22p-9295-amber

April 19, 2024

[94]:
```
from sklearn.datasets import load_iris

iris = load_iris()
X = iris.data
y = iris.target
```

[95]:
```
import pandas as pd

column_names = ["sepal_length", "sepal_width", "petal_length", "petal_width"]
iris_df = pd.DataFrame(X, columns=column_names)
iris_df["target"] = y
```

[96]:
```
binary_df = iris_df[iris_df["target"] != 2]
```

[97]:
```
X_binary = binary_df.drop("target", axis=1)
y_binary = binary_df["target"]
```

[98]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_binary, y_binary,␣
 ↪test_size=0.2, random_state=42)
```

[99]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_binary, y_binary,␣
 ↪test_size=0.2, random_state=42)
```

[100]:
```
from sklearn.linear_model import Perceptron

perceptron_model = Perceptron()
perceptron_model.fit(X_train, y_train)
```

[100]:
```
Perceptron()
```

[101]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score
```

```python
y_pred = perceptron_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```

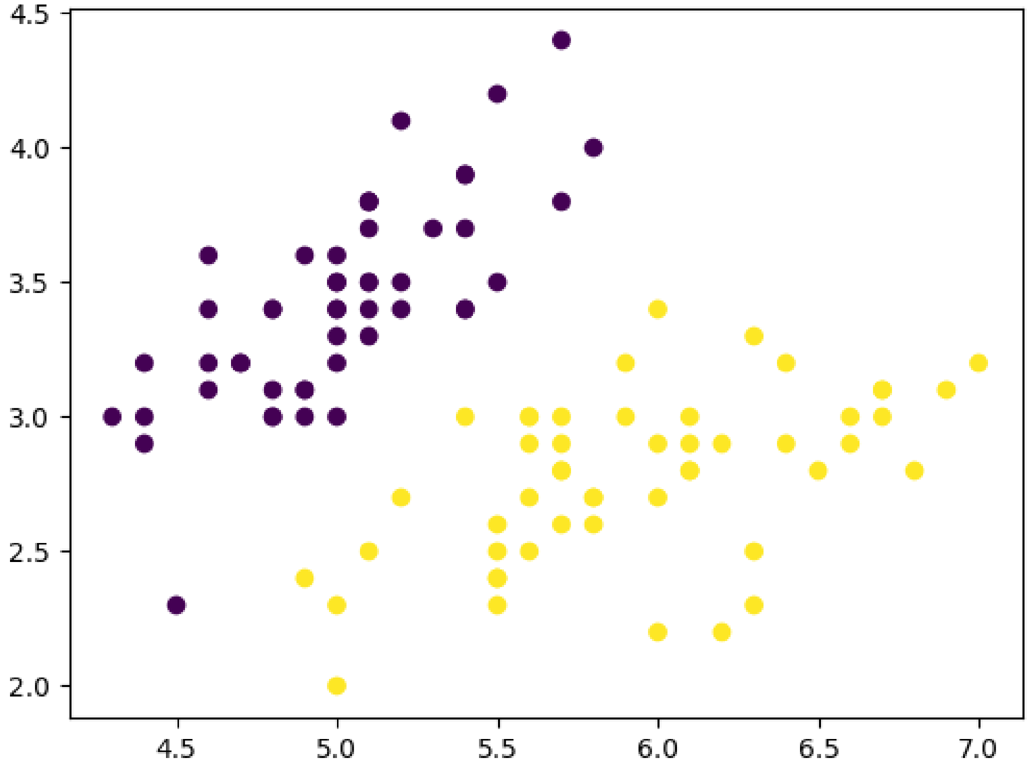
[102]:

```python
def train_weights(train, l_rate, n_epoch):

    weights = [0.0 for i in range(len(train[0]))]

    for epoch in range(n_epoch):

        sum_error = 0.0

        for row in train:

            prediction = predict(row, weights)

            error = row[-1] - prediction

            sum_error += error**2

            weights[0] = weights[0] + l_rate * error

            for i in range(len(row)-1):

                weights[i + 1] = weights[i + 1] + l_rate * error * row[i]



    return weights

def predict(row, weights):
    activation = weights[0]
    for i in range(len(row)-1):
        activation += weights[i + 1] * row[i]
    return 1.0 if activation >= 0.0 else 0.0

l_rate = 0.01
n_epoch = 1000

weights = train_weights(X_train.values, l_rate, n_epoch)

y_pred = [predict(row, weights) for row in X_test.values]

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Scratch Implementation")
print("Verginica and Versicolor")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Scratch Implementation
Verginica and Versicolor
Accuracy: 0.6
Precision: 0.5
Recall: 1.0
F1 Score: 0.6666666666666666
```

[104]:

```
# Visualize the data using a scatter plot
import matplotlib.pyplot as plt

plt.scatter(data['sepal length (cm)'], data['sepal width (cm)'],␣
 ↪c=data['target'])
plt.show()
```


4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0