NAME:PAREENITA A.SHIRSATH PRN:221101062 ROLL.NO:57 T.E.A.I.&.D.S.

**ML EXPERIMENT NO : 5**

**CODE:**

from google.colab import files

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

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

from sklearn.svm import SVC

from sklearn.metrics import classification\_report, accuracy\_score

# Upload the dataset

uploaded = files.upload()

# Load the dataset

df = pd.read\_csv("Mall\_Customers.csv")

# Display first few rows

display(df.head())

# Selecting relevant features

X = df[['Annual Income (k$)', 'Spending Score (1-100)']]

# Standardizing the features

scaler = StandardScaler()

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

# Creating target variable (Clustering customers into 2 groups for simplicity)

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters=2, random\_state=42)

y = kmeans.fit\_predict(X\_scaled)

# Splitting data into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

# Training SVM model

svm\_model = SVC(kernel='rbf', random\_state=42)

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

# Predictions

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

# Evaluation

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

# Visualization

plt.figure(figsize=(8,6))

sns.scatterplot(x=X\_scaled[:, 0], y=X\_scaled[:, 1], hue=y, palette='coolwarm')

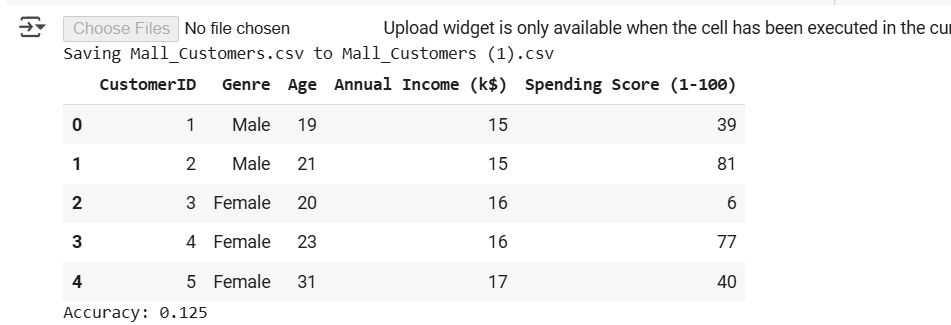
plt.title("Customer Segmentation using SVM")

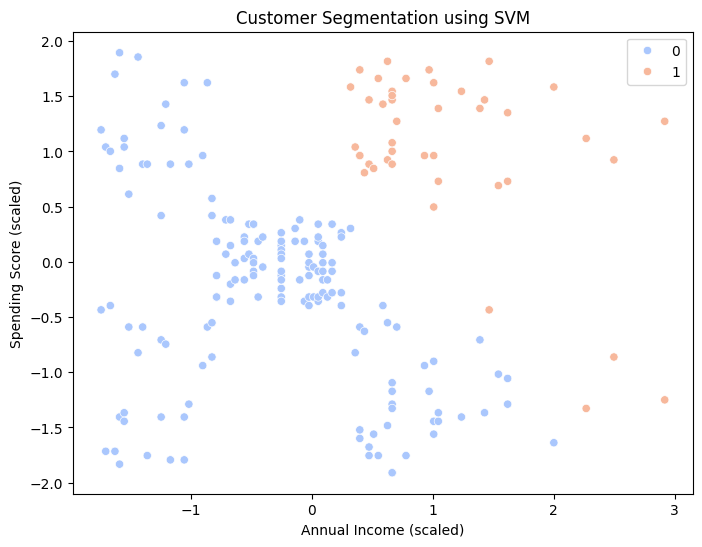
plt.xlabel("Annual Income (scaled)")

plt.ylabel("Spending Score (scaled)")

plt.show()

**OUTPUT:**

****



NAME:PAREENITA A.SHIRSATH PRN:221101062 ROLL.NO:57 T.E.A.I.&.D.S.

**ML EXPERIMENT NO : 6**

**CODE:**

from google.colab import files

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

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

# Upload the dataset

uploaded = files.upload()

# Load the dataset

df = pd.read\_csv("Mall\_Customers.csv")

# Display first few rows

display(df.head())

# Selecting relevant features

X = df[['Annual Income (k$)', 'Spending Score (1-100)']]

# Standardizing the features

scaler = StandardScaler()

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

# Initializing weights

weights = np.random.rand(X\_scaled.shape[1])

bias = np.random.rand()

learning\_rate = 0.01

# Hebbian Learning Function

def hebbian\_learning(X, y, weights, bias, learning\_rate):

    for i in range(len(X)):

        weights += learning\_rate \* y[i] \* X[i]

        bias += learning\_rate \* y[i]

    return weights, bias

# Creating target variable (Clustering customers into 2 groups for simplicity)

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters=2, random\_state=42)

y = kmeans.fit\_predict(X\_scaled)

# Splitting data into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

# Training using Hebbian Learning

weights, bias = hebbian\_learning(X\_train, y\_train, weights, bias, learning\_rate)

# Predictions

y\_pred = np.sign(np.dot(X\_test, weights) + bias)

# Evaluation

accuracy = np.mean(y\_pred == y\_test)

print("Accuracy:", accuracy)

# Visualization

plt.figure(figsize=(8,6))

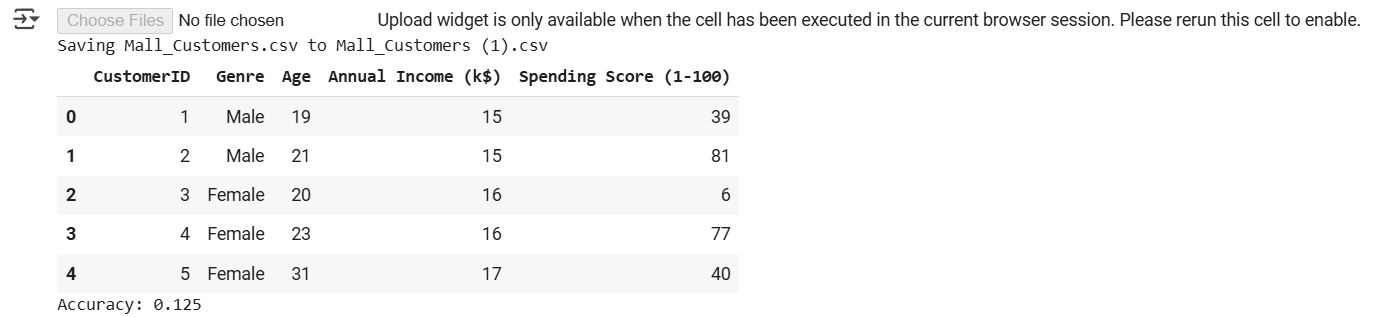
sns.scatterplot(x=X\_scaled[:, 0], y=X\_scaled[:, 1], hue=y, palette='coolwarm')

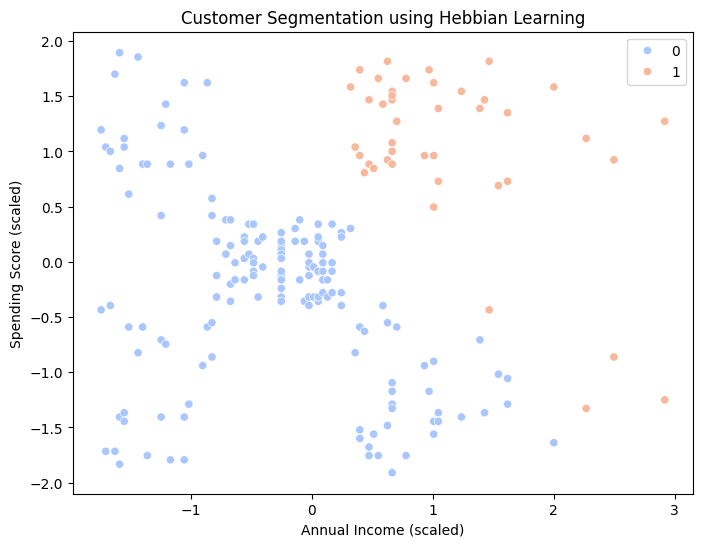
plt.title("Customer Segmentation using Hebbian Learning")

plt.xlabel("Annual Income (scaled)")

plt.ylabel("Spending Score (scaled)")

plt.show()

**OUTPUT:**



NAME:PAREENITA A.SHIRSATH PRN:221101062 ROLL.NO:57 T.E.A.I.&.D.S.

**ML EXPERIMENT NO : 7**

**CODE:**

from google.colab import files

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

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

from sklearn.mixture import GaussianMixture

# Upload the dataset

uploaded = files.upload()

# Load the dataset

df = pd.read\_csv("Mall\_Customers.csv")

# Display first few rows

display(df.head())

# Selecting relevant features

X = df[['Annual Income (k$)', 'Spending Score (1-100)']]

# Standardizing the features

scaler = StandardScaler()

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

# Applying Expectation-Maximization using Gaussian Mixture Model

gmm = GaussianMixture(n\_components=2, random\_state=42)

gmm.fit(X\_scaled)

y\_pred = gmm.predict(X\_scaled)

# Splitting data into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y\_pred, test\_size=0.2, random\_state=42)

# Evaluation

accuracy = np.mean(gmm.predict(X\_test) == y\_test)

print("Accuracy:", accuracy)

# Visualization

plt.figure(figsize=(8,6))

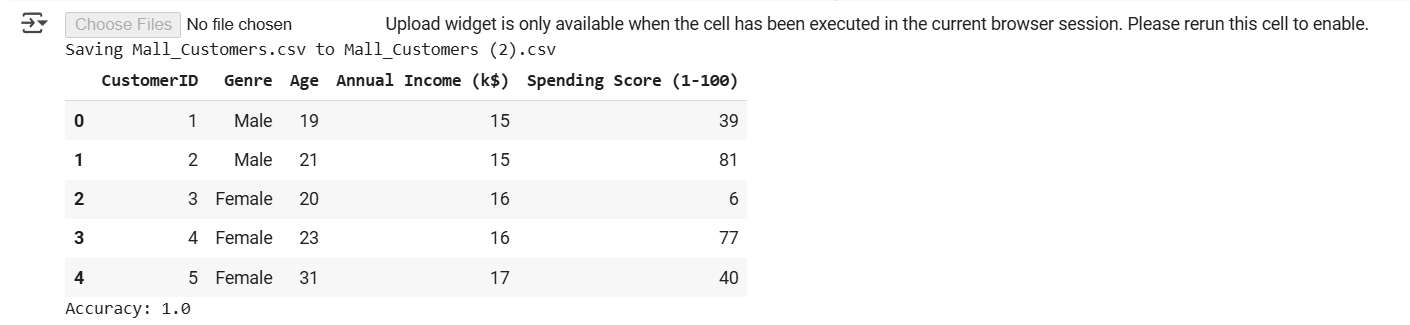
sns.scatterplot(x=X\_scaled[:, 0], y=X\_scaled[:, 1], hue=y\_pred, palette='coolwarm')

plt.title("Customer Segmentation using Expectation-Maximization")

plt.xlabel("Annual Income (scaled)")

plt.ylabel("Spending Score (scaled)")

plt.show()

**OUTPUT:** ****



NAME:PAREENITA A.SHIRSATH PRN:221101062 ROLL.NO:57 T.E.A.I.&.D.S.

**ML EXPERIMENT NO : 8**

**CODE:**

from google.colab import files

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Upload the dataset

uploaded = files.upload()

# Load the dataset

df = pd.read\_csv("Mall\_Customers.csv")

# Display first few rows

display(df.head())

# Selecting relevant features

X = df[['Annual Income (k$)', 'Spending Score (1-100)']]

# McCulloch-Pitts Model Implementation

def mcp\_neuron(inputs, weights, threshold):

    weighted\_sum = np.dot(inputs, weights)

    return 1 if weighted\_sum >= threshold else 0

# Define binary input dataset

X\_binary = np.where(X > X.median(), 1, 0)

weights = np.array([1, 1])  # Example weights

threshold = 1  # Example threshold

# Applying McCulloch-Pitts neuron

y\_pred = np.array([mcp\_neuron(row, weights, threshold) for row in X\_binary])

# Removed .values as it's not needed for NumPy arrays

# Display output

print("Predicted Binary Outputs:", y\_pred)

# Visualization

plt.figure(figsize=(8,6))

plt.scatter(X['Annual Income (k$)'], X['Spending Score (1-100)'], c=y\_pred, cmap='coolwarm')

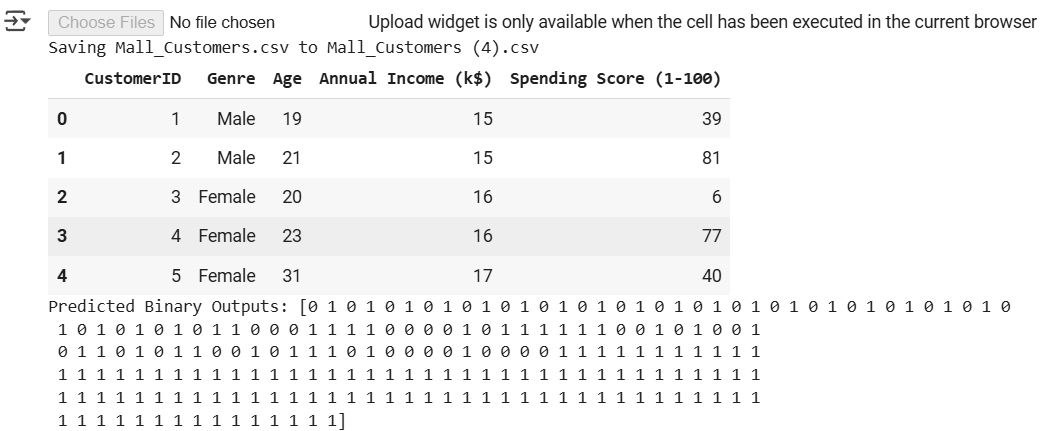
plt.title("Customer Segmentation using McCulloch-Pitts Model")

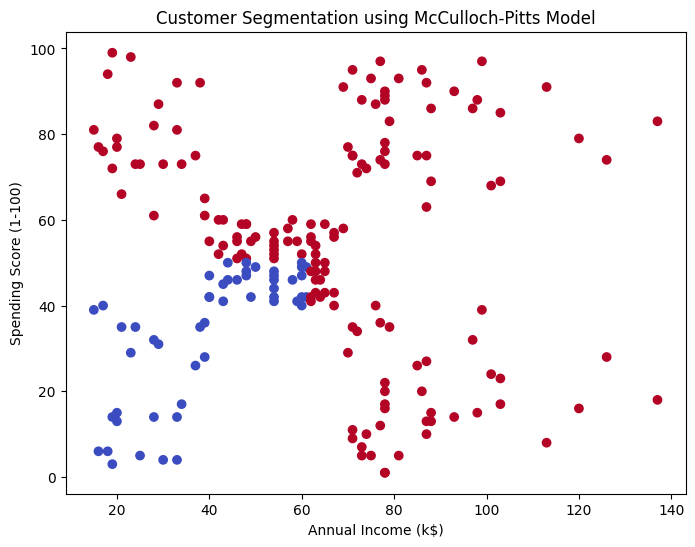
plt.xlabel("Annual Income (k$)")

plt.ylabel("Spending Score (1-100)")

plt.show()

**OUTPUT:**

****



NAME:PAREENITA A.SHIRSATH PRN:221101062 ROLL.NO:57 T.E.A.I.&.D.S.

**ML EXPERIMENT NO : 9**

**CODE:**

from google.colab import files

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Upload the dataset

uploaded = files.upload()

# Load the dataset

df = pd.read\_csv("Mall\_Customers.csv")

# Display first few rows

display(df.head())

# Selecting relevant features

X = df[['Annual Income (k$)', 'Spending Score (1-100)']]

y = (X['Spending Score (1-100)'] > X['Spending Score (1-100)'].median()).astype(int)  # Binary target

# Standardizing the features

X = (X - X.mean()) / X.std()

# Single Layer Perceptron Implementation

def perceptron\_train(X, y, learning\_rate=0.01, epochs=100):

    weights = np.zeros(X.shape[1])

    bias = 0

    for \_ in range(epochs):

        for i in range(len(y)):

            prediction = np.dot(X.iloc[i], weights) + bias

            y\_pred = 1 if prediction >= 0 else 0

            error = y[i] - y\_pred

            weights += learning\_rate \* error \* X.iloc[i]

            bias += learning\_rate \* error

    return weights, bias

# Training perceptron

weights, bias = perceptron\_train(X, y)

# Prediction

def perceptron\_predict(X, weights, bias):

    return np.where(np.dot(X, weights) + bias >= 0, 1, 0)

y\_pred = perceptron\_predict(X, weights, bias)

# Display output

print("Predicted Binary Outputs:", y\_pred)

# Visualization

plt.figure(figsize=(8,6))

plt.scatter(X['Annual Income (k$)'], X['Spending Score (1-100)'], c=y\_pred, cmap='coolwarm')

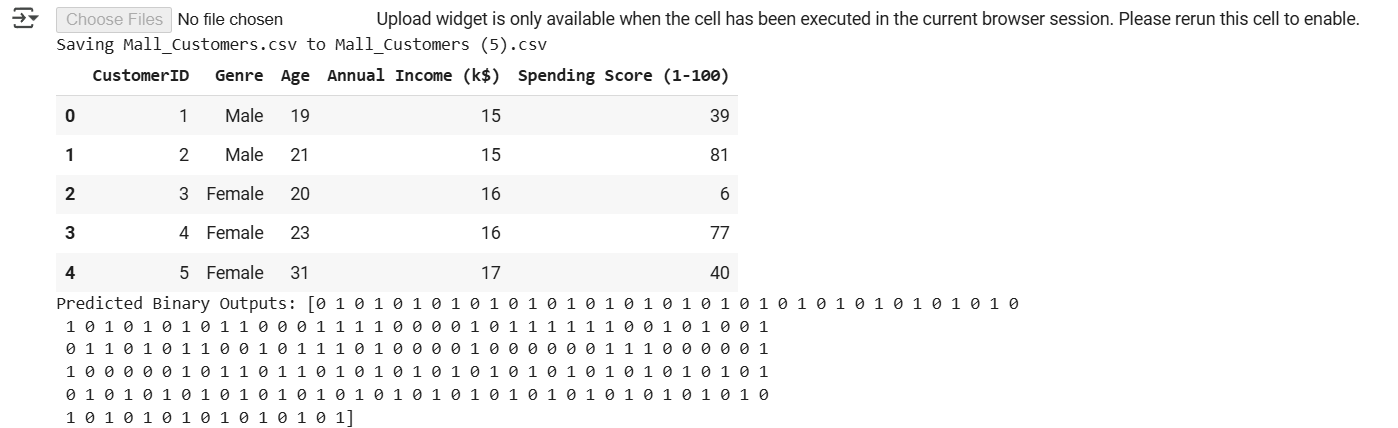
plt.title("Customer Segmentation using Single Layer Perceptron")

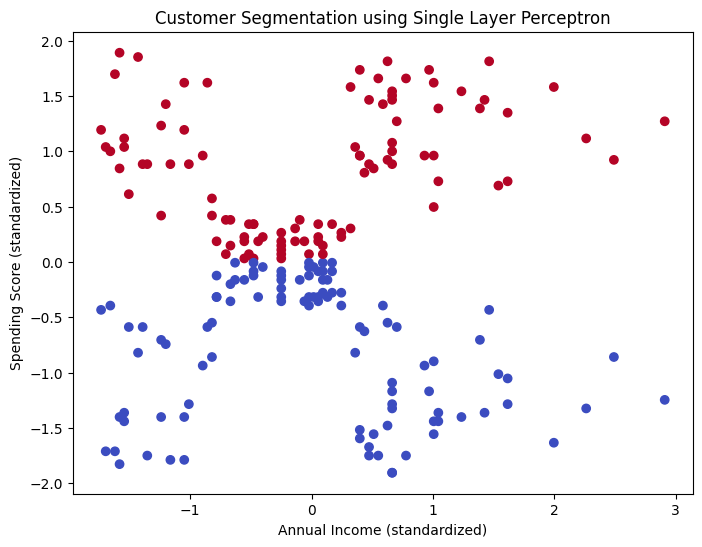
plt.xlabel("Annual Income (standardized)")

plt.ylabel("Spending Score (standardized)")

plt.show()

**OUTPUT:**





NAME:PAREENITA A.SHIRSATH PRN:221101062 ROLL.NO:57 T.E.A.I.&.D.S.

**ML EXPERIMENT NO : 10**

**CODE:**

from google.colab import files

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

# Upload the dataset

uploaded = files.upload()

# Load the dataset

df = pd.read\_csv("Mall\_Customers.csv")

# Display first few rows

display(df.head())

# Selecting relevant features

X = df[['Annual Income (k$)', 'Spending Score (1-100)']]

# Standardizing the features

scaler = StandardScaler()

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

# Applying PCA

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X\_scaled)

# Explained variance ratio

print("Explained Variance Ratio:", pca.explained\_variance\_ratio\_)

# Visualization

plt.figure(figsize=(8,6))

plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c='blue', alpha=0.5)

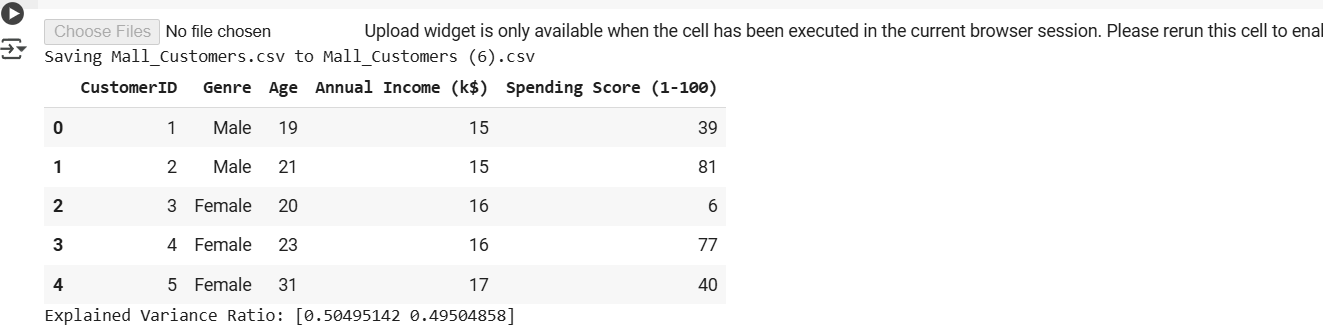
plt.title("Customer Segmentation using PCA")

plt.xlabel("Principal Component 1")

plt.ylabel("Principal Component 2")

plt.show()

**OUTPUT:**

****

