysis of Data in Lab Report** | **Modern Tool Usage** | **Ethics and Safety** | **Individual and Teamwork** |
|  |  | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** |
| Afif Arif Siddiqi | 344504 |  |  |  |  |  |
| Muhammad Ali Farooq | 331879 |  |  |  |  |  |
| Muhammad Ahmed Mohsin | 333060 |  |  |  |  |  |
| Danial Ahmad | 331388 |  |  |  |  |  |
| Ehtishaam Tanveer | 333074 |  |  |  |  |  |
| Muhammad Umer | 345834 |  |  |  |  |  |

**Table of Contents**

[2 K-Means Clustering 3](#_Toc148106241)

[2.1 Introduction 3](#_Toc148106242)

[2.2 Objectives 3](#_Toc148106243)

[2.3 Theory 3](#_Toc148106244)

[3 Lab Tasks 4](#_Toc148106245)

[3.1 Task 1 – 2-Means Clustering 4](#_Toc148106246)

[3.2 Task 2 – K-Means Clustering 5](#_Toc148106247)

[3.3 Task 3 – Cost Function 8](#_Toc148106248)

[3.4 Task 4 – Your Own Dataset 10](#_Toc148106249)

[4 Conclusion 12](#_Toc148106250)

# K-Means Clustering

## Introduction

This laboratory exercise is focused on K-means clustering which is a widely used unsupervised learning technique. Clustering is used on unlabeled data to look for interesting groups and patterns.

## Objectives

The following are the main objectives of this lab:

* Understand the basics of K-means clustering, including its algorithm and steps.
* Apply the K-means clustering algorithm to a real-world dataset.
* Evaluate the performance of a K-means clustering model.
* Interpret the results of K-means clustering analysis.

## Theory

K-means clustering is an unsupervised learning technique that is used to find groups, clusters or patterns in unlabeled. As the dataset is not labelled, only the arrangement of the inputs on the feature space are available. In K-means clustering, K number of clusters are set and then the examples are compared to the cluster centroids. The distance of each feature is used as a metric to define which cluster it belongs to. The cluster centroids are iteratively shifted and the examples belonging to them also change. After enough iterations, useful groups in the feature space are obtained. To determine the best number for clusters, a cost function can be calculated for each K number.

# Lab Tasks

## Task 1 – 2-Means Clustering

In this task, use the provided dataset. Write the code which performs clustering of the dataset into 2 clusters. The pseudocode for the clustering algorithm is provided as follows:

specify K number of centroids

randomly initialize K number of centroids u

for j = 1:epochs

for i = 1:m

c(i) = index of closest cluster to training example

for k = 1:K

u(k) = mean of all training examples indexed to k

plot of x1 and x2 clusters

To determine the index c(i), you will need to write a function that calculates the Euclidean distance between the points in the feature space. This function will be used to find the closest centroid from each training example. After determining the indexes, the cluster centroids themselves are updated by taking the average of the x values. For k-th cluster, the training examples with index k will be averaged. This completes one iteration of clustering after which a scatter plot is made. The iterations are repeated until interesting groups are obtained in the plots.

Due to the initial randomization of cluster centroids, you may have to repeat the clustering a few times. Also, ensure the random centroids are from within the domain of the feature space.

### TASK 1 CODE STARTS HERE ###

*# Distance function*

*def* compute\_euclidean\_distance(*x1*, *x2*):

    return np.sqrt(np.sum((x1 - x2) \*\* 2))

*# Load data*

data = pd.read\_csv(path\_data)

data = data.iloc[:, :-1]

*# Cluster data*

*def* cluster\_data(*K*, *epochs*, *max\_range*, *data*, *plots*=False):

*# Initialize arrays*

    u = np.random.rand(K, 2) \* max\_range

    c = np.zeros(len(data))

    for j in range(epochs):

*# Calculate distances to centroids*

        distances = np.linalg.norm(data.values[:, None, :] - u, *axis*=2)

*# Assign closest centroid index*

        c = np.argmin(distances, *axis*=1)

*# Calculate new centroids*

        u = np.array([np.mean(data.iloc[c == k], *axis*=0) for k in range(K)])

*# Plot*

        if plots:

            plt.scatter(data.iloc[:, 0], data.iloc[:, 1], *c*=c, *s*=3)

            plt.scatter(u[:, 0], u[:, 1], *c*="red", *marker*="x")

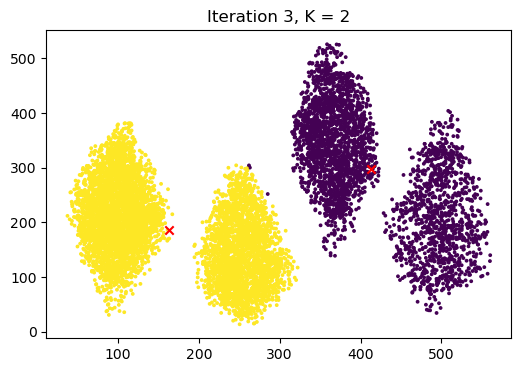
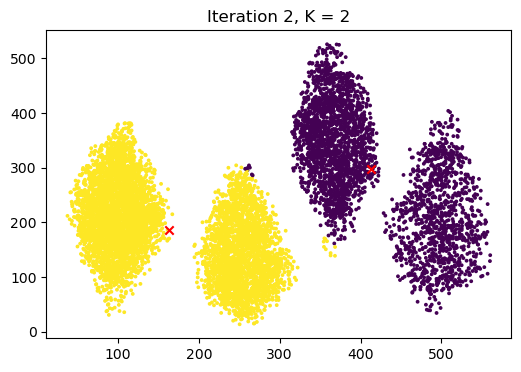
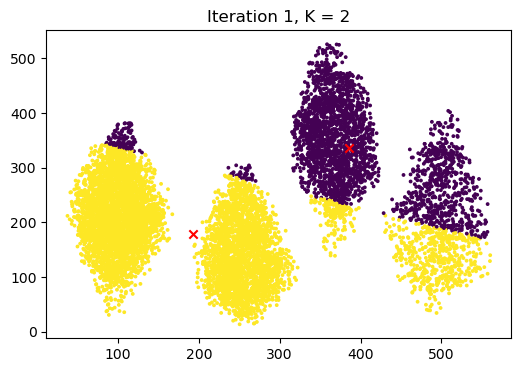
            plt.title(*f*"Iteration {j + 1}, K = {K}")

            plt.show()

cluster\_data(2, 5, 500, data, *plots*=True)

### TASK 1 CODE ENDS HERE ###

### TASK 1 OUTPUT SCREENSHOT STARTS HERE ###



### TASK 1 OUTPUT SCREENSHOT ENDS HERE ###

## Task 2 – K-Means Clustering

Load dataset 1 into a dataframe and perform the following

1. Print the dataset using the head and tail functions
2. Print any 3 rows from the dataset
3. Print any 5 elements from the dataset
4. Use the mean, mode and median functions for each column in the dataset

Provide all the codes and screenshots of the final output.

### TASK 2 CODE STARTS HERE ###

*# Load data*

data = pd.read\_csv(path\_data)

data = data.iloc[:, :-1]

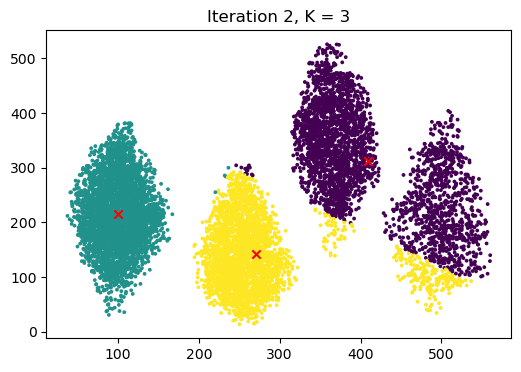
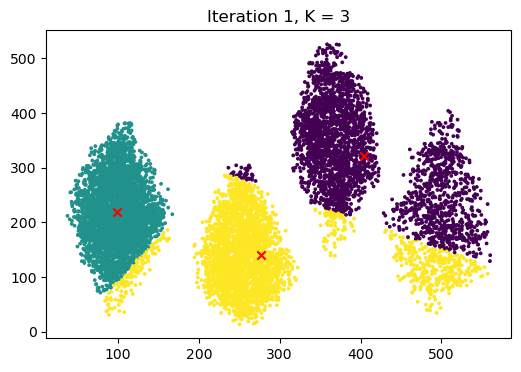
cluster\_data(3, 5, 500, data, *plots*=True)

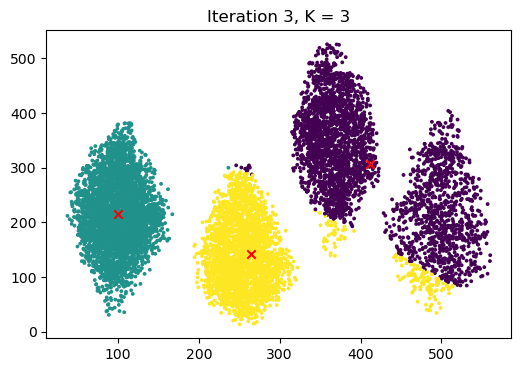
cluster\_data(4, 5, 500, data, *plots*=True)

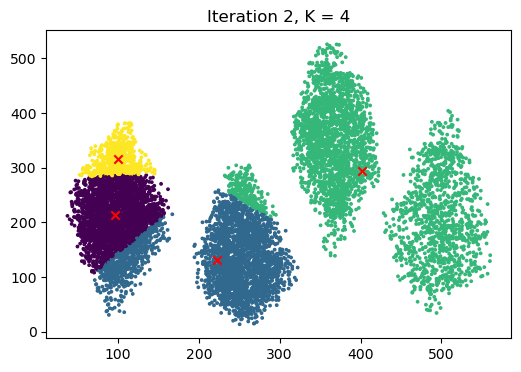
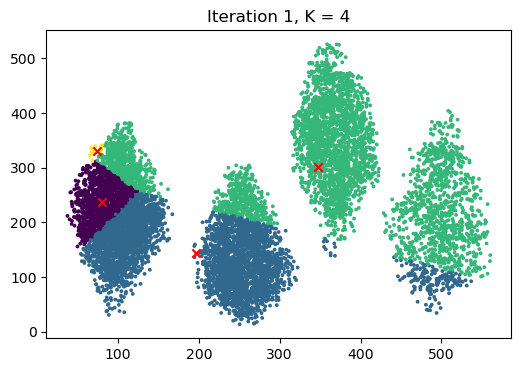
cluster\_data(5, 5, 500, data, *plots*=True)

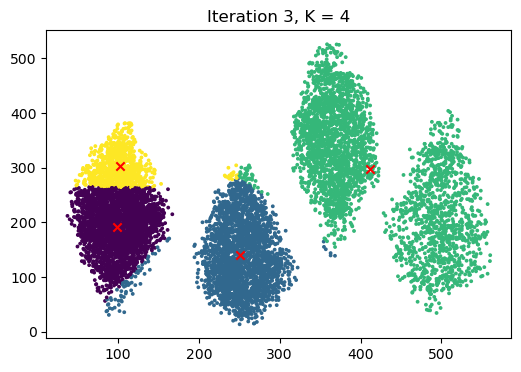
### TASK 2 CODE ENDS HERE ###

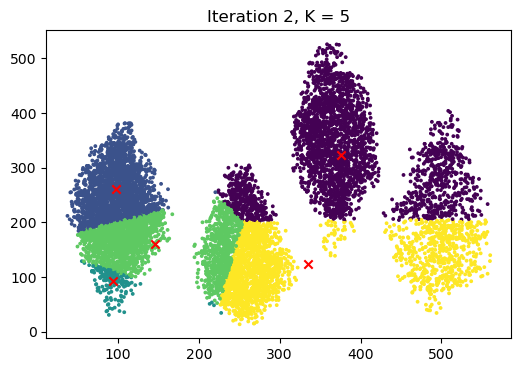
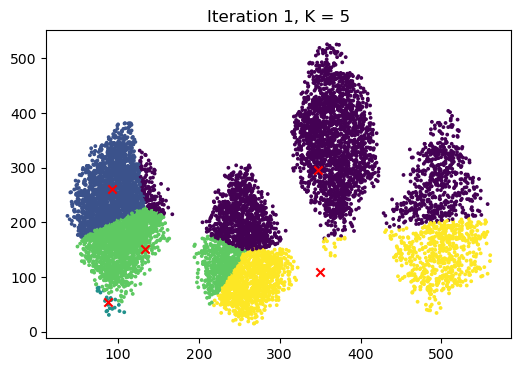
### TASK 2 OUTPUT SCREENSHOTS START HERE ###

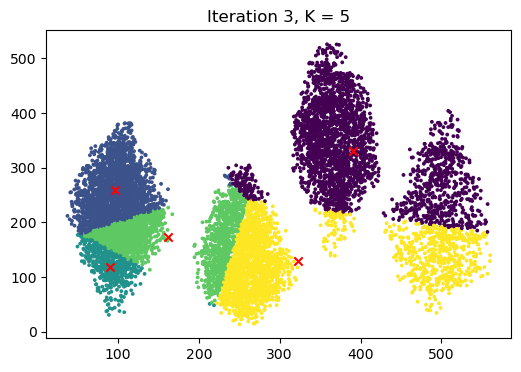












### TASK 2 OUTPUT SCREENSHOTS END HERE ###

## Task 3 – Cost Function

Load the given dataset into the python program for this task. In this task, you will modify your code so that it performs clustering from

k = 2, 3, 4… 10.

For each k value, perform 20 iterations (epochs) of centroid update before moving to the next value of k. Additionally, at the last iteration, determine the cost for that K-value:

for K = 2:10

randomly initialize K number of centroids u

for j = 1:epochs

for i = 1:m

compute c(i)

for k = 1:K

compute u(k)

plot of x1 and x2 clusters

compute cost for current K value

plot of cost and K

Store the costs for each k in a list. After the last iteration of the last cluster, make a plot of k vs. cost.

### TASK 3 CODE STARTS HERE ###

*# Load data*

data = pd.read\_csv(path\_data)

data = data.iloc[:, :-1]

*# Cluster data*

*def* cluster\_data(*K*, *epochs*, *max\_range*, *data*, *plots*=False):

*# Initialize arrays*

    u = np.random.rand(K, 2) \* max\_range

    c = np.zeros(len(data))

    for j in range(epochs):

*# Calculate distances to centroids*

        distances = np.linalg.norm(data.values[:, None, :] - u, *axis*=2)

*# Assign closest centroid index*

        c = np.argmin(distances, *axis*=1)

*# Calculate new centroids*

        u = np.array([np.mean(data.iloc[c == k], *axis*=0) for k in range(K)])

*# Plot*

        if plots:

            plt.scatter(data.iloc[:, 0], data.iloc[:, 1], *c*=c, *s*=3)

            plt.scatter(u[:, 0], u[:, 1], *c*="red", *marker*="x")

            plt.title(*f*"Iteration {j + 1}, K = {K}")

            plt.show()

*# cost  1 / m \* np.sum(np.linalg.norm(x - u, axis=1) \*\* 2)*

    cost = 1 / len(data) \* np.sum(np.linalg.norm(data.values - u[c], *axis*=1) \*\* 2)

    return cost

costs = []

for k in range(2, 11):

    costs.append(cluster\_data(k, 20, 500, data))

*# Customize the plot*

plt.plot(

    range(2, 11), costs,

*color*="blue", *marker*="o",

*linestyle*="dashed", *linewidth*=1,

*markersize*=5)

*# Add labels and title*

plt.xlabel("Number of clusters (K)")

plt.ylabel("Cost")

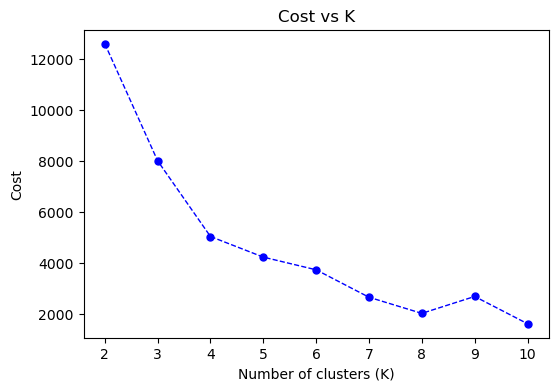
plt.title("Cost vs K")

*# Show the plot*

plt.show()

### TASK 3 CODE ENDS HERE ###

### TASK 3 OUTPUT SCREENSHOT STARTS HERE ###



### TASK 3 OUTPUT SCREENSHOT ENDS HERE ###

## Task 4 – Your Own Dataset

Download your own CSV dataset from the internet (e.g. Kaggle). Your dataset must have at least 500 rows and at least 4 feature columns. Perform clustering of your dataset, showcase the plots and provide explanation.

### TASK 4 CODE STARTS HERE ###

*# Load data*

data = pd.read\_csv("lab5\_task4.csv")

data = data[

    [

        "MedInc",

        "MedHouseVal",

    ]

]

*# Cluster data*

*def* cluster\_data(*K*, *epochs*, *max\_range*, *data*, *plots*=False):

*# Initialize arrays*

    u = np.random.rand(K, 2) \* max\_range

    c = np.zeros(len(data))

    for j in range(epochs):

*# Calculate distances to centroids*

        distances = np.linalg.norm(data.values[:, None, :] - u, *axis*=2)

*# Assign closest centroid index*

        c = np.argmin(distances, *axis*=1)

*# Calculate new centroids*

        u = np.array([np.mean(data.iloc[c == k], *axis*=0) for k in range(K)])

*# Plot*

        if plots:

            plt.scatter(data.iloc[:, 0], data.iloc[:, 1], *c*=c, *s*=3)

            plt.scatter(u[:, 0], u[:, 1], *c*="red", *marker*="x")

            plt.title(*f*"Iteration {j + 1}, K = {K}")

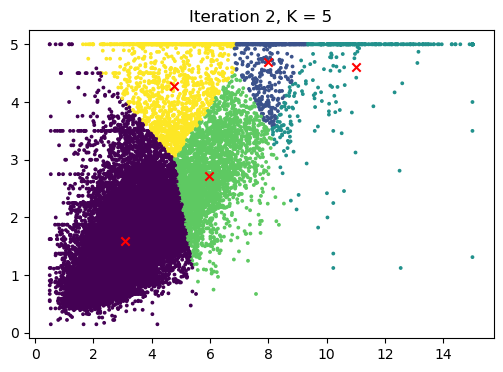
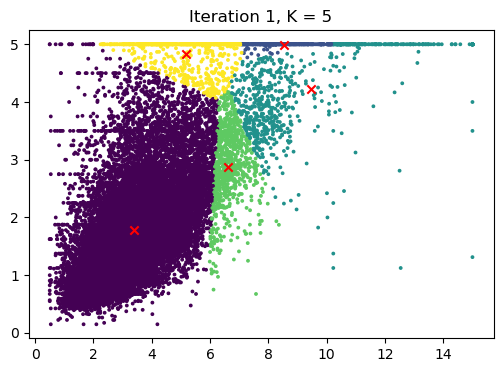
            plt.show()

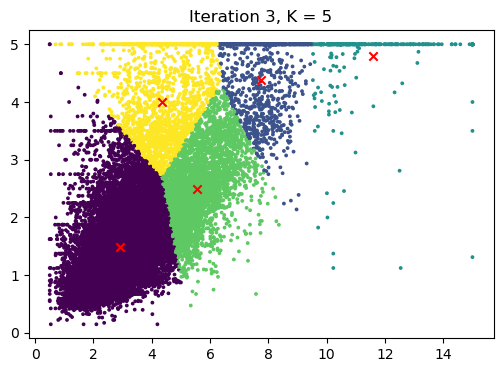
cluster\_data(5, 5, np.max(data.iloc[:, 0] + data.iloc[:, 1]) / 2,

data, *plots*=True)

### TASK 4 CODE ENDS HERE ###

### TASK 4 OUTPUT SCREENSHOTS START HERE ###





### TASK 4 OUTPUT SCREENSHOTS START HERE ###

# Conclusion

In this laboratory exercise, we learned about K-means clustering, a widely used unsupervised learning technique. We implemented the K-means clustering algorithm in Python and applied it to real-world data. We saw that K-means clustering can be used to identify interesting groups and patterns in unlabeled data. This can be useful for a variety of tasks, such as market segmentation, customer profiling, and fraud detection.