

# MODEL DEPLOYMENT ON THE CLOUD HEROKU

Name: MEMUDU Alimatou sadia

Batch Code: LISUM02

Date of submission: 07/08/2021

Submitted to:

## Model Deployment Stages

### Stage1: Choosing a simple data

This is the [data](#) used for this project. The dataset contains several parameters which are considered important during the application for Masters Programs.

The parameters included are:

GRE Scores ( out of 340 )

TOEFL Scores ( out of 120 )

University Rating ( out of 5 )

Statement of Purpose and Letter of Recommendation Strength ( out of 5 )

Undergraduate GPA ( out of 10 )

Research Experience ( either 0 or 1 )

Chance of Admit ( ranging from 0 to 1 )

### Stage2: Build and save a model using Flask

The model's goal is to predict the chance of a student to get admitted into a university. We used Pickle to serialize the model for future use in the [admission\\_model.py](#) file.

```
# Splitting the data
x = data.iloc[:, :-1].values
y = data.iloc[:, 7].values

# split dataset

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=0)

# Fitting linear regression Regression to the dataset
lin_regressor = LinearRegression(normalize=True)
lin_regressor.fit(X_train, y_train)

# To save the model to the disk (serialization) for future use
pickle.dump(lin_regressor, open('admission_model.pkl', 'wb'))
```

### Stage3: Deployment using Flask

Firstly, [App.py](#) is built, a flask app that used the deserialized model to accept new data and predict a student percentage to get admission.

```
1  import numpy as np
2  import pickle
3  from flask import Flask, request, render_template
4
5  app = Flask(__name__)
6
7  # Performing deserialization using pickle
8  model = pickle.load(open("admission_model.pkl", 'rb'))
9
10
11 @app.route('/')
12 def index():
13     return render_template(
14         'index.html',
15         data=[{'UR': 'University Rating'}, {'UR': 1}, {'UR': 2}, {'UR': 3}, {'UR': 4}, {'UR': 5}],
16         data1=[{'ReS': 'Research'}, {'ReS': 0}, {'ReS': 1}])
17
```

Now the function below accepts the data and return the predicted percentage

```
19 @app.route("/predict", methods=['GET', 'POST'])
20 def predict():
21     input_data = list(request.form.values())
22     if int(input_data[0]) & int(input_data[1]) & input_data[3].isdigit() & input_data[4].isdigit() & input_data[5].isdigit() == True:
23         pass
24     else:
25         print(ValueError)
26
27     input_values = [x for x in input_data]
28     arr_val = np.array(input_values)
29     prediction = model.predict(arr_val)
30     output = round(prediction[0], 2)*100
31     return render_template('index.html', prediction_text=" The Chance of Getting into the University is {} %".format(output),
32                           data=[{'UR': 'University Rating'}, {'UR': 1}, {'UR': 2}, {'UR': 3}, {'UR': 4}, {'UR': 5}],
33                           data1=[{'ReS': 'Gender'}, {'ReS': 0}, {'ReS': 1}])
34
35
36 if __name__ == '__main__':
37     app.run(debug=True)
38
```

The [index.html](#) is a file that contains the structure of the web app design and [AppStyle.css](#) is used to beautify the web design.

Secondly, you have to write: “python app.py” in the terminal to run the flask application then a link will display.

```
Terminal Local + -
PS C:\Users\Alimat sadia\my pyPrograms> cd "data science"
PS C:\Users\Alimat sadia\my pyPrograms\data science> cd "ADMISSION WEB APPLICATION"
PS C:\Users\Alimat sadia\my pyPrograms\data science\ADMISSION WEB APPLICATION> 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
* Restarting with stat
* Debugger is active!
* Debugger PIN: 603-954-664
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Thirdly, clicking on the link will direct you to the flask web application interface shown below.

The screenshot displays the 'ADMISSION CHANCE PREDICTION' web application. On the left, a 'Model Description' sidebar explains the app's purpose and lists required exam scores. The main form on the right contains input fields for 'Your GRE score', 'Your TOEFL score', 'University Rating' (a dropdown), 'Enter your SOP score', 'Enter your LOR score', 'Enter your CGPA', and 'Research' (a dropdown). A 'Predict' button is located at the bottom of the form.

**Model Description**

This web application is a chance of getting into university prediction system. The aim of this project is to predict a student's chance of getting into university based on its previous exam scores. After inputting the corresponding data, a predicted chance will display.

The exam score needed are :  
The GRE Score  
TOEFL Score  
University Rating  
SOP- Statement of Purpose  
LOR- Letter of Recommendation  
CGPA- Cumulative Grade Point  
Average  
Research-- Made a research?  
(Yes=1/No=0)

**ADMISSION CHANCE PREDICTION**

Your GRE score  
Your TOEFL score  
University Rating  
Enter your SOP score  
Enter your LOR score  
Enter your CGPA  
Research  
Predict

On this interface, a description of the web app's function is explained by the left and on the right there is the input section that collects the user's data.

This data will be fed into the deserialized model which will provide an output (percentage of change to get admitted) as illustrated below.

This screenshot shows the same web application with sample data entered into the input fields. The 'Your GRE score' field contains '250', 'Your TOEFL score' contains '100', 'University Rating' is set to '4', 'Enter your SOP score' contains '1.5', 'Enter your LOR score' contains '4.2', 'Enter your CGPA' contains '3.2', and 'Research' is set to '0'. The 'Predict' button remains visible.

**Model Description**

This web application is a chance of getting into university prediction system. The aim of this project is to predict a student's chance of getting into university based on its previous exam scores. After inputting the corresponding data, a predicted chance will display.

The exam score needed are :  
The GRE Score  
TOEFL Score  
University Rating  
SOP- Statement of Purpose  
LOR- Letter of Recommendation  
CGPA- Cumulative Grade Point  
Average  
Research-- Made a research?  
(Yes=1/No=0)

**ADMISSION CHANCE PREDICTION**

250  
100  
4  
1.5  
4.2  
3.2  
0  
Predict

Finally, clicking on the predict button will displayed the predicted value as show below.

This screenshot shows the web application after the 'Predict' button has been clicked. The input fields are now disabled, and the output 'The Chance of Getting into the University is 53.0 %' is displayed at the bottom of the form area.

**Model Description**

This web application is a chance of getting into university prediction system. The aim of this project is to predict a student's chance of getting into university based on its previous exam scores. After inputting the corresponding data, a predicted chance will display.

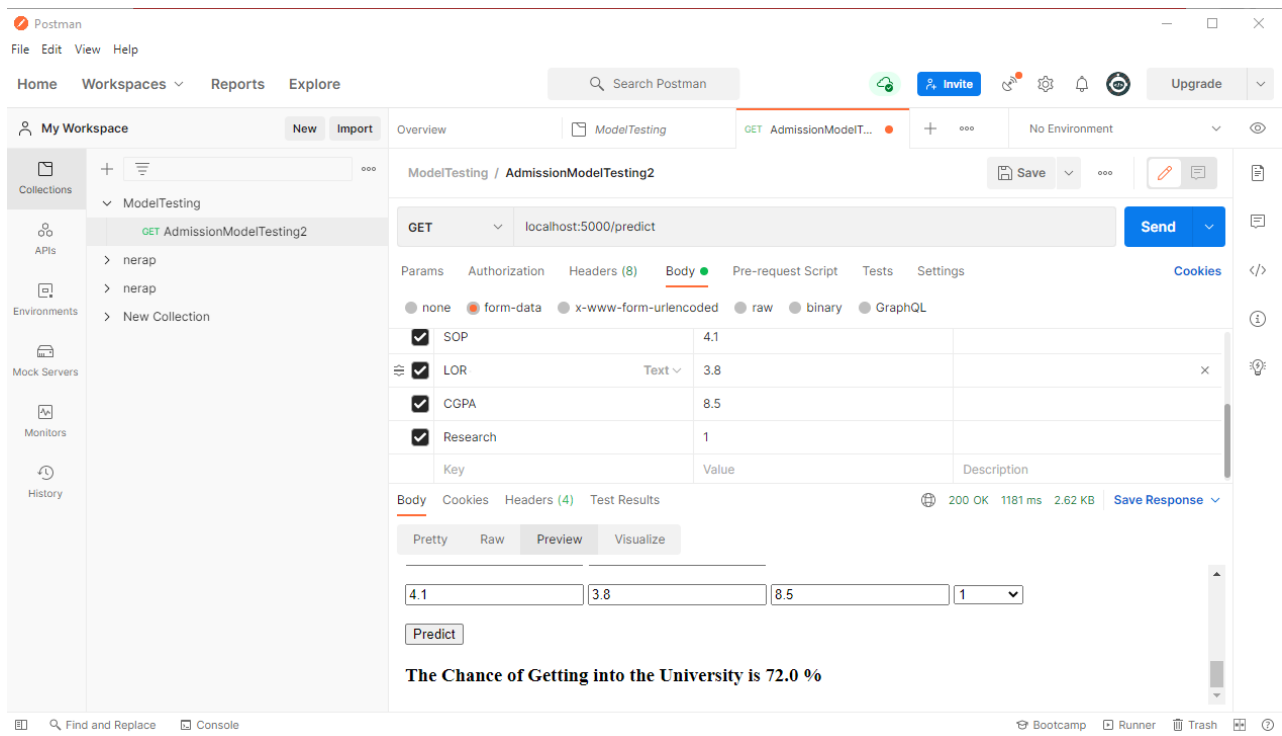
The exam score needed are :  
The GRE Score  
TOEFL Score  
University Rating  
SOP- Statement of Purpose  
LOR- Letter of Recommendation  
CGPA- Cumulative Grade Point  
Average  
Research-- Made a research?  
(Yes=1/No=0)

**ADMISSION CHANCE PREDICTION**

Your GRE score  
Your TOEFL score  
University Rating  
Enter your SOP score  
Enter your LOR score  
Enter your CGPA  
Gender  
Predict  
The Chance of Getting into the University is 53.0 %

## Step4: Testing the flask API on the cloud using POSTMAN

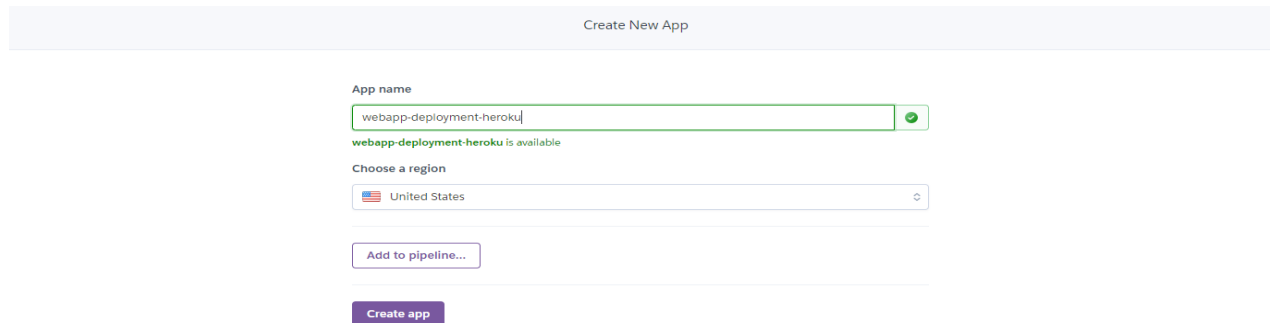
Before deploying a web app, it is recommended to test the model what I accomplished in the picture below.



After a successful testing step, the model is ready for deployment.

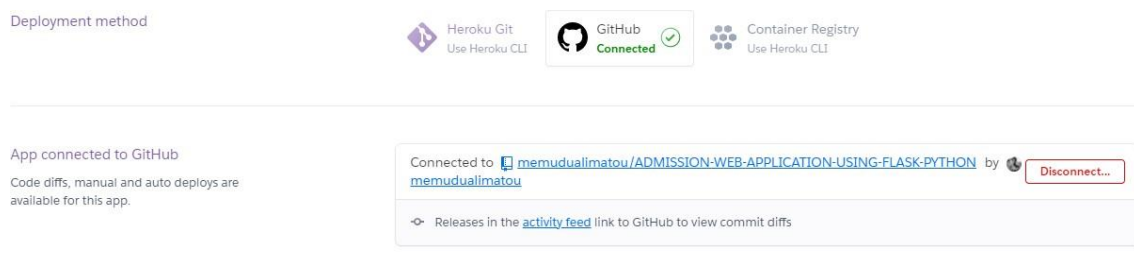
## Step5: Deployment of the flask API on the cloud using HEROKU

I created a Heroku account after the login then by the right corner, I click on “New App” to create a new app to deploy.



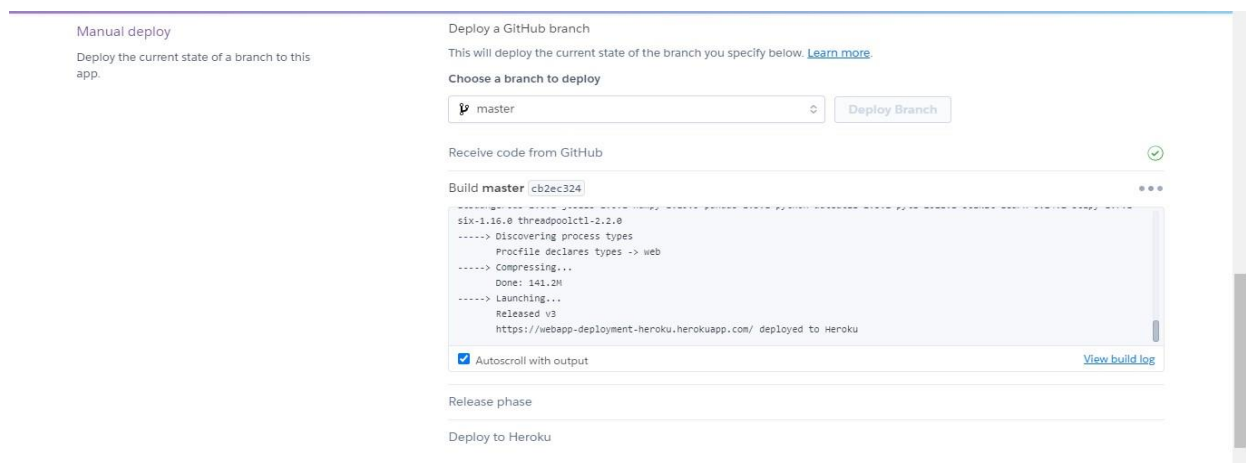
The screenshot shows the 'Create New App' form on Heroku. At the top, there's a header 'Create New App'. Below it, the 'App name' field contains 'webapp-deployment-heroku' with a green checkmark icon to its right. A message below the field states 'webapp-deployment-heroku is available'. The 'Choose a region' dropdown menu is set to 'United States'. There are two buttons: 'Add to pipeline...' and 'Create app'.

After inserting the project name, I clicked on Create app then connected this [project repository](#) to the deployment method in Heroku.



The screenshot shows the 'Deployment method' configuration page. It lists three options: 'Heroku Git' (Use Heroku CLI), 'GitHub' (Connected with a green checkmark), and 'Container Registry' (Use Heroku CLI). Below this, it states 'App connected to GitHub' and 'Code diffs, manual and auto deploys are available for this app.' To the right, it shows the connection details: 'Connected to memudualimatou/ADMISSION-WEB-APPLICATION-USING-FLASK-PYTHON by memudualimatou' with a 'Disconnect...' button. A link to 'Releases in the activity feed' is also present.

After a successful connection, I deployed the project manually, this step is all about the virtual installation of libraries required to run the web app. This project's library dependencies and their different versions are listed in the [requirements.txt](#) file. The [link](#) presents in the Build master section below directs to the web app deployed in the cloud.



The screenshot shows the 'Manual deploy' section of the Heroku dashboard. It includes a 'Deploy a GitHub branch' button. Below it, a message states 'This will deploy the current state of the branch you specify below. [Learn more](#)'. The 'Choose a branch to deploy' dropdown is set to 'master'. To the right is a 'Deploy Branch' button. Below this, it says 'Receive code from GitHub' with a green checkmark. The 'Build master' section shows the build ID 'cb2ec324' and a log of the deployment process, including 'Discovering process types', 'Profile declares types -> web', 'Compressing...', 'Done: 141.2M', 'Launching...', and 'Released v3'. A link to 'View build log' is provided. At the bottom, there's a checkbox for 'Autoscroll with output' and a 'Deploy to Heroku' button.

A deployment is garentee after a succesfull installation of all requirement libraries as shown below.

The image shows the Heroku deployment interface. On the left, under 'Manual deploy', it says 'Deploy the current state of a branch to this app.' On the right, under 'Deploy a GitHub branch', it says 'This will deploy the current state of the branch you specify below. [Learn more](#)'. Below this, there's a section 'Choose a branch to deploy' with a dropdown menu showing 'master' and a 'Deploy Branch' button. To the right of the dropdown are four green checkmarks indicating successful steps: 'Receive code from GitHub', 'Build master cb2ec324', 'Release phase', and 'Deploy to Heroku'. Below these steps, it says 'Your app was successfully deployed.' and there is a 'View' button.

Clicking on the “View” button directs to the deployed web app [link](#). The admission Flask API will display. Check the URL section of my web browser the “webapp-deployment-heroku” has been successfully deployed into cloud using Heroku.

The image is a screenshot of a web browser displaying the 'ADMISSION CHANCE PREDICTION' web application. The browser's address bar shows 'webapp-deployment-heroku.herokuapp.com'. The page has a purple background. On the left, there's a 'Model Description' section with text about the application's purpose and a list of exam scores needed. On the right, there's a form titled 'ADMISSION CHANCE PREDICTION' with input fields for 'Your GRE score', 'Your TOEFL score', 'University Rating', 'Enter your SOP score', 'Enter your LOR score', 'Enter your CGPA', and 'Research'. A 'Predict' button is at the bottom of the form.

Testing the web app by inserting some data and view the result.

The screenshot shows a web application titled "ADMISSION CHANCE PREDICTION". On the left, a "Model Description" box explains the application's purpose and lists required exam scores: GRE, TOEFL, University Rating, SOP, LOR, CGPA, and Average. The main form contains input fields for "Your GRE score" (250), "Your TOEFL score" (100), "University Rating" (3), "Enter your SOP score" (1.5), "Enter your LOR score" (4.2), "Enter your CGPA" (9.5), and "Gender" (1). A "Predict" button is at the bottom.

The predicted chance of getting admitted into a university is displayed.

This screenshot shows the same web application after a prediction. The "Predict" button now displays the result: "The Chance of Getting into the University is 69.0 %". The input fields remain the same as in the previous screenshot.

Project repository: <https://github.com/memudualimatou/ADMISSION-WEB-APPLICATION-USING-FLASK-PYTHON>

Deployed web app link: <https://webapp-deployment-heroku.herokuapp.com>