BARI ANKIT (56)

Exp – 7 : Perceptron

Code :

```
import numpy as np

from sklearn.model_selection import train_test_split

from sklearn import datasets


def unit_step_func(x):

  return np.where(x > 0 , 1, 0)


class Perceptron:


  def __init__(self, learning_rate=0.01, n_iters=1000):

    self.lr = learning_rate

    self.n_iters = n_iters

    self.activation_func = 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.where(y > 0 , 1, 0)

        for _ in range(self.n_iters):
            for idx, x_i in enumerate(X):
                linear_output = np.dot(x_i, self.weights) + self.bias
                y_predicted = self.activation_func(linear_output)

                update = self.lr * (y_[idx] - y_predicted)
                self.weights += update * x_i
                self.bias += update


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


if __name__ == "__main__":

    def accuracy(y_true, y_pred):
        accuracy = np.sum(y_true == y_pred) / len(y_true)
        return accuracy
```

```
 X, y = datasets.make_blobs(
  n_samples=150, n_features=2, centers=2, cluster_std=1.05, random_state=2
 )
 X_train, X_test, y_train, y_test = train_test_split(
  X, y, test_size=0.2, random_state=123
 )

 p = Perceptron(learning_rate=0.01, n_iters=1000)
 p.fit(X_train, y_train)
 predictions = p.predict(X_test)

 print("Perceptron classification accuracy :", accuracy(y_test, predictions))
```

Output :

```
Perceptron classification accuracy : 1.0
```