**Data Science Internship**

**Deployment on Heroku**

**Name: Taimoor Razi**

**Batch Code: LISUM11: 30**

**Submitted to: Data Glacier**

**Submission Date: 25th July, 2022**

Table of Contents

[Step 1: Generating Dataset 3](#_Toc109640464)

[Step 2: Training & Saving Model 4](#_Toc109640465)

[Step 3: Create web app using flask (integrating HTML) 5](#_Toc109640466)

[Step 4: Create Procfile and requirement.txt 7](#_Toc109640467)

[Step 5: Upload files on Github repository 8](#_Toc109640468)

[Step 6: Creating and Connecting Heroku app to Github repository 9](#_Toc109640469)

[Step 7: Deploying app on Heroku 10](#_Toc109640470)

# Step 1: Generating Dataset

For this assignment, I created my own simple dataset as an excel file with four columns and eight entries. The target variable is the people salaries while the independent variables/training features are “experience”,” test\_score” and “interview\_score” of an individual. The dataset is shown in the following figure.

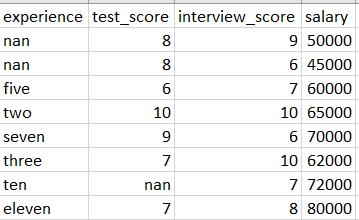


Figure 1: Dataframe

# Step 2: Training & Saving Model

A simple python file containing the model is made as model.py.

Firstly, some feature engineering is done. The missing values in the experience column are assumed to be for fresh graduate with zero experience and hence are replace with integer 0. The missing value for the test\_score is column is replaced by the average of the column feature. Finally, the categorical values in the experience column is replaced with integer values by creating a function to handle it. Since the dataset is very small, the whole data is trained on a linear regression model. Finally the model is saved as a pickle file The figure below shows all the steps.

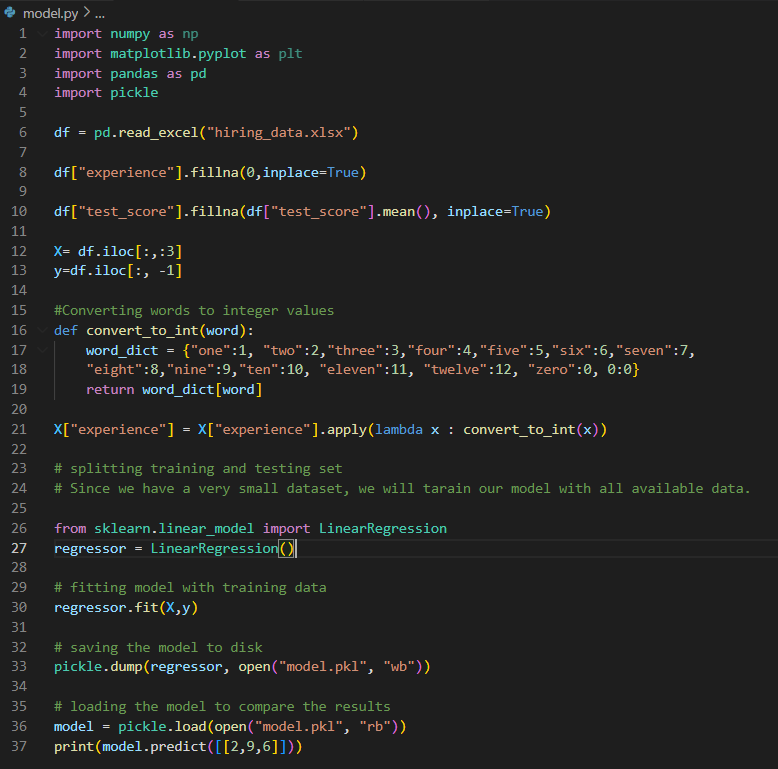


Figure 2: Code for model.py

# Step 3: Create web app using flask (integrating HTML)

A new python file called “app.py” is created.

After importing flask, we create our web app by initializing a flask object which act as a WSGI application.

Then a decorator is created (app.route) with the url “/” as the argument. A home function is created inside the decorator which returns html.index file as render\_template. This html.index file is created to give our web app some text and design on home display.

Then another decorator (app.route()) is created with url as “/predict” and methods=[“POST”] request as the second argument . This means that after we input the score the page will be redirected to this route. A predict function is defined inside app.route(). And this make prediction of salary for the input features. The result is rendered on HTML GUI as the function returns the html.index file as render template.

The code written and the app display after predictions is shown in the following figures.

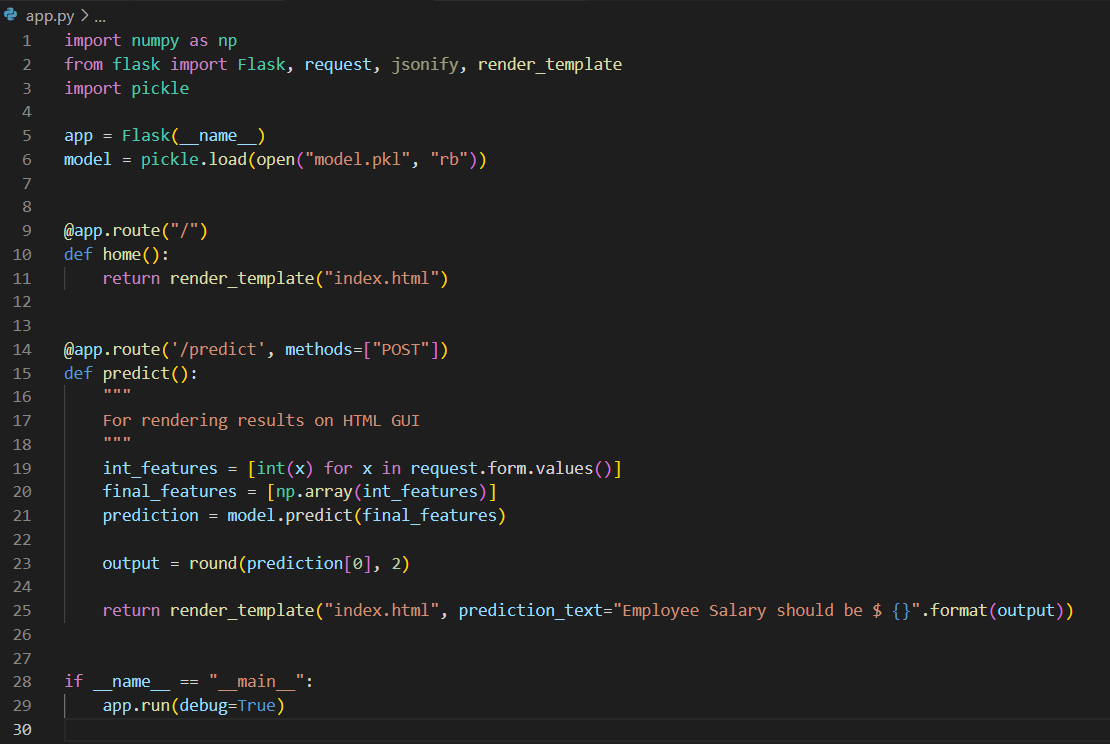


Figure 3: Code for app.py

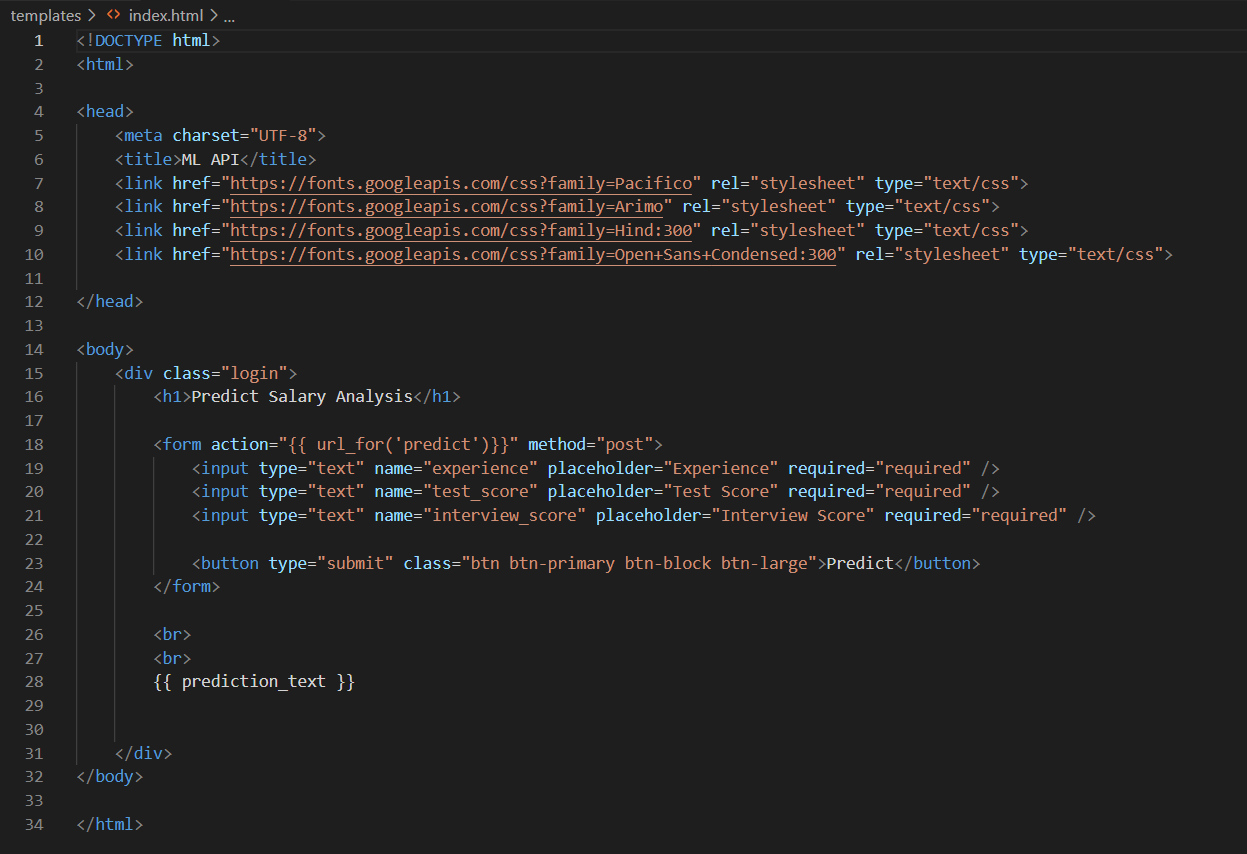


Figure : Code for index.html

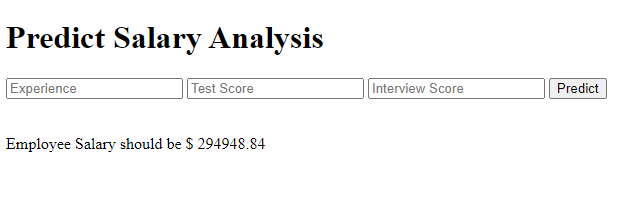


Figure 5: Web App Display

# Step 4: Create Procfile and requirement.txt

Make a Procfile specifying our Webserver as gunicorn and our app as “app.py” and “app” again as instance of our Flask object. This is shown in the following figure.

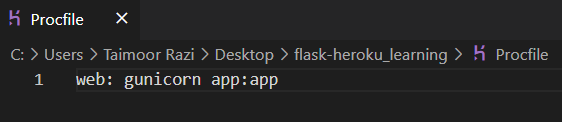


Figure 6: Procfile

Also, we will make a requirement.txt containing dependencies and their version used in our environment. This can be done manually or generated automatically. Heroku will read this to understand which packages and versions are we using. This is shown in the following figure.

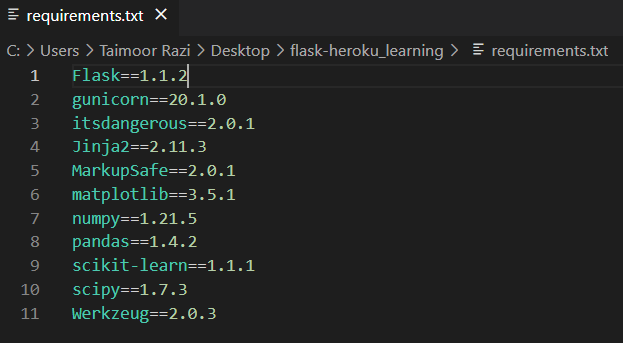


Figure 7: requirements.txt

# Step 5: Upload files on Github repository

Next, we upload all the files necessary (model.pkl, app.py, Procfile, html.index, requirements.txt) in a new Github repository. The result of this step is shown in the figure below.

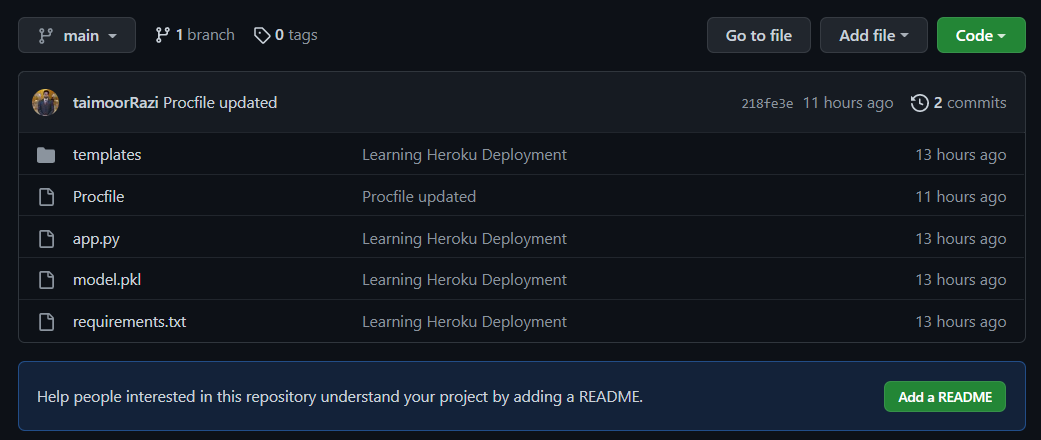


Figure 8: Files uploaded to Github repository

# Step 6: Creating and Connecting Heroku app to Github repository

Then we create an account on Heroku platform and create an app with name “salarypredictapi1”. And connect the deployment of app to our github repository.

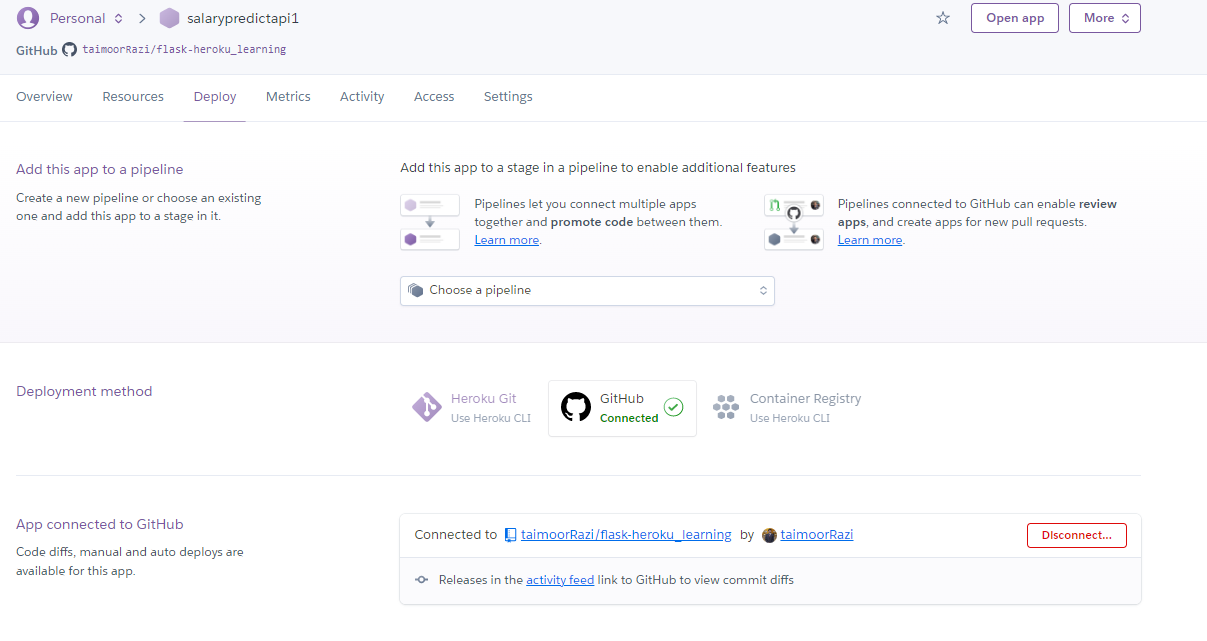


Figure 9: Heroku app connected to Github repository

# Step 7: Deploying app on Heroku

And then finally we deploy it and the result shown in the following figures.

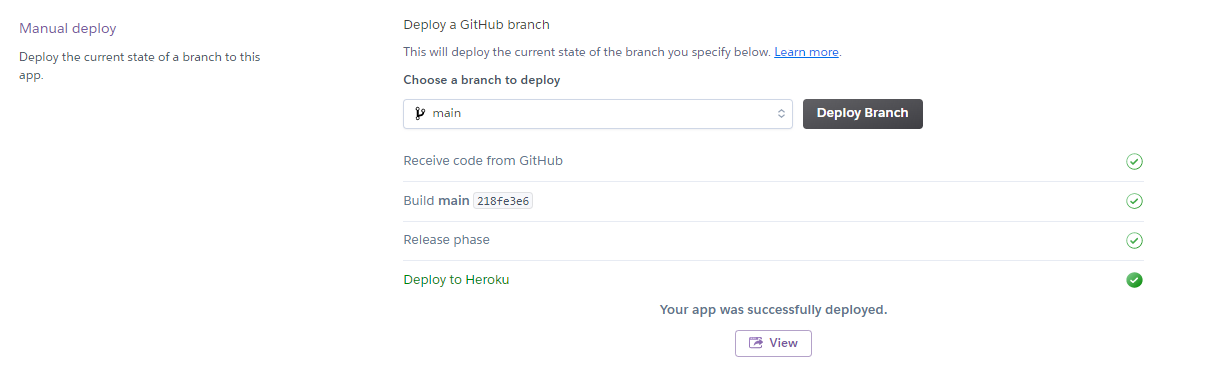


Figure 10: App deployed on Heroku

The app gets built on Heroku platform and the url of app where it is hosted is provided. The app looks exactly the same how it looked on our local system. It can be tested by putting some input features and see if it is working as intended.

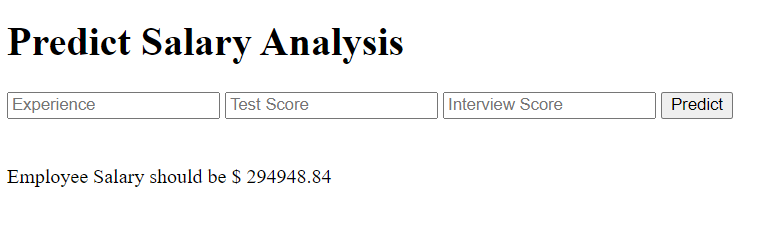


Figure 11: App display