Rayan Forsat

610398155

ASsignment 1

Problem 1

1. T: تشخیص نژاد اسب

   E: عکس اسب ها با لیبل نژاد آنها

   P: درصد پاسخگویی درست نژاد اسب توسط مدل

2. T: دسته بندي بازديد كنندگان سايت براي معرفي محصولات متناسب تر به آن ها

   E: داده هایی از خرید قبلی مشتریان

   P: تشابه خریدهای مشتریانی که در یک دسته قرار گرفته شده اند یا میزان استقبال آنها از محصول پیشنهادی

3. T: شناسایی خواست هاي مشكوك

   E: داده هایی از درخواست هایی از کاربران عادی

   P: درصد شناسایی درخواست های غیرعادی

4. T:تشخیص ترند تغییرات قیمت محصولات در بورس

   E:داده هایی از ترند های قبلی و ویژگی های محیطی آن زمان

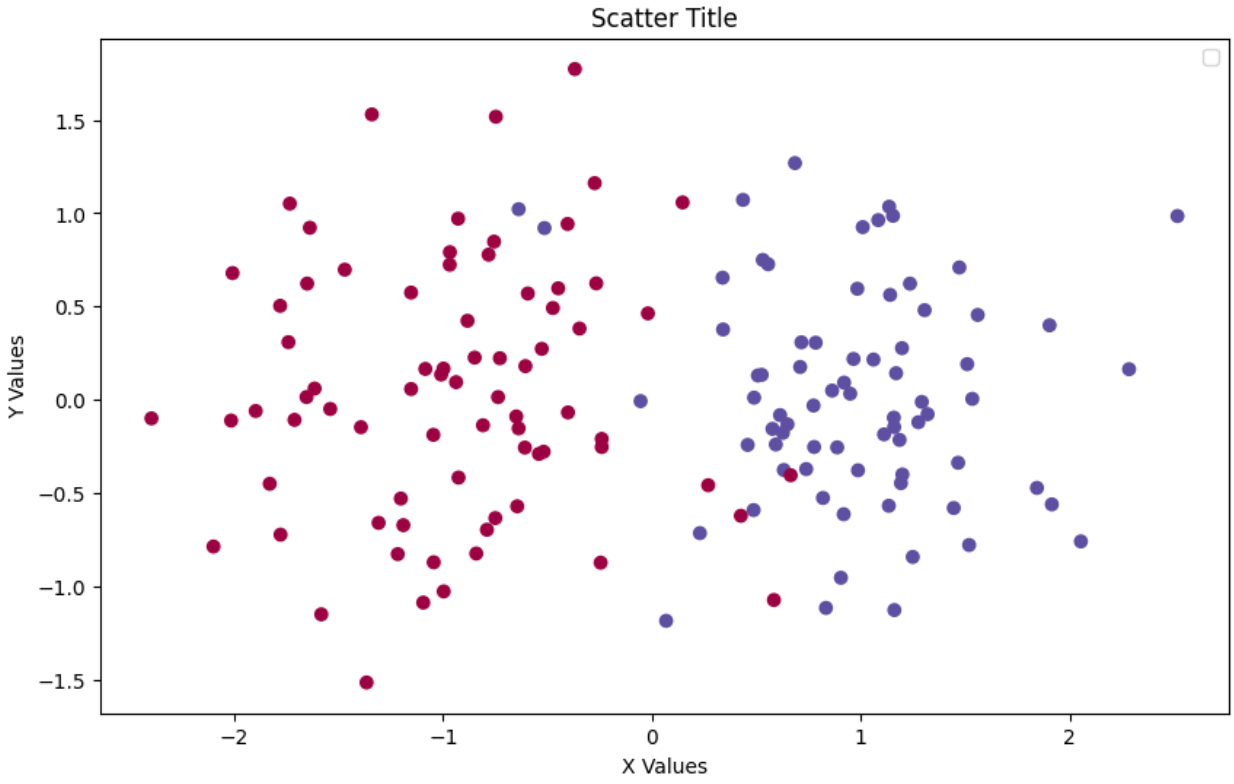
   P: میزان درستی حدس ترند ها

1. Supervised Learning
2. Unsupervised Learning
3. Supervised Learning
4. Supervised Learning

Supervised Learning → 1. Classification 2.Regression

Problem 2

1.



2. خیر زیرا داده ها به صورت خطی جدا نمیشوند

3.

```
class Perceptron:
    def __init__(self, learning_rate=0.01, n_iters=1000):
        self.learning_rate = learning_rate
        self.n_iters = n_iters
        self.weights = None
        self.bias = None
        Self.pocket = 0
        Self.pocket_weights = None
        Self.pocket_bias = None

    def fit(self, X, y):
        n_samples = X.shape[0]
        n_features = X.shape[1]

        self.weights = np.zeros(n_features)
        self.bias = 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)

                if y_predicted == y[idx]:
                    update = 0
                elif y_predicted == 0 and y[idx] == 1:
                    update = self.learning_rate * x_i
                elif y_predicted == 1 and y[idx] == 0:
                    update = -self.learning_rate * x_i

                self.weights += update
            newacc = np.sum(self.predict(X) == y) / len(y)
            if newacc > self.pocket:
                self.pocket = newacc
                self.pocket_weights = self.weights
                self.pocket_bias = self.bias
```

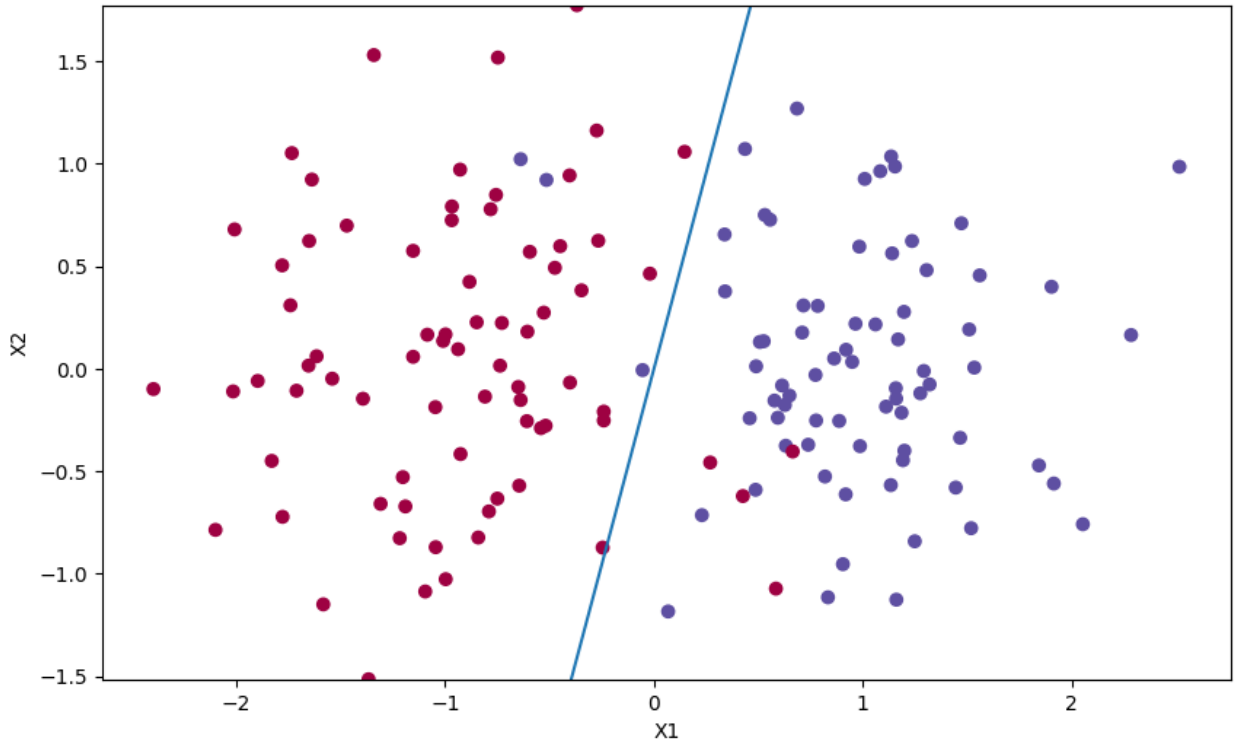
4

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

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

4.

```
Train Accuracy: 0.9533333333333334
```



```
Test Accuracy: 0.98
```