

Name: Kirtoria Ward

Batch Code: LISUM38

Submission Date: 11/21/2024

Submitted to: GitHub

Introduction

The Pass/Fail Predictor project is a Flask-based machine learning application designed to predict whether a student will pass or fail based on two grades. This project involved selecting a toy dataset, training a Logistic Regression model, deploying it on Heroku, and creating an API-based and web app interface. The live app URL is <https://ward-pass-fail-c7779ab7c92c.herokuapp.com/>.

The purpose of this report is to document the steps taken to complete the deployment and showcase the functionality of the deployed app.

Steps for Deployment

1. Selecting the Toy Data

The dataset used for this project included two features, Grade1 and Grade2, and a binary label, Result. The label indicates whether a student passed (1) or failed (0). The dataset is as follows:

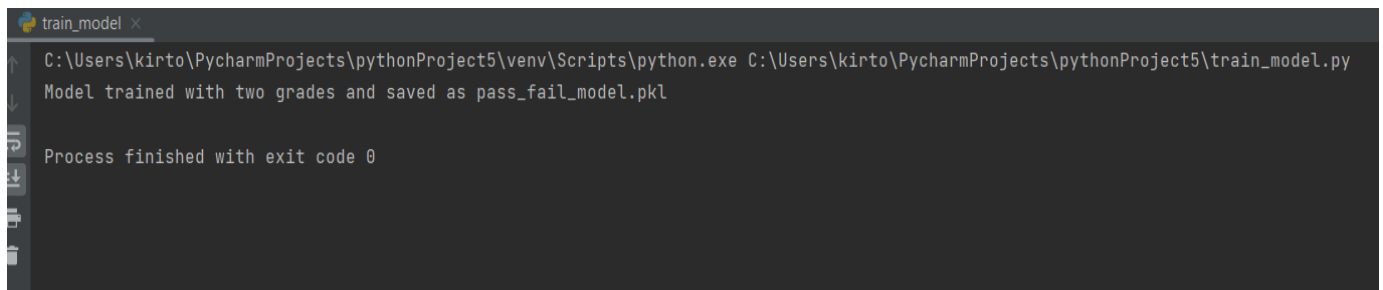
```
# Creating dataset with two grades
data = {
    'Grade1': [50, 55, 65, 70, 85, 45, 90, 56, 60, 76], # First grade
    'Grade2': [48, 60, 66, 75, 80, 50, 92, 54, 62, 78], # Second grade
    'Result': [0, 0, 1, 1, 1, 0, 1, 0, 1, 1] # 1 = Pass, 0 = Fail
}
```

The toy dataset is small and simple, making it ideal for this assignment. It represents a basic classification problem where the Grade1 and Grade2 features are used to predict the Result.

2. Training and Saving the Model

The Logistic Regression model was trained using the above dataset. The train_model.py script:

1. Prepared the data for training by separating features (Grade1 and Grade2) and the target variable (Result).
2. Split the dataset into training and testing sets (80% for training, 20% for testing).
3. Trained a Logistic Regression model to predict whether a student would pass or fail.
4. Saved the trained model as pass_fail_model.pkl using the pickle library.

A screenshot of a terminal window with a dark background. The title bar at the top says 'train_model x'. The command line shows the full path to the Python interpreter and the script: 'C:\Users\kirto\PycharmProjects\pythonProject5\venv\Scripts\python.exe C:\Users\kirto\PycharmProjects\pythonProject5\train_model.py'. The output of the script is displayed in two lines: 'Model trained with two grades and saved as pass_fail_model.pkl' and 'Process finished with exit code 0'. On the left side of the terminal, there is a vertical toolbar with icons for running, stopping, and other actions.

```
train_model x
C:\Users\kirto\PycharmProjects\pythonProject5\venv\Scripts\python.exe C:\Users\kirto\PycharmProjects\pythonProject5\train_model.py
Model trained with two grades and saved as pass_fail_model.pkl
Process finished with exit code 0
```

3. Deploying the Model on Heroku

The Flask app was developed using app.py to handle predictions via an API and a web form.

The app was prepared for deployment with the following steps:

1. Created necessary files:
 - requirements.txt: Listed dependencies like Flask, Scikit-learn, and Gunicorn.
 - Procfile: Configured Gunicorn to serve the app.
 - runtime.txt: Specified the Python version.

- templates/index.html: Defined the web interface for inputs and predictions.
2. Deployed the app on Heroku using Git. The terminal output confirmed successful deployment.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

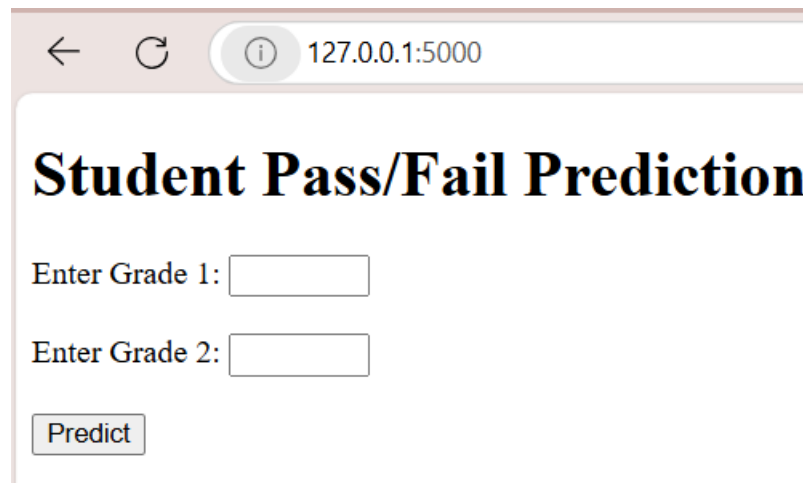
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

(venv) PS C:\Users\kirto\PycharmProjects\pythonProject5> git push heroku main
Everything up-to-date
(venv) PS C:\Users\kirto\PycharmProjects\pythonProject5> █
```

4. Testing the App

After deployment, the app was tested both locally and on Heroku:

- **Locally:** The app was accessed via **<http://127.0.0.1:5000/>** to confirm it worked before deployment.



← ↻ ⓘ 127.0.0.1:5000

Student Pass/Fail Prediction

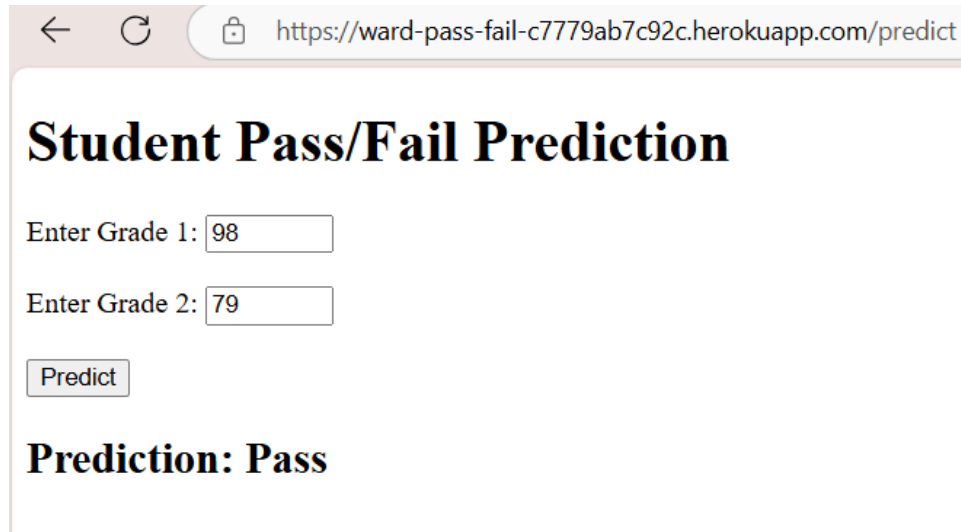
Enter Grade 1:

Enter Grade 2:

- **Live:** The app was accessed at <https://ward-pass-fail-c7779ab7c92c.herokuapp.com/>.

Predictions were tested with valid and invalid inputs. Predictions are shown below:

1. A prediction where the student has a passing grade



← ↻ 🔒 https://ward-pass-fail-c7779ab7c92c.herokuapp.com/predict

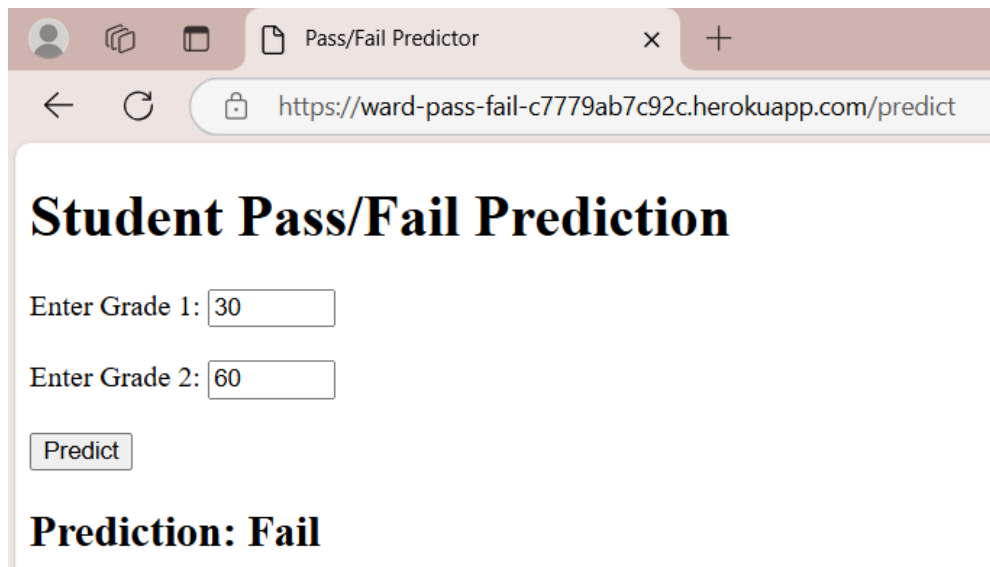
Student Pass/Fail Prediction

Enter Grade 1:

Enter Grade 2:

Prediction: Pass

2. A prediction where the student has a failing grade



👤 📁 📄 Pass/Fail Predictor × +

← ↻ 🔒 https://ward-pass-fail-c7779ab7c92c.herokuapp.com/predict

Student Pass/Fail Prediction

Enter Grade 1:

Enter Grade 2:

Prediction: Fail

3. Handling invalid inputs – Negative inputs

Pass/Fail Predictor

https://ward-pass-fail-c7779ab7c92c.herokuapp.com

Student Pass/Fail Prediction

Enter Grade 1:

! Value must be greater than or equal to 0.

Predict

4. Handling invalid inputs – Numbers over 100

Pass/Fail Predictor

https://ward-pass-fail-c7779ab7c92c.herokuappapp.com/predict

Student Pass/Fail Prediction

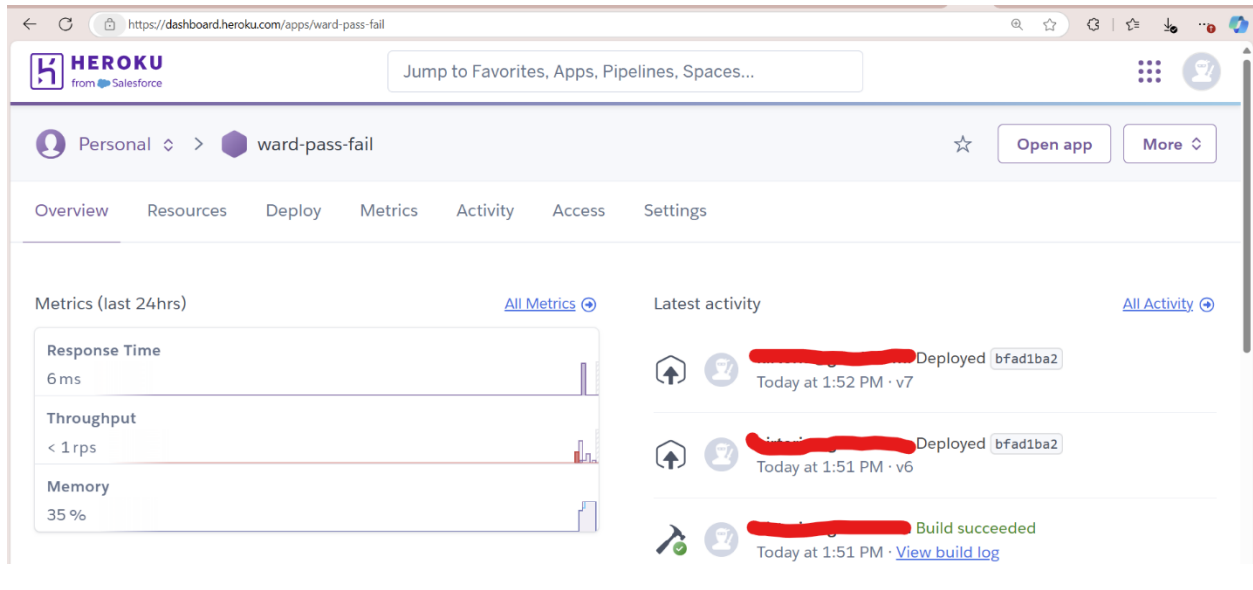
Enter Grade 1:

! Value must be less than or equal to 100.

Predict

5. Heroku Dashboard Overview

The Heroku Dashboard confirmed the app was successfully deployed and running. The app and status were verified.



Conclusion

The Pass/Fail Predictor project was successfully completed. The Logistic Regression model was trained, and the Flask app was deployed on Heroku. The app is live and accessible at <https://ward-pass-fail-c7779ab7c92c.herokuapp.com/>. The deployment demonstrated effective use of Flask, machine learning, and cloud platforms for real-world applications.