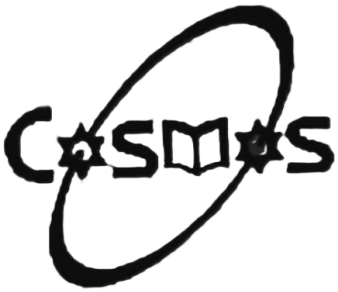
Cosmos College of Management & Technology

(Affiliated to Pokhara University)

Sitapaila, Kathmandu



Artifical Intelligence (CMP 346) Lab Report

LAB REPORT NO: 06

LAB REPORT ON:
PATTERN RECOGNITION

SUBMITTED BY:-

Name: Kushal Prasad Joshi

Roll No: 230345

Faculty: Science and Technology

Group: B

SUBMITTED TO:-

Department: ICT

Lab Instructor: Mr. Ranjan Raj Aryal

Date of Experiment: 18th Janaury 2026

Date of Submission: 23rd Janaury 2026

TITLE

PATTERN RECOGNITION

OBJECTIVES

1. To understand the basic concepts of pattern recognition.
2. To implement KNN algorithm.
3. To implement K-Means Clustering algorithm.

REQUIREMENTS

- Python
- VS Code
- Libraries: OpenCV, NumPy, Matplotlib

THEORY

Pattern recognition is a branch of machine learning that focuses on the identification and classification of patterns in data. It involves the use of algorithms and statistical methods to recognize patterns and make predictions based on the identified patterns.

K-Nearest Neighbors (kNN): kNN is one of the simplest of classification algorithms available for supervised learning. The idea is to search for closest match of the test data in feature space. We will look into it with below image.

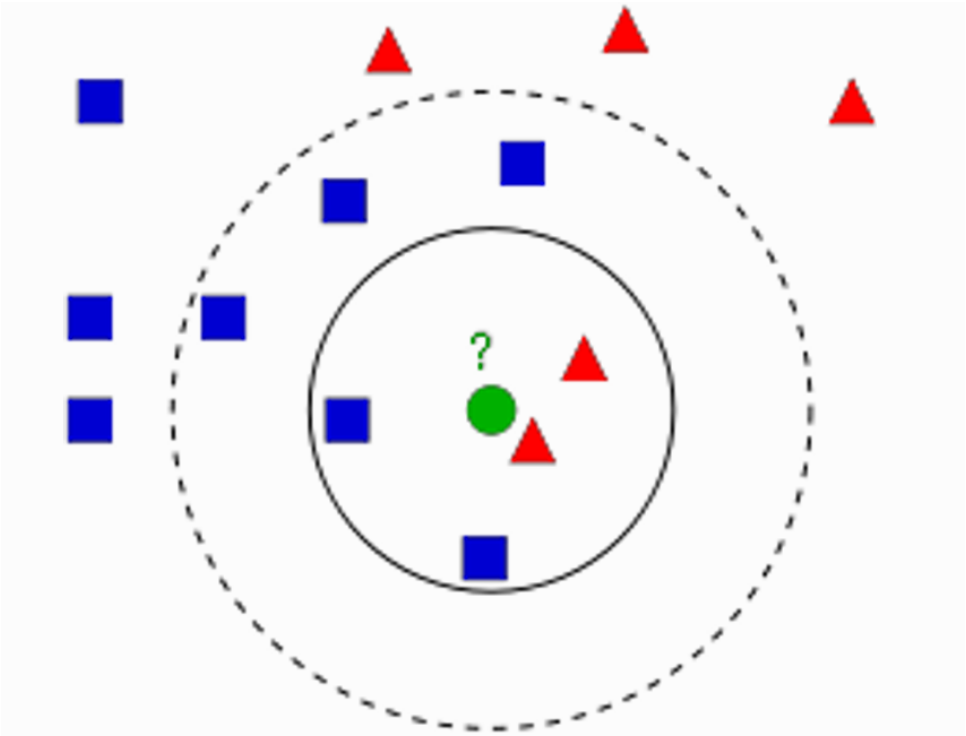


Figure 1: K-Nearest Neighbors

In the image, there are two families, Blue Squares and Red Triangles. We call each family as Class. Their houses are shown in their town map which we call feature space. (You can consider a feature space as a space where all data are projected. For example, consider a 2D coordinate space. Each data has two features, x and y coordinates. You can represent this data in your 2D coordinate space. Now imagine if there are three features, you need 3D space. Now consider N features, where you need N-dimensional space, This N-dimensional space is its feature space. In this image, you can consider it as a 2D case with two features).

Now a new member comes into the town and creates a new home, which is shown as green circle. He should be added to one of these Blue/Red families. We call that process, Classification now, let us apply this algorithm, kNN.

One method is to check who is his nearest neighbor. From the image, it is clear it is the Red Triangle family. So he is also added into Red Triangle. This method is called simply Nearest Neighbor, because classification depends only on the nearest neighbor.

K-Means Clustering: K-Means Clustering is an unsupervised machine learning algorithm used to group similar data points into clusters based on their features. The algorithm works by iteratively assigning data points to the nearest cluster centroid and updating the centroids based on the assigned points until convergence.

IMPLEMENTATION

```
1 import cv2
2 import numpy as np
```

```
import matplotlib.pyplot as plt

# Feature set containing (x,y) values of 25 known/training data
trainData = np.random.randint(0, 100, (25, 2)).astype(np.float32)

# Labels each one either Red or Blue with numbers 0 and 1
responses = np.random.randint(0, 2, (25, 1)).astype(np.float32)

# Take Red families and plot them
red = trainData[responses.ravel() == 0]
plt.scatter(red[:, 0], red[:, 1], 80, c='r', marker='^')

# Take Blue families and plot them
blue = trainData[responses.ravel() == 1]
plt.scatter(blue[:, 0], blue[:, 1], 80, c='b', marker='s')
plt.show()
```

Listing 1: K-Nearest Neighbors Implementation

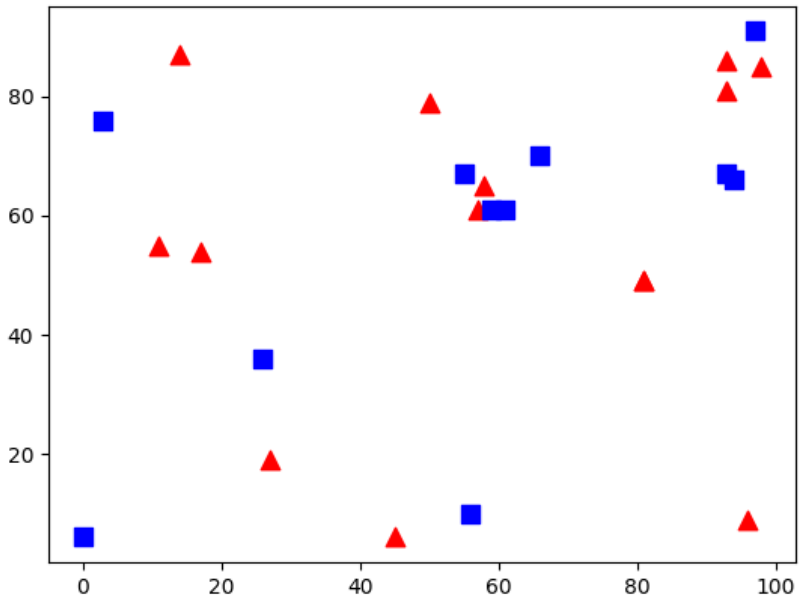


Figure 2: Output of K-Nearest Neighbors Implementation

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

```

```
X = np.random.randint(25,50,(25,2))
Y = np.random.randint(60,85,(25,2))
Z = np.vstack((X,Y))

# convert to np.float32
Z = np.float32(Z)

# define criteria and apply kmeans()
criteria = (cv2.TERM_CRITERIA_EPS + cv2.
    TERM_CRITERIA_MAX_ITER, 10, 1.0)
ret,label,center=cv2.kmeans(Z,2,None,criteria,10,cv2.
    KMEANS_RANDOM_CENTERS)

# Now separate the data, Note the flatten()
A = Z[label.ravel()==0]
B = Z[label.ravel()==1]

# Plot the data
plt.scatter(A[:,0],A[:,1])
plt.scatter(B[:,0],B[:,1],c = 'r')
plt.scatter(center[:,0],center[:,1],s = 80,c = 'y', marker =
    's')
plt.xlabel('Height'),plt.ylabel('Weight')
plt.show()
```

Listing 2: K-Means Clustering Implementation

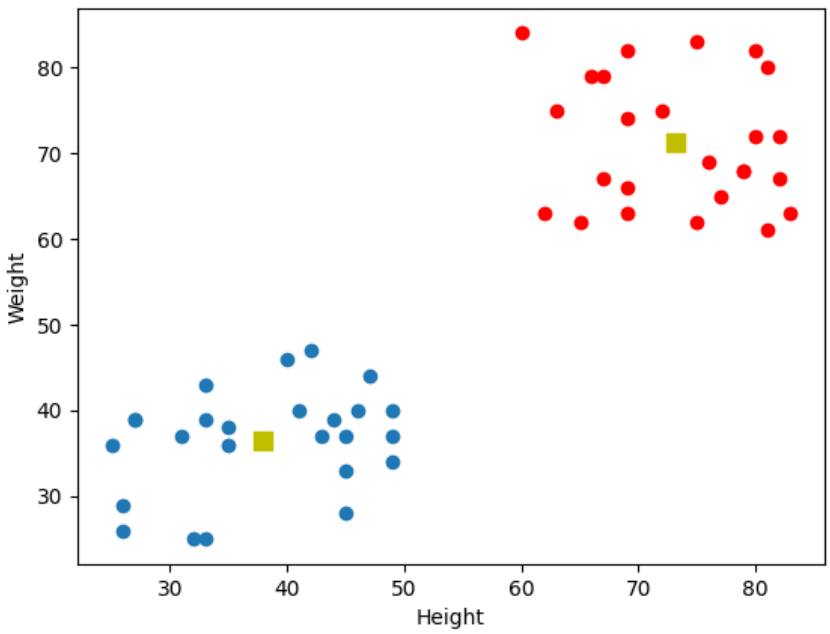


Figure 3: Output of K-Means Clustering Implementation

```
"""
Question:
Solve the problem using K-Means Clustering method and analyze the results.

Given:
- Generate two sets of random data:
  x in the range [25, 100]
  y in the range [175, 255]
- Combine them into a single dataset
- Plot the histogram
- Apply K-Means clustering
- Analyze the clustering result
"""

# ------- IMPORT REQUIRED LIBRARIES --------
import numpy as np
import cv2
from matplotlib import pyplot as plt

# ---------- DATA GENERATION ----------
# Generate 25 random values between 25 and 100
x = np.random.randint(25, 100, 25)

# Generate 25 random values between 175 and 255
y = np.random.randint(175, 255, 25)

# Combine both arrays into a single array
z = np.hstack((x, y))

# Reshape data into a column vector (required by OpenCV K-Means)
z = z.reshape((50, 1))

# Convert data type to float32 (required by OpenCV)
z = np.float32(z)

# ----- HISTOGRAM OF ORIGINAL DATA ------
plt.hist(z, 256, [0, 256])
plt.title("Histogram of Original Data")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()

# ------- APPLY K-MEANS CLUSTERING ------
# Define stopping criteria:
# Stop after 10 iterations or when accuracy reaches epsilon =
```

```
    1.0
criteria = (cv2.TERM_CRITERIA_EPS + cv2.
    TERM_CRITERIA_MAX_ITER,
            10, 1.0)

# Number of clusters (since data has two distinct groups)
K = 2

# Apply K-Means clustering
ret, label, center = cv2.kmeans(
    z,                          # input data
    K,                          # number of clusters
    None,                       # initial labels
    criteria,                   # termination criteria
    10,                         # number of attempts
    cv2.KMEANS_RANDOM_CENTERS
)

# ------ DISPLAY CLUSTER CENTERS -------
print("Cluster Centers:")
print(center)

# ------- SEPARATE DATA BASED ON CLUSTERS ------
cluster1 = z[label.ravel() == 0]
cluster2 = z[label.ravel() == 1]

# ------- PLOT CLUSTERED DATA --------
plt.hist(cluster1, 256, [0, 256], alpha=0.6, label="Cluster 1
    ")
plt.hist(cluster2, 256, [0, 256], alpha=0.6, label="Cluster 2
    ")

# Plot cluster centers as vertical lines
plt.axvline(center[0], color='black', linestyle='--', label='
    Center 1')
plt.axvline(center[1], color='red', linestyle='--', label='
    Center 2')

plt.title("K-Means Clustering Result (K = 2)")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.legend()
plt.show()

# ------ RESULT ANALYSIS -------
"""
Analysis:
- The histogram of the original data shows two distinct peaks
```

```
    .
- K-Means with K = 2 successfully separates the data into two
    clusters.
- One cluster corresponds to low-range values (25-100).
- The other cluster corresponds to high-range values
   (175-255).
- Cluster centers represent the mean value of each group.
- Since the data is well separated, K-Means converges quickly
    and accurately.
"""

print("K-Means clustering successfully divided the data into
   two distinct clusters.")
```

Listing 3: K-Means Clustering Implementation

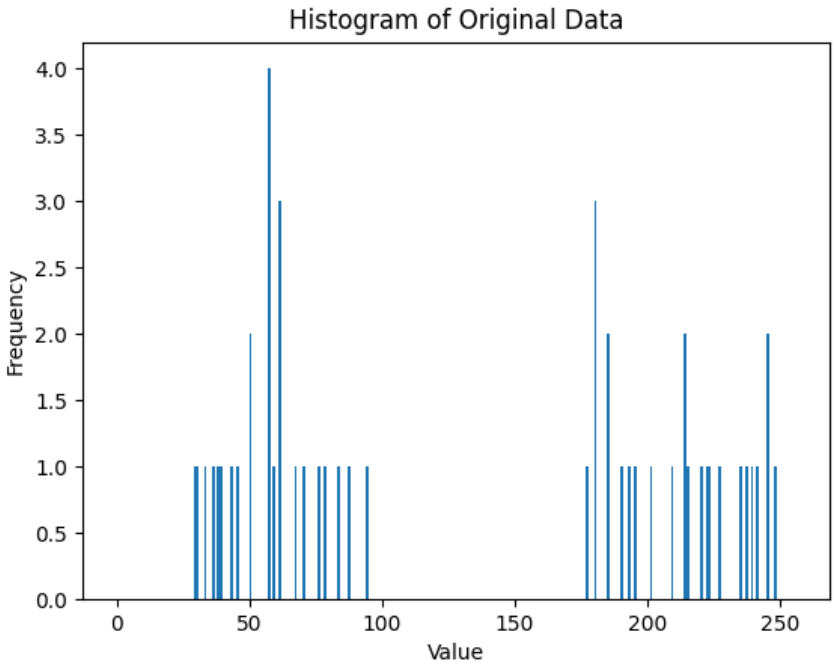


Figure 4: Output of K-Means Clustering Question

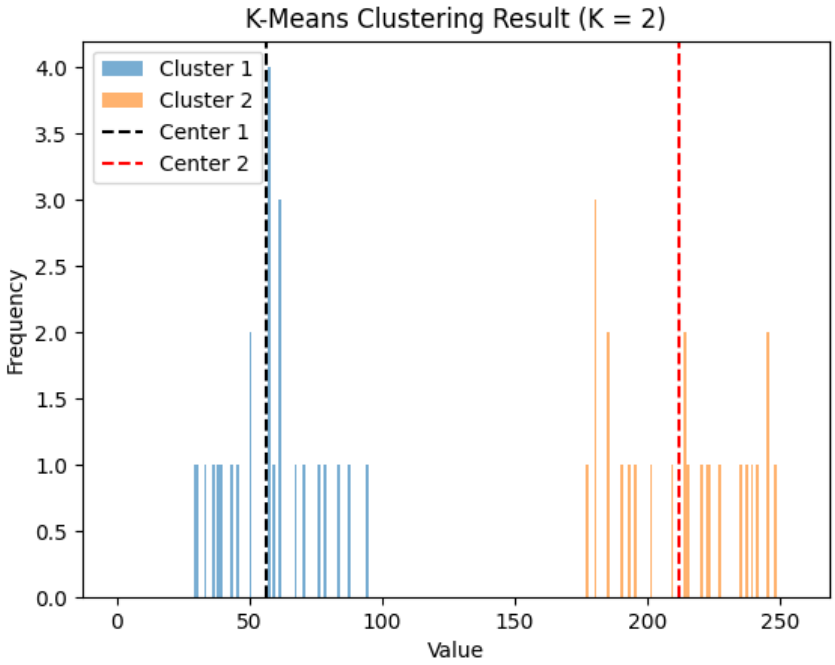


Figure 5: Output of K-Means Clustering Solution Histogram

```
K-Means clustering successfully divided the data into two
   distinct clusters.
```

Listing 4: Output of the K-Means Clustering Problem Implementation

RESULTS AND CONCLUSION

In this lab, we explored the concepts of pattern recognition through the implementation of K-Nearest Neighbors (kNN) and K-Means Clustering algorithms. The kNN algorithm effectively classified data points based on their proximity to known classes, while the K-Means Clustering algorithm successfully grouped similar data points into distinct clusters. The implementations demonstrated the practical applications of these algorithms in pattern recognition tasks.