

Deployment of Student Performance Prediction Model App on Heroku

Name: Mufunwa Nemushungwa

Batch: LISUM27

Submission date: 05 December 2023

Submitted to: Heroku

Step 1: Select toy dataset.

Student_Performance

Hours Studied	Previous Scores	Sleep Hours	Sample Question Papers Practiced	Performance Index
7	99	9	1	91.0
4	82	4	2	65.0
8	51	7	2	45.0
5	52	5	2	36.0
7	75	8	5	66.0
3	78	9	6	61.0
7	73	5	6	63.0
8	45	4	6	42.0
5	77	8	2	61.0
4	89	4	0	69.0
8	91	4	5	84.0
8	79	6	2	73.0
3	47	9	2	27.0
6	47	4	2	33.0
5	79	7	8	68.0
2	72	4	3	43.0
8	73	8	4	67.0
6	83	7	2	70.0
2	54	4	9	30.0
5	75	7	0	63.0
1	99	4	3	71.0
6	96	9	0	85.0
9	74	7	6	73.0
1	85	5	6	57.0

Step 2: Training Machine Learning model

```
/Users/mufunwa/Downloads/Flask Deployment/model.py
x app.py x model.py x index.html*
1 # Importing the libraries
2 import numpy as np
3 import pandas as pd
4 import pickle
5
6 dataset = pd.read_csv('Student_Performance.csv')
7
8 X = dataset.iloc[:, :4]
9
10
11
12
13 y = dataset.iloc[:, -1]
14
15 from sklearn.linear_model import LinearRegression
16 regressor = LinearRegression()
17
18 #Fitting model with trainig data
19 regressor.fit(X, y)
20
21 # Saving model to disk
22 pickle.dump(regressor, open('model.pkl','wb'))
23
24 # Loading model to compare the results
25 model = pickle.load(open('model.pkl','rb'))
26 print(model.predict([[8, 88, 7,3]]))
```

Step 3: Creating Web App

```
/Users/mufunwa/Downloads/Flask Deployment/app.py
x app.py x model.py x index.html*
1 import numpy as np
2 from flask import Flask, request,render_template
3 import pickle
4
5 app = Flask(__name__)
6 model = pickle.load(open('model.pkl', 'rb'))
7
8 @app.route('/')
9 def home():
10     return render_template('index.html')
11
12 @app.route('/predict',methods=['POST'])
13 def predict():
14     """
15     For rendering results on HTML GUI
16     """
17     int_features = [int(x) for x in request.form.values()]
18     final_features = [np.array(int_features)]
19     prediction = model.predict(final_features)
20
21     output = round(prediction[0], 1)
22
23     return render_template('index.html', prediction_text='Student Performance should be {}'.format(output))
24
25 if __name__ == "__main__":
26     app.run(port=5000,debug=True)
```

Step 4: Committing code in GitHub.

The screenshot shows the GitHub repository page for 'Heroku-demo'. The repository is public and has 1 branch (main) and 0 tags. The file list includes: static (Create favicon.ico, 3 hours ago), templates (Update index.html, 4 hours ago), LICENSE (Initial commit, yesterday), Procfile (Add files via upload, yesterday), README.md (Initial commit, yesterday), Student_Performance.csv (Add files via upload, yesterday), app.py (Add files via upload, yesterday), model.pkl (Add files via upload, yesterday), model.py (Add files via upload, yesterday), requirements.txt (Add files via upload, yesterday), and runtime.txt (Create runtime.txt, 21 minutes ago). The README.md file is open, showing the title 'Heroku-demo'. On the right, the 'About' section describes the project as a 'Student Performance calculator app deployed on Heroku'. It also shows 'Releases' (No releases published), 'Packages' (No packages published), and 'Deployments' (5 deployments, with the latest being 'heroku-demo-app3' 19 minutes ago).

Step 5: Linking GitHub to Heroku

The screenshot shows the Heroku dashboard for the 'heroku-demo-app3' application. The 'Deployment method' section shows that the app is connected to GitHub. The 'App connected to GitHub' section indicates that code diffs, manual, and auto deploys are available for this app. The 'Automatic deploys' section shows that automatic deploys from the 'main' branch are enabled. A message states: 'You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions [here](#).' Below this, it says 'Automatic deploys from main are enabled'. A note explains: 'Every push to main will deploy a new version of this app. Deploys happen automatically; be sure that this branch in GitHub is always in a deployable state and any tests have passed before you push. [Learn more](#).' There is a checkbox for 'Wait for CI to pass before deploy' which is currently checked. At the bottom, there is a button to 'Disable Automatic Deploys'.

Step 6: Deployment of ML app on Heroku

The screenshot shows the Heroku dashboard for an application named 'heroku-demo-app3'. The 'Deploy' tab is selected, showing the 'main' branch as the source. The build output is visible, indicating a successful deployment. The build log shows the following steps:

```
-----  
----> Installing requirements with pip  
----> Discovering process types  
Procfile declares types => web  
----> Compressing...  
Done: 138.4M  
----> Launching...  
Released v9  
https://heroku-demo-app3-1829666e0c38.herokuapp.com/ deployed to Heroku  
-----
```

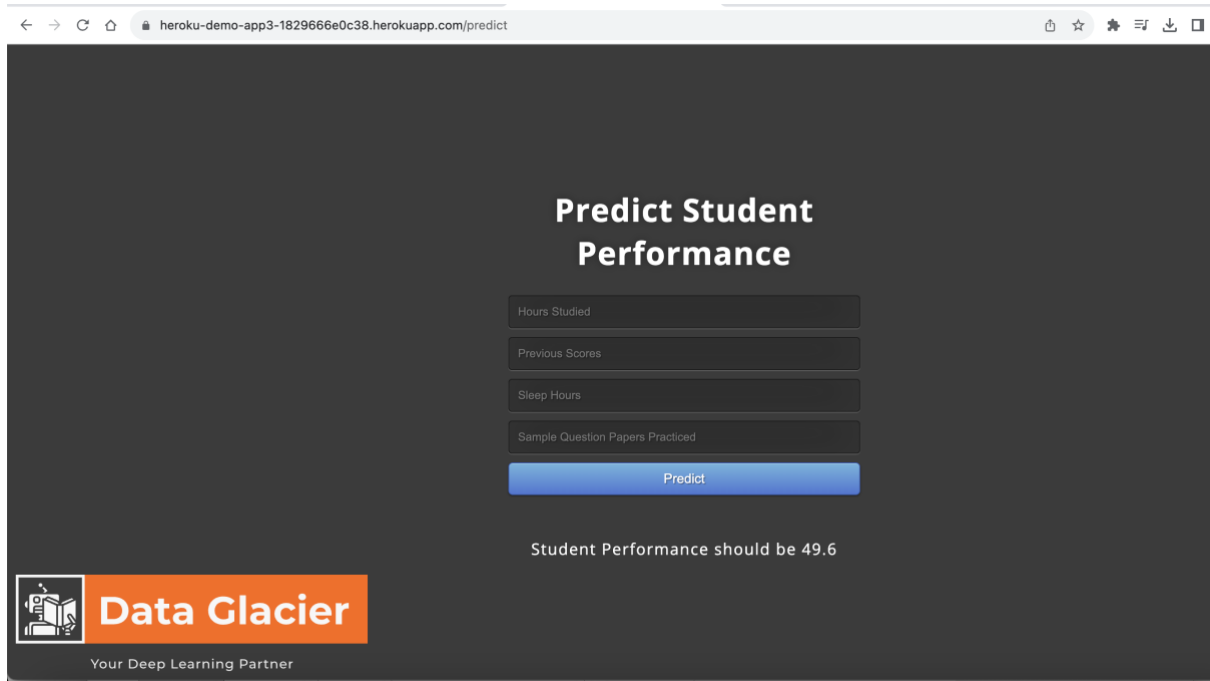
The build output also includes a checkbox for 'Autoscroll with output' and a link to 'View build log'. The 'Release phase' section is also visible, showing the 'Deploy to Heroku' button.

Step 7: Testing the web app.

The screenshot shows the web application interface for 'Predict Student Performance'. The app has a dark background with a white title and input fields. The input fields contain the following values:

- 4
- 67
- 7
- 2

A blue 'Predict' button is located below the input fields. The Data Glacier logo is visible in the bottom left corner, with the tagline 'Your Deep Learning Partner'.



App url: <https://heroku-demo-app3-1829666e0c38.herokuapp.com/predict>