Deployment on Flask

Name: Kirtoria Ward

Batch Code: LISUM38

Submission Date: 10/22/24

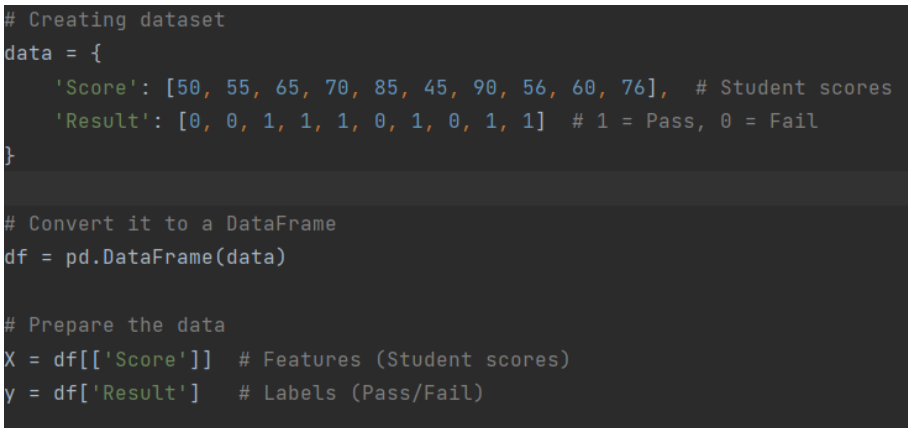
Submitted to: GitHub

Project Explanation:

This project focuses on building a **Pass/Fail Predictor** using a machine learning model and deploying it as a web application with Flask. A simple dataset of student scores and their pass/fail outcomes is used to train a **logistic regression** model, which classifies whether a student passes or fails based on their score. The trained model is saved and integrated into a Flask application, which handles predictions via POST requests. The Flask app accepts a student's score as input and returns a JSON response indicating whether the student has passed or failed. This project demonstrates the complete process of model training, deployment, and serving predictions through a web interface.

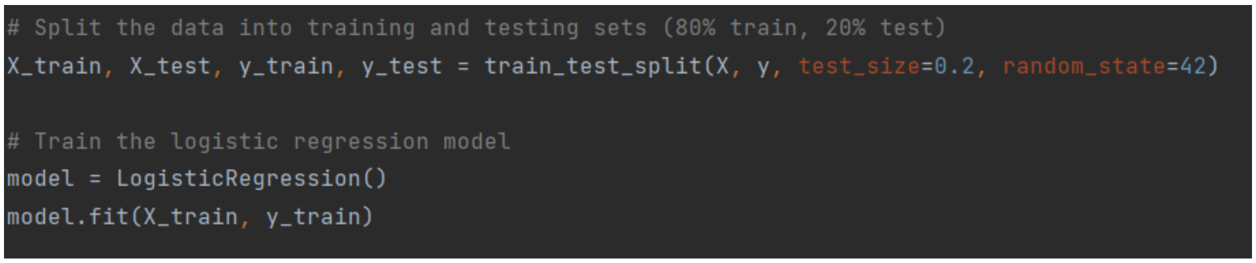
Model Setup for Pass/Fail Predictor

Loading the Data:



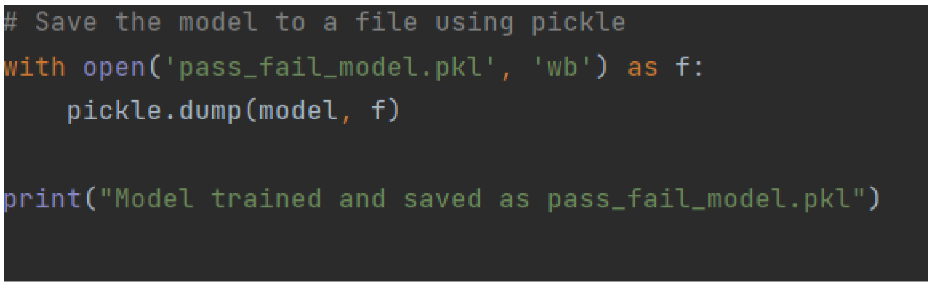
```python
# Creating dataset
data = {
    'Score': [50, 55, 65, 70, 85, 45, 90, 56, 60, 76],  # Student scores
    'Result': [0, 0, 1, 1, 1, 0, 1, 0, 1, 1]  # 1 = Pass, 0 = Fail
}

# Convert it to a DataFrame
df = pd.DataFrame(data)

# Prepare the data
X = df[['Score']]  # Features (Student scores)
y = df['Result']   # Labels (Pass/Fail)
```

Training the data:



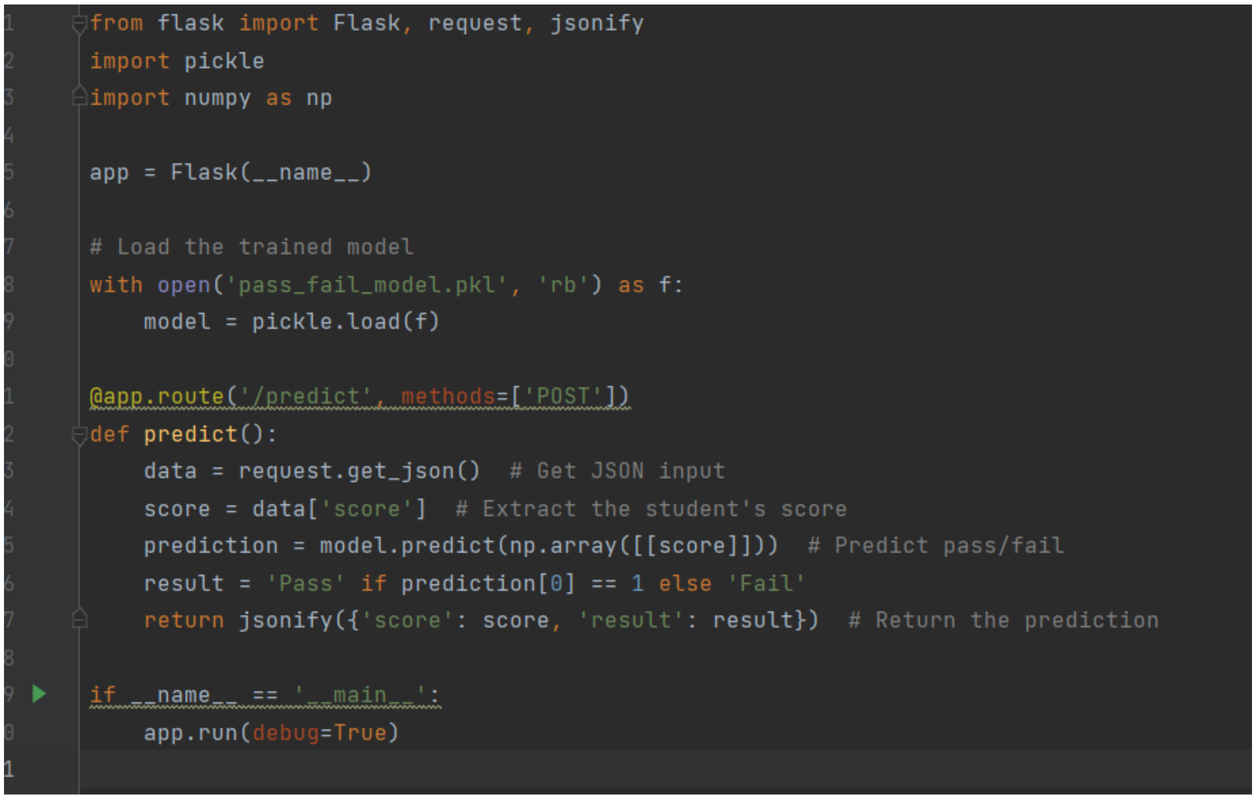
```python
# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)
```

Saving the model to a file:



```python
# Save the model to a file using pickle
with open('pass_fail_model.pkl', 'wb') as f:
    pickle.dump(model, f)

print("Model trained and saved as pass_fail_model.pkl")
```

Flask Web App Setup:

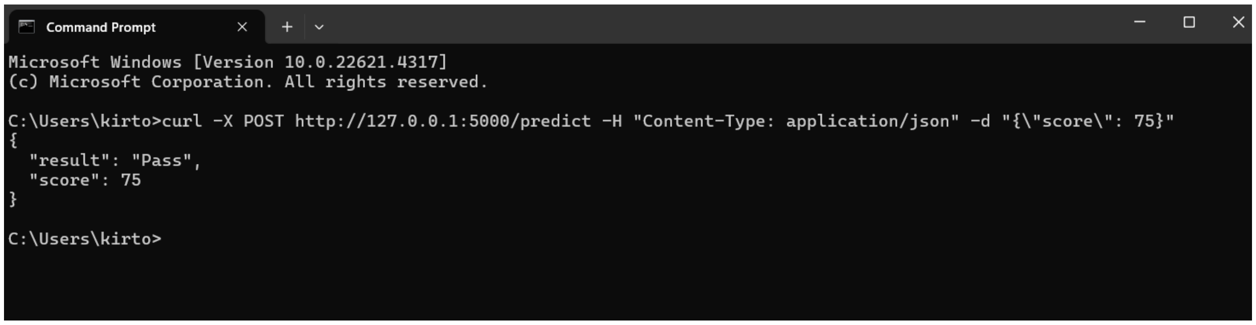


```python
from flask import Flask, request, jsonify
import pickle
import numpy as np

app = Flask(__name__)

# Load the trained model
with open('pass_fail_model.pkl', 'rb') as f:
    model = pickle.load(f)

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json()  # Get JSON input
    score = data['score']  # Extract the student's score
    prediction = model.predict(np.array([[score]]))  # Predict pass/fail
    result = 'Pass' if prediction[0] == 1 else 'Fail'
    return jsonify({'score': score, 'result': result})  # Return the prediction

if __name__ == '__main__':
    app.run(debug=True)
```

Example Prediction Response:

When you test the Flask app with a score, here is an example of what you should get

This example shows if a student has a score of 75%:



```
Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\kirto>curl -X POST http://127.0.0.1:5000/predict -H "Content-Type: application/json" -d "{\"score\": 75}"
{
  "result": "Pass",
  "score": 75
}

C:\Users\kirto>
```