Muhammad Hassan Shah
20P-0025
BSCS-6C
AI-Lab

Lab 05

MuhammadHassanShah_20 | MuhammadHassanShah_20

Markdown | Python 3 (ipykernel)

Load the iris dataset using scikit-learn library

[20]:
```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score


# Load iris dataset
data = load_iris()
```

Create a Pandas DataFrame with the dataset and add column names

[21]:
```
df = pd.DataFrame(data=data.data, columns=data.feature_names)
df['target'] = pd.DataFrame(data = data.target)
df
```

[21]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0

MuhammadHassanShah_20 × MuhammadHassanShah_20 ×

Markdown Python 3 (ipykernel)

Convert the problem into a binary classification problem by only considering two classes and removing the third one. For example, we can keep only "setosa" and "versicolor" classes and remove "virginica". Visualize the data using a scatter plot.

[22]:
```
df['target'] = df['target'].replace(2,1)
df
```

[22]:

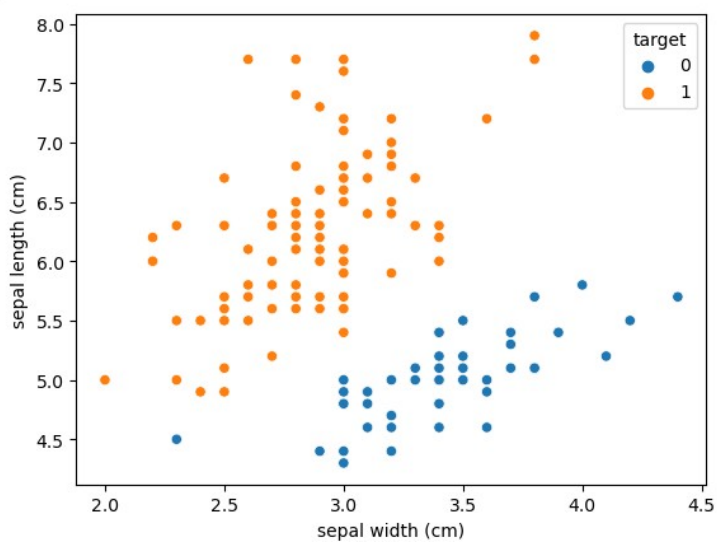
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	1
146	6.3	2.5	5.0	1.9	1
147	6.5	3.0	5.2	2.0	1
148	6.2	3.4	5.4	2.3	1
149	5.9	3.0	5.1	1.8	1

MuhammadHassanShah_20 × MuhammadHassanShah_20 ×

Markdown Python 3 (ipykernel)

150 rows × 5 columns

[23]:
```
import matplotlib.pyplot as plt
sns.scatterplot(data=df, x='sepal width (cm)', y='sepal length (cm)', hue='target')
plt.show()
```



MuhammadHassanShah_20 × MuhammadHassanShah_20 ×

Markdown Python 3 (ipykernel)

Split the data into train and test sets

[31]:
```
X = df[["sepal length (cm)" , "sepal width (cm)","petal length (cm)","petal width (cm)"]]
y = pd.DataFrame(data = df["target"])
X_train, X_test, y_train ,y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

[25]:
```
X_train
```

[25]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
60	5.0	2.0	3.5	1.0
116	6.5	3.0	5.5	1.8
144	6.7	3.3	5.7	2.5
119	6.0	2.2	5.0	1.5
108	6.7	2.5	5.8	1.8
...	...	...	...	...
9	4.9	3.1	1.5	0.1
103	6.3	2.9	5.6	1.8
67	5.8	2.7	4.1	1.0
117	7.7	3.8	6.7	2.2
47	4.6	3.2	1.4	0.2

105 rows × 4 columns

MuhammadHassanShah_20 × MuhammadHassanShah_20 ×

Markdown Python 3 (ipykernel)

Remove the target column from the train and test sets

[32]:
```
y_train = y_train.to_numpy()
y_train.shape
```

[32]:
```
(105, 1)
```

[33]:
```
y_train = y_train.reshape(105,)
```

Apply the built-in Perceptron algorithm from scikit-learn

[34]:
```
perceptron = Perceptron()
perceptron.fit(X_train, y_train)
```

[34]:
```
Perceptron()
```

Evaluate the accuracy, precision, recall, and F1 score of the model.

[35]:
```
y_pred = perceptron.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)
```

MuhammadHassanShah_20 × MuhammadHassanShah_20 ×

Markdown Python 3 (ipykernel)

Evaluate the accuracy, precision, recall, and F1 score of the model.

[35]:
```
y_pred = perceptron.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
```

Apply the Perceptron algorithm from scratch using above code snippets

[36]:
```
def train_weights(train, l_rate, n_epoch):

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

    for epoch in range(n_epoch):
```

MuhammadHassanShah_20 × MuhammadHassanShah_20 ×

Markdown Python 3 (ipykernel)

Apply the Perceptron algorithm from scratch using above code snippets

[36]:
```
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  #bias(t+1) = bias(t) + learning_rate * (expected(t) - predicted

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

                weights[i + 1] = weights[i + 1] + l_rate * error * row[i] #w(t+1) = w(t) + learning_rate * (expected(

        print('epoch=%d, lrate=%.3f, error=%.3f' % (epoch, l_rate, sum_error))

    return weights
```

MuhammadHassanShah_20 × MuhammadHassanShah_20 ×

Markdown Python 3 (ipykernel)

[37]:
```python
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
```

[39]:
```python
l_rate = 0.01
n_epoch = 10

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)
```

```
epoch=0, lrate=0.010, error=68.730
epoch=1, lrate=0.010, error=68.730
epoch=2, lrate=0.010, error=68.730
epoch=3, lrate=0.010, error=68.730
epoch=4, lrate=0.010, error=68.730
epoch=5, lrate=0.010, error=68.730
epoch=6, lrate=0.010, error=68.730
epoch=7, lrate=0.010, error=68.730
epoch=8, lrate=0.010, error=68.730
epoch=9, lrate=0.010, error=68.730
```

Evaluate the accuracy, precision, recall, and F1 score of the model.

MuhammadHassanShah_20 × MuhammadHassanShah_20 ×

Markdown Python 3 (ipykernel)

epoch=2, lrate=0.010, error=68.730
epoch=3, lrate=0.010, error=68.730
epoch=4, lrate=0.010, error=68.730
epoch=5, lrate=0.010, error=68.730
epoch=6, lrate=0.010, error=68.730
epoch=7, lrate=0.010, error=68.730
epoch=8, lrate=0.010, error=68.730
epoch=9, lrate=0.010, error=68.730

Evaluate the accuracy, precision, recall, and F1 score of the model.

[40]:
```python
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 score:', f1)
```

```
Accuracy: 0.6444444444444445
Precision: 0.6444444444444445
Recall: 1.0
F1 score: 0.7837837837837839
```

[ ]: