code

August 17, 2023

[12]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score
```

[13]:
```
# Load the data from the Excel file
data = pd.read_excel("~/dataset/logistic-regression/Pumpkin_Seeds_Dataset.xlsx")

# Preprocess the data
X = data.drop(columns=['Class'])
y = (data['Class'] == 'Çerçevelik').astype(int)  # Convert 'Çerçevelik' to 1,␣
 ↪else 0

# Split the data into training, validation, and testing sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.5,␣
 ↪random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.6,␣
 ↪random_state=42)

# Normalize/standardize the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)
```

[14]:
```
# Implement Logistic Regression
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

def predict(X, weights):
    z = np.dot(X, weights)
    return sigmoid(z)

def logistic_regression(X, y, learning_rate, num_iterations):
    num_samples, num_features = X.shape
    weights = np.zeros(num_features)
```

```python
    for _ in range(num_iterations):
        y_pred = predict(X, weights)
        gradient = np.dot(X.T, (y_pred - y)) / num_samples
        weights -= learning_rate * gradient

    return weights
```

[15]:
```python
# Calculate accuracy, precision, and recall
def calculate_metrics(y_true, y_pred):
    y_pred_binary = (y_pred >= 0.5).astype(int)
    accuracy = accuracy_score(y_true, y_pred_binary)
    precision = precision_score(y_true, y_pred_binary, zero_division=1)
    recall = recall_score(y_true, y_pred_binary, zero_division=1)
    return accuracy, precision, recall
```

[16]:
```python
def logistic_regression_calc(learning_rate,num_iterations):
    weights = logistic_regression(X_train_scaled, y_train, learning_rate,␣
 ↪num_iterations)

    # Make predictions on test set
    test_predictions = sigmoid(np.dot(X_test_scaled, weights))

    test_accuracy, test_precision, test_recall = calculate_metrics(y_test,␣
 ↪test_predictions)
    print(f"Test Data evaluation(Learning rate = {learning_rate} and Number of␣
 ↪iterations = {num_iterations}):")
    print("Accuracy:", test_accuracy)
    print("Precision:", test_precision)
    print("Recall:", test_recall)
```

[17]:
```python
logistic_regression_calc(0.01,100)
```

```
Test Data evaluation(Learning rate = 0.01 and Number of iterations = 100):
Accuracy: 0.8573333333333333
Precision: 0.8601583113456465
Recall: 0.8578947368421053
```

[18]:
```python
logistic_regression_calc(0.001,100)
```

```
Test Data evaluation(Learning rate = 0.001 and Number of iterations = 100):
Accuracy: 0.8453333333333334
Precision: 0.8473684210526315
Recall: 0.8473684210526315
```

[19]:
```python
logistic_regression_calc(0.0001,100)
```

```
Test Data evaluation(Learning rate = 0.0001 and Number of iterations = 100):
Accuracy: 0.844
```

```
Precision: 0.8469656992084432
Recall: 0.8447368421052631
```

[20]:
```python
logistic_regression_calc(0.01,200)
```

```
Test Data evaluation(Learning rate = 0.01 and Number of iterations = 200):
Accuracy: 0.8586666666666667
Precision: 0.8605263157894737
Recall: 0.8605263157894737
```

[21]:
```python
logistic_regression_calc(0.001,200)
```

```
Test Data evaluation(Learning rate = 0.001 and Number of iterations = 200):
Accuracy: 0.8466666666666667
Precision: 0.8496042216358839
Recall: 0.8473684210526315
```

[22]:
```python
logistic_regression_calc(0.0001,200)
```

```
Test Data evaluation(Learning rate = 0.0001 and Number of iterations = 200):
Accuracy: 0.844
Precision: 0.8469656992084432
Recall: 0.8447368421052631
```