Name – Subham Beura

Branch – CE

ID – B521060

ML Lab

Write a program to implement perceptron for different learning tasks.

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
import pandas as pd


data = pd.read_csv('lab1_dataset.csv')
X = data.drop('target', axis=1).values
y = data['target'].values

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)


scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

class BinaryPerceptron:
 def __init__(self, learning_rate=0.01, n_iters=1000):
 self.lr = learning_rate
 self.n_iters = n_iters
 self.activation_func = self._unit_step_func
 self.weights = None
 self.bias = None

 def fit(self, X, y):
 n_samples, n_features = X.shape
 self.weights = np.zeros(n_features)
 self.bias = 0

 y_ = np.array([1 if i > 0 else 0 for i in y])

 for _ in range(self.n_iters):
 for idx, x_i in enumerate(X):
 predicted = self.predict(x_i)
 update = self.lr * (y_[idx] - predicted)
 self.weights += update * x_i
```

```
 self.bias += update

 def predict(self, X):
 linear_model = np.dot(X, self.weights) + self.bias
        y_predicted = self.activation_func(linear_model)
        return y_predicted

    def _unit_step_func(self, x):
        return np.where(x>=0, 1, 0)

perceptron = BinaryPerceptron(learning_rate=0.001, n_iters=1500)
perceptron.fit(X_train, y_train)

y_pred = perceptron.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
```

0.8524590163934426