Deploy a Flask App on Heroku

Name: Raahul Gomatam Vasu

Batch Code: LISUM01

Submission Date: 04/12/2023

SubmiBed To: Data Glacier

Heroku lets you deploy a Flask app online for **free**.

Heroku is a container-based cloud PlaMorm as a Service PaaS . Developers use Heroku to deploy, manage, and scale modern apps. Our plaMorm is **elegant**, **flexible**, and **easy to use**, offering developers the simplest path to geTng their apps to market.

Overview

Prerequisites

Before you can start, you need to do three things:

- Install Git
- Install Python

Sign-up Heroku account

Install Heroku

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

If you are using Mac, you can download HomeBrew from your Terminal.

Then run this command to install Heroku.

```
$ brew tap heroku/brew && brew install heroku
```

If you want to install Heroku on Windows, go to this link:
hBps://devcenter.heroku.com/arYcles/heroku-cli#other-installaYon-methods

Ge5ng Started With Heroku

Start with an empty project in your preferred text editor. In this tutorial, I am using VSCode.

Create a New Working Directory

```
$ mkdir <your-app-name>
$ cd <your-app-name>
```

Create Virtual Environment

It is recommended to create a virtual environment so that there is no conflict with Heroku during the deployment.
If you have installed Python with Anaconda, you can create your empty environment this way.

```
$ conda create --name ml-model-heroku python=3.6
```

AcYvate the environment.

```
$ conda activate ml-model-heroku
```

or you can use venv. I recommend this way.

```
$ virtualenv venv
$ . venv/bin/activate
```

Install Requests, Flask and Gunicorn

To run the app, we need to install the Flask Python Framework with which we will build our app and Gunicorn to be our Python WSGI HTTP Server.

```
$ pip install requests[security] flask gunicorn
```

Using Exis>ng The Flask App

Now, let's use exisYng Flask app. Also, you can use create a new one in this way.

Create App Folder

```
$ mkdir app
$ cd app
```

Write Your Flask app

app.py > ...

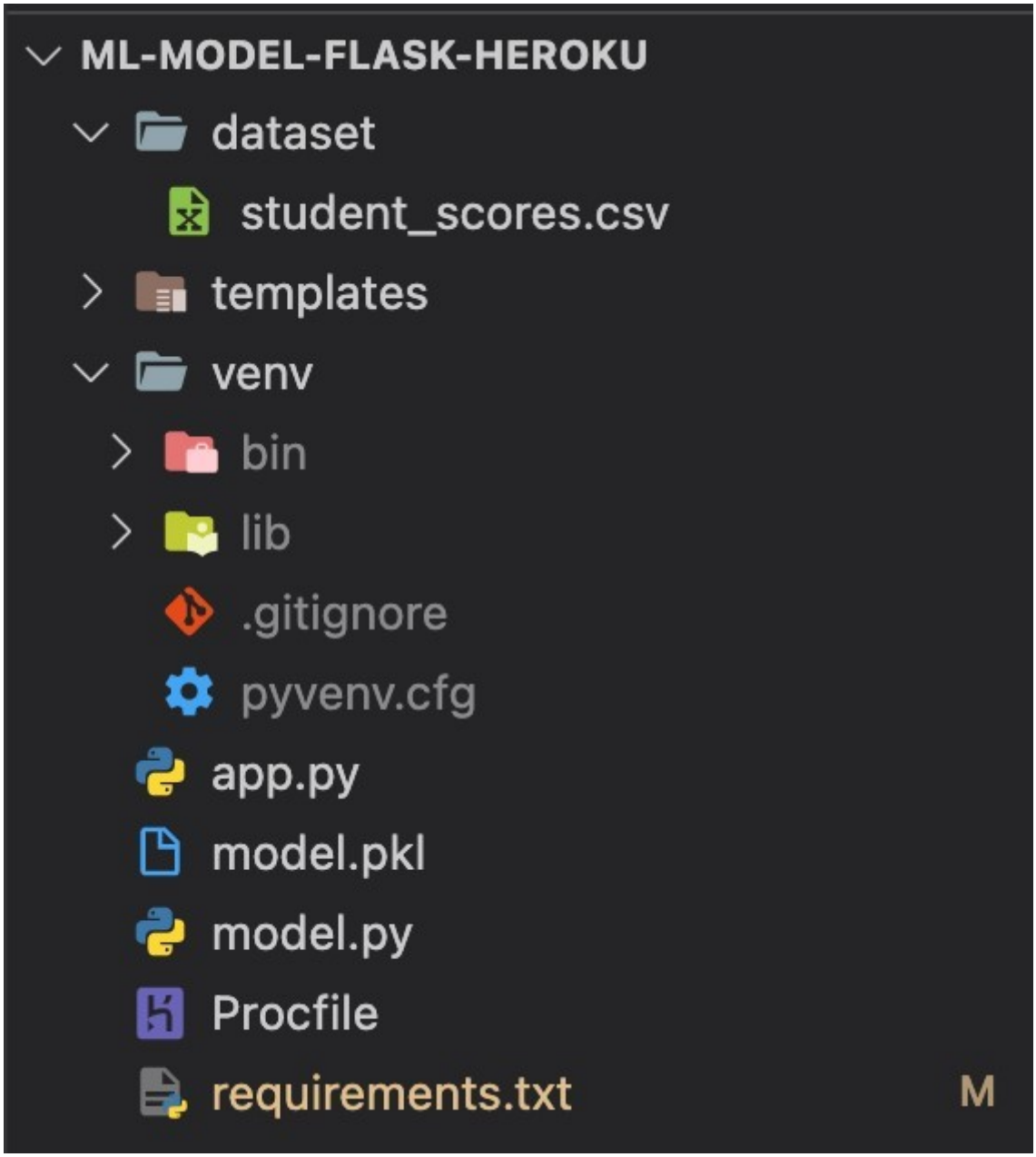
```python
import pickle
import numpy as np
from flask import Flask, request, render_template

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    init_features = float(request.form['time'])
    y_array = np.asarray(init_features)
    final_features = y_array.reshape(-1,1)
    prediction = model.predict(final_features)
    return render_template('index.html', prediction_text='Predicted Class: {}'.format(prediction))

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

(variable) form: ImmutableMultiDict

Run Flask App

Move to the parent directory.

```
$ cd ../
```



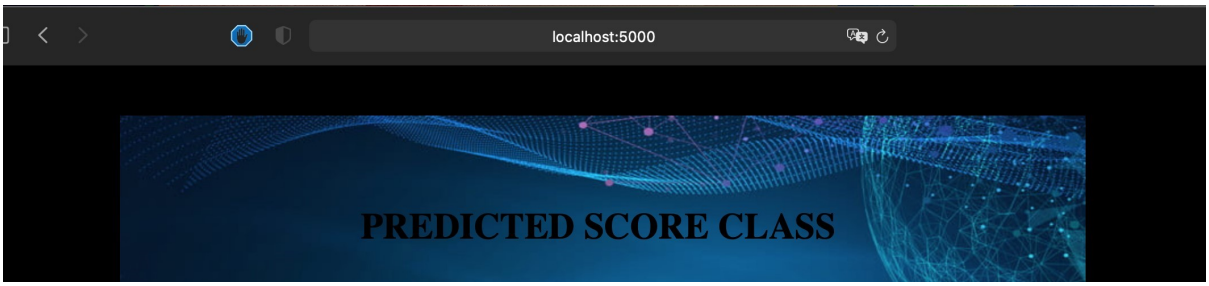
Check the files are in the correct locaYon!

To verify if the app works, try running the app first.

```
$ python app.py
```

In your browser, go to the IP given and see if the app works.

```
(base) Ardas-MacBook-Pro:ml-model-flask-heroku ardaozmen$ . venv/bin/activate
(venv) (base) Ardas-MacBook-Pro:ml-model-flask-heroku ardaozmen$ python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 187-240-900
```



Make Required Files to Deploy to Heroku

Create Requirements.txt

Now, let's freeze the environment and add it to the requirements.txt so that the required packages are installed when deploying the app.

```
$ pip freeze > requirements.txt
```

Create Procfile

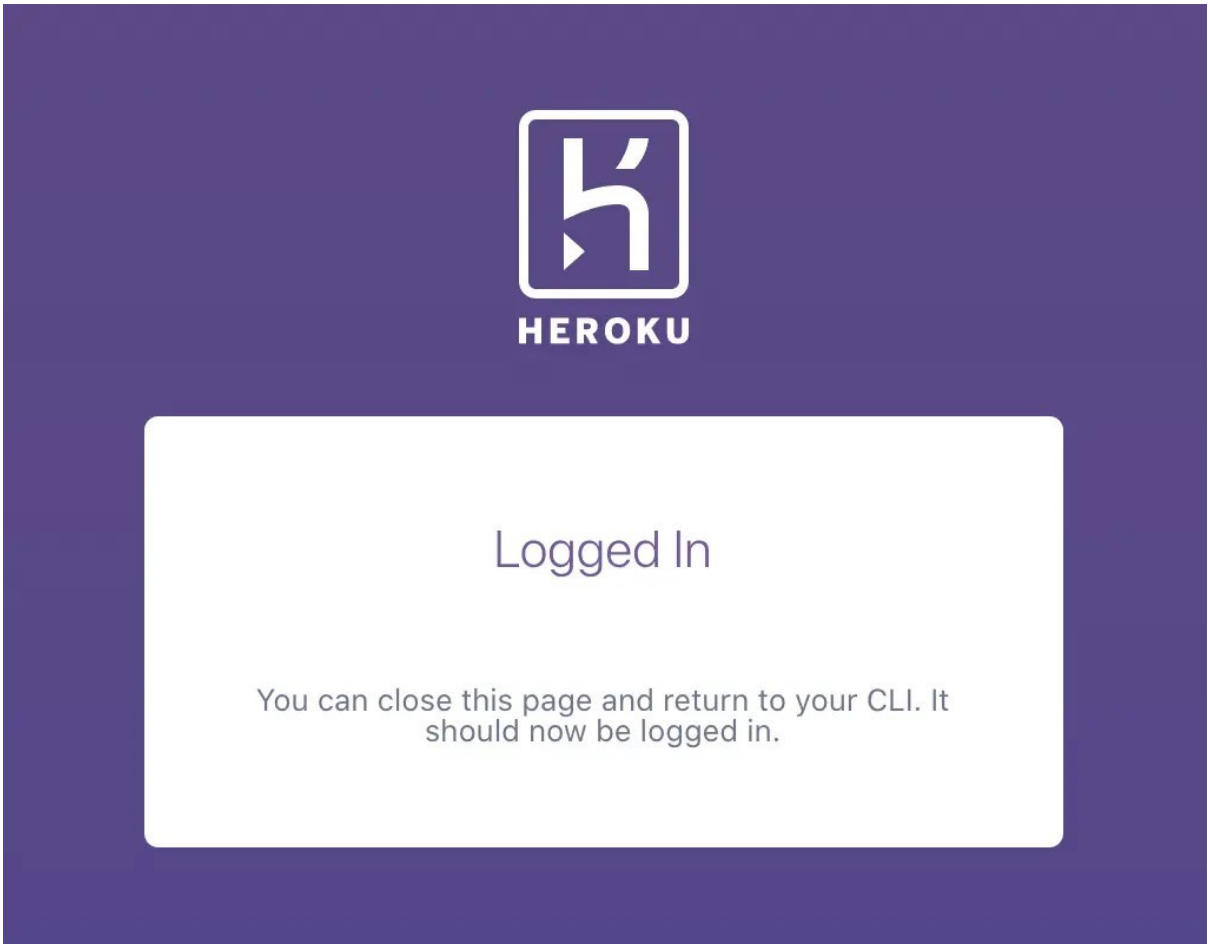
We need to create a Procfile that contains the commands that Heroku will need to run.

- To create the Procfile to run the pp.py app:

```
$ echo web: gunicorn run:app >> Procfile
```

Deploy Your App to Heroku

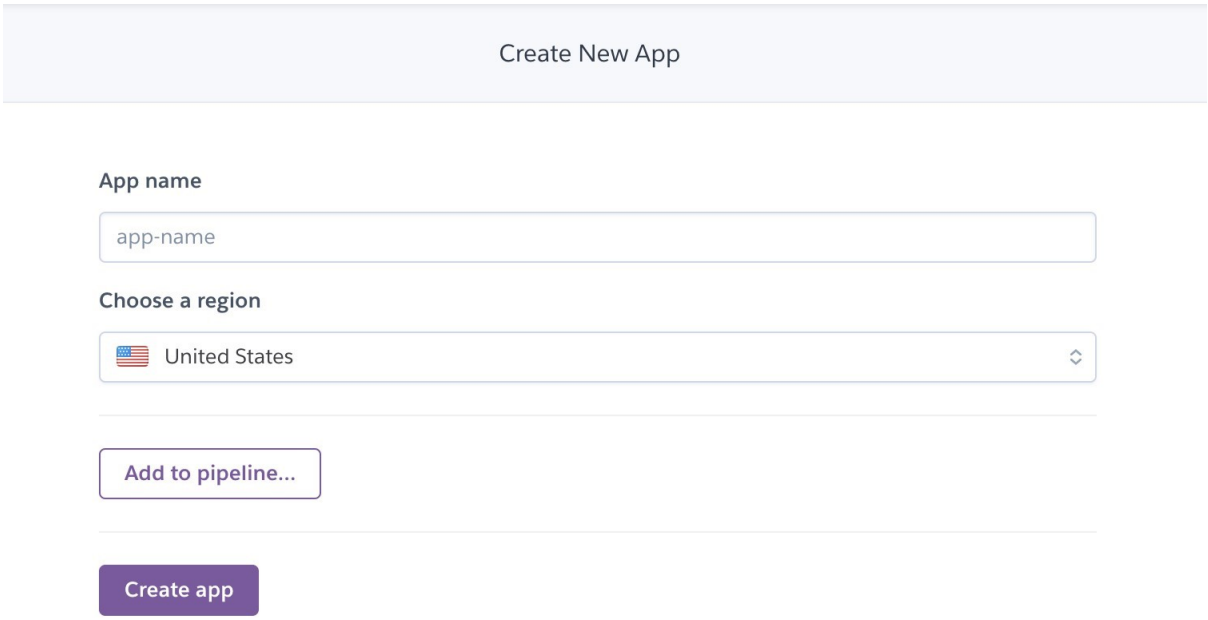
Login Heroku



Create Heroku App

```
$ heroku create <app-name>
```

or Go to Heroku Dashboard



Add Heroku App to Remote

```
$ git init
$ heroku git:remote -a <heroku-app-name>
$ git remote -v
```

Deploy the Flask App to Heroku

```
$ git add .
$ git commit -m "Init deploy"
```

```
$ git push heroku master
```

All done, the Flask app should be live.

Kill The App

DeacYvate app

```
$ heroku ps:scale web=0
```

ReacYvate App

```
heroku ps:scale web=1
```

Delete App

```
$ heroku apps:destroy <heroku-app-name>
```

Conclusion

LOOK AT THE PROJECT ON HEROKU