1. **Design a single unit perceptron for classification of a linearly separable binary dataset without using pre-defined models. Use the Perceptron() from sklearn.**

# Import necessary libraries

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import Perceptron

from sklearn.datasets import make\_classification

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

from sklearn.metrics import accuracy\_score

# Generate a synthetic linearly separable binary dataset

X, y = make\_classification(n\_samples=100, n\_features=2, n\_classes=2, n\_clusters\_per\_class=1, n\_redundant=0, random\_state=42)

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Visualize the dataset

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

plt.scatter(X\_train[:, 0], X\_train[:, 1], c=y\_train, cmap='coolwarm', edgecolor='k', s=50)

plt.title("Linearly Separable Binary Dataset")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.show()

# Create and train the perceptron model

perceptron = Perceptron(max\_iter=1000, tol=1e-3, random\_state=42)

perceptron.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = perceptron.predict(X\_test)

# Calculate accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy on the test set: {accuracy \* 100:.2f}%")

# Visualize the decision boundary

x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, 0.01), np.arange(y\_min, y\_max, 0.01))

# Predict on the grid

Z = perceptron.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

# Plot the decision boundary

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

plt.contourf(xx, yy, Z, alpha=0.8, cmap='coolwarm')

plt.scatter(X\_train[:, 0], X\_train[:, 1], c=y\_train, edgecolor='k', cmap='coolwarm', s=50, label="Train Data")

plt.scatter(X\_test[:, 0], X\_test[:, 1], c=y\_test, edgecolor='k', cmap='coolwarm', s=100, marker='\*', label="Test Data")

plt.title("Perceptron Decision Boundary")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.legend()

plt.show()

1. **Identify the problem with single unit Perceptron. Classify using Or-, And- and Xor-ed data and analyze the result.**

# Import necessary libraries

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import Perceptron

from sklearn.metrics import accuracy\_score

# Define OR, AND, and XOR datasets

or\_data = {

"X": np.array([[0, 0], [0, 1], [1, 0], [1, 1]]),

"y": np.array([0, 1, 1, 1])

}

and\_data = {

"X": np.array([[0, 0], [0, 1], [1, 0], [1, 1]]),

"y": np.array([0, 0, 0, 1])

}

xor\_data = {

"X": np.array([[0, 0], [0, 1], [1, 0], [1, 1]]),

"y": np.array([0, 1, 1, 0])

}

# Function to train and visualize perceptron performance

def analyze\_perceptron(data, title):

X, y = data["X"], data["y"]

# Train perceptron

perceptron = Perceptron(max\_iter=1000, tol=1e-3, random\_state=42)

perceptron.fit(X, y)

# Predict and calculate accuracy

y\_pred = perceptron.predict(X)

accuracy = accuracy\_score(y, y\_pred)

print(f"{title} Classification Accuracy: {accuracy \* 100:.2f}%")

# Visualize decision boundary

x\_min, x\_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5

y\_min, y\_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, 0.01), np.arange(y\_min, y\_max, 0.01))

Z = perceptron.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

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

plt.contourf(xx, yy, Z, alpha=0.8, cmap='coolwarm')

plt.scatter(X[:, 0], X[:, 1], c=y, edgecolor='k', cmap='coolwarm', s=100)

plt.title(f"{title} with Perceptron")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.show()

# Analyze OR, AND, and XOR datasets

analyze\_perceptron(or\_data, "OR Gate")

analyze\_perceptron(and\_data, "AND Gate")

analyze\_perceptron(xor\_data, "XOR Gate")

# Discussion:

# The perceptron performs well on OR and AND datasets because they are linearly separable.

# However, it fails on the XOR dataset because XOR is not linearly separable.