

Name: Reeka Hazarika

Batch code: LISP01

Submission date: 26-MAR-2021

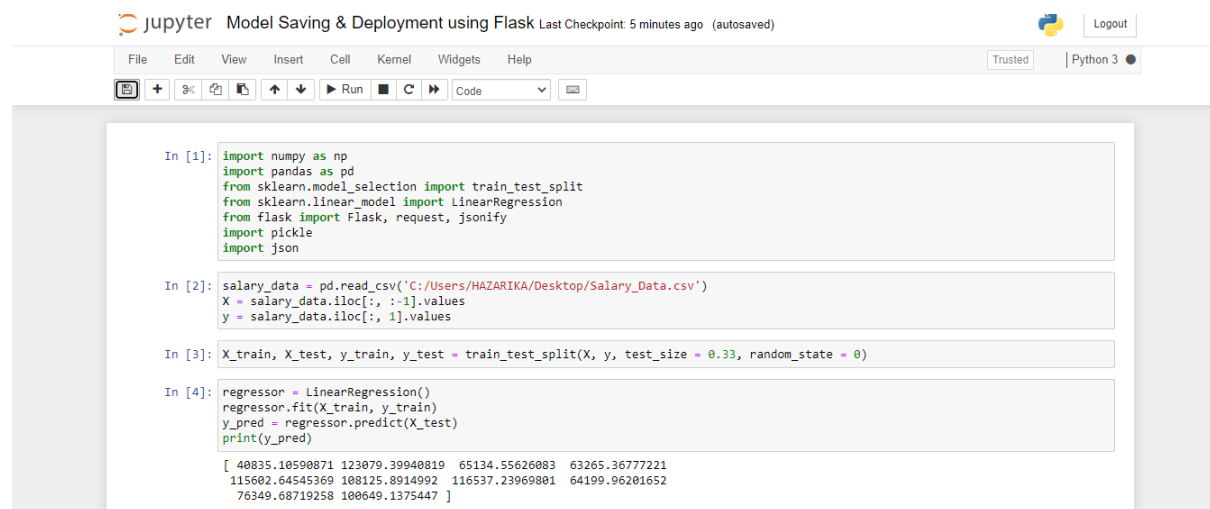
Submitted to: Data Glacier

Cloud and API deployment on Heroku

Step 1:

Train the ML model:

Predict the salary of an employee based on experience using Linear Regression Model.



The image shows a Jupyter Notebook interface with the title "Model Saving & Deployment using Flask". The notebook contains four code cells. The first cell imports necessary libraries: numpy, pandas, sklearn (model_selection and linear_model), flask, pickle, and json. The second cell reads a CSV file from the local path 'C:/Users/HAZARIKA/Desktop/Salary_Data.csv', extracts features (X) and target values (y), and splits them into training and testing sets using train_test_split. The third cell performs the same split operation. The fourth cell creates a LinearRegression model, fits it to the training data, and predicts the salary for the test data. The output of the fourth cell is a list of predicted salaries.

```
In [1]: import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from flask import Flask, request, jsonify
import pickle
import json

In [2]: salary_data = pd.read_csv('C:/Users/HAZARIKA/Desktop/Salary_Data.csv')
X = salary_data.iloc[:, :-1].values
y = salary_data.iloc[:, 1].values

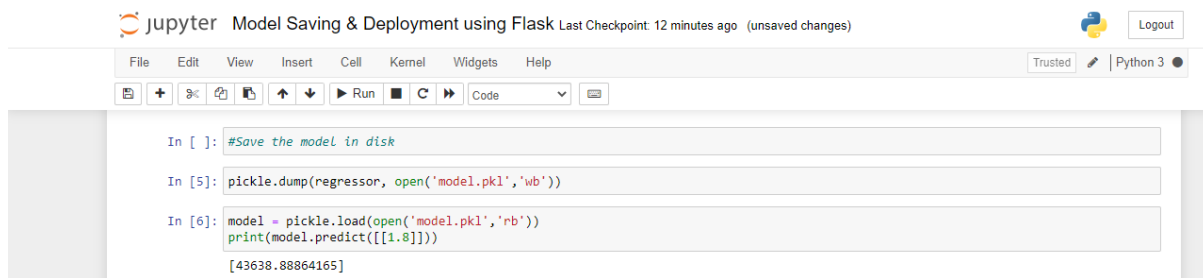
In [3]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 0)

In [4]: regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
print(y_pred)

[ 40835.10590871 123079.39940819  65134.55626083  63265.36777221
 115602.64545369 108125.8914992  116537.23969801  64199.96201652
 76349.68719258 100649.1375447 ]
```

Step 2:

Saving the trained model to the disk using the *pickle* library.



The screenshot shows a Jupyter Notebook titled "Model Saving & Deployment using Flask". The interface includes a top bar with the Jupyter logo, title, and a "Logout" button. Below the top bar is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar with icons for file operations and execution is also present. The notebook contains three code cells:

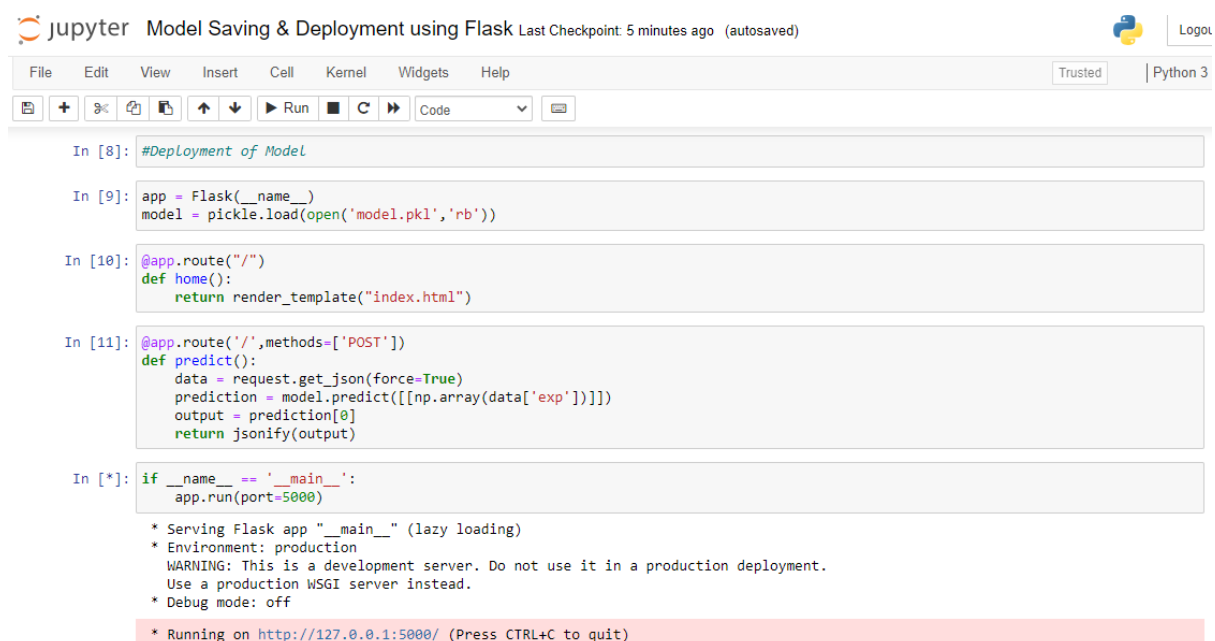
```
In [ ]: #Save the model in disk

In [5]: pickle.dump(regressor, open('model.pkl','wb'))

In [6]: model = pickle.load(open('model.pkl','rb'))
        print(model.predict([[1.8]]))
        [43638.88864165]
```

Step 3:

Deployment of Model



The screenshot shows a Jupyter Notebook titled "Model Saving & Deployment using Flask". The interface includes a top bar with the Jupyter logo, title, and a "Logout" button. Below the top bar is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar with icons for file operations and execution is also present. The notebook contains five code cells:

```
In [8]: #Deployment of Model

In [9]: app = Flask(__name__)
        model = pickle.load(open('model.pkl','rb'))

In [10]: @app.route("/")
        def home():
            return render_template("index.html")

In [11]: @app.route('/',methods=['POST'])
        def predict():
            data = request.get_json(force=True)
            prediction = model.predict([np.array(data['exp'])])
            output = prediction[0]
            return jsonify(output)

In [*]: if __name__ == '__main__':
        app.run(port=5000)

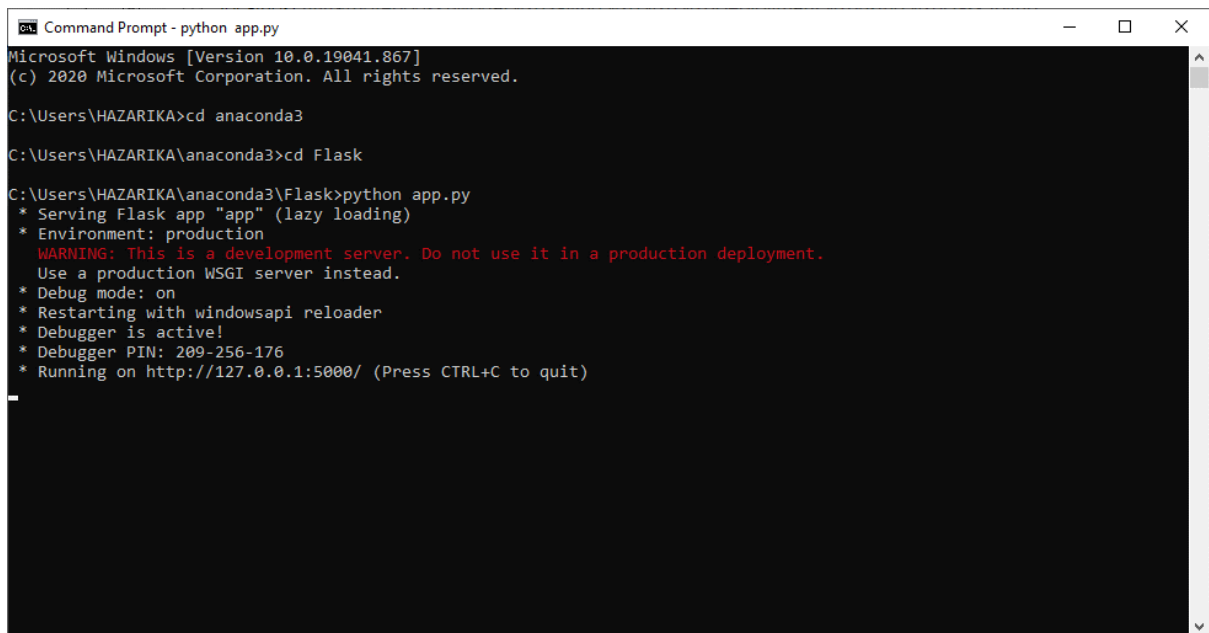
* Serving Flask app "__main__" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

- Created the instance of the *Flask()* and loaded the model.
- Bounded “/” with the method *predict()* in which predict method gets the data from the json passed by the requestor.
- *model.predict()* method takes input from the json and converts it into 2D *numpy array* the results are stored into the variable named *output*.
- Return this variable after converting it into the json object using flask's *jsonify()* method.
- Run our server by following above code section and using port 5000.

Step 4:

Checking python app.py file in CMD



```
Command Prompt - python app.py
Microsoft Windows [Version 10.0.19041.867]
(c) 2020 Microsoft Corporation. All rights reserved.

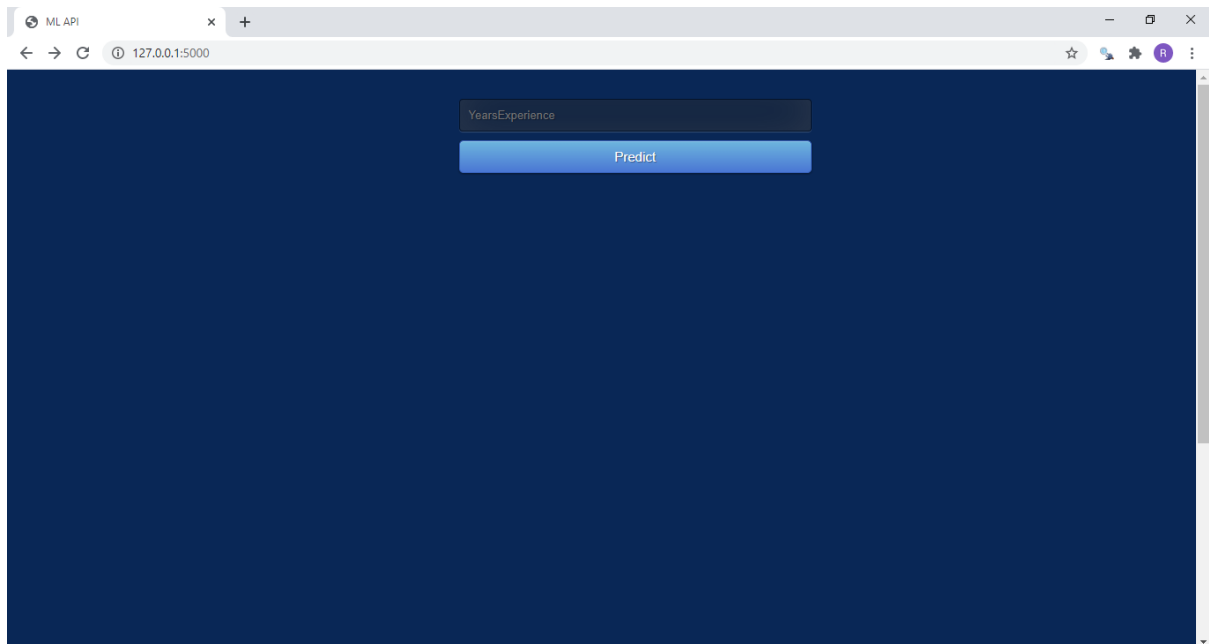
C:\Users\HAZARIKA>cd anaconda3

C:\Users\HAZARIKA\anaconda3>cd Flask

C:\Users\HAZARIKA\anaconda3\Flask>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 windowsapi reloader
* Debugger is active!
* Debugger PIN: 209-256-176
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Step 5:

Creating the Web App by typing the URL in the browser



Step 6:

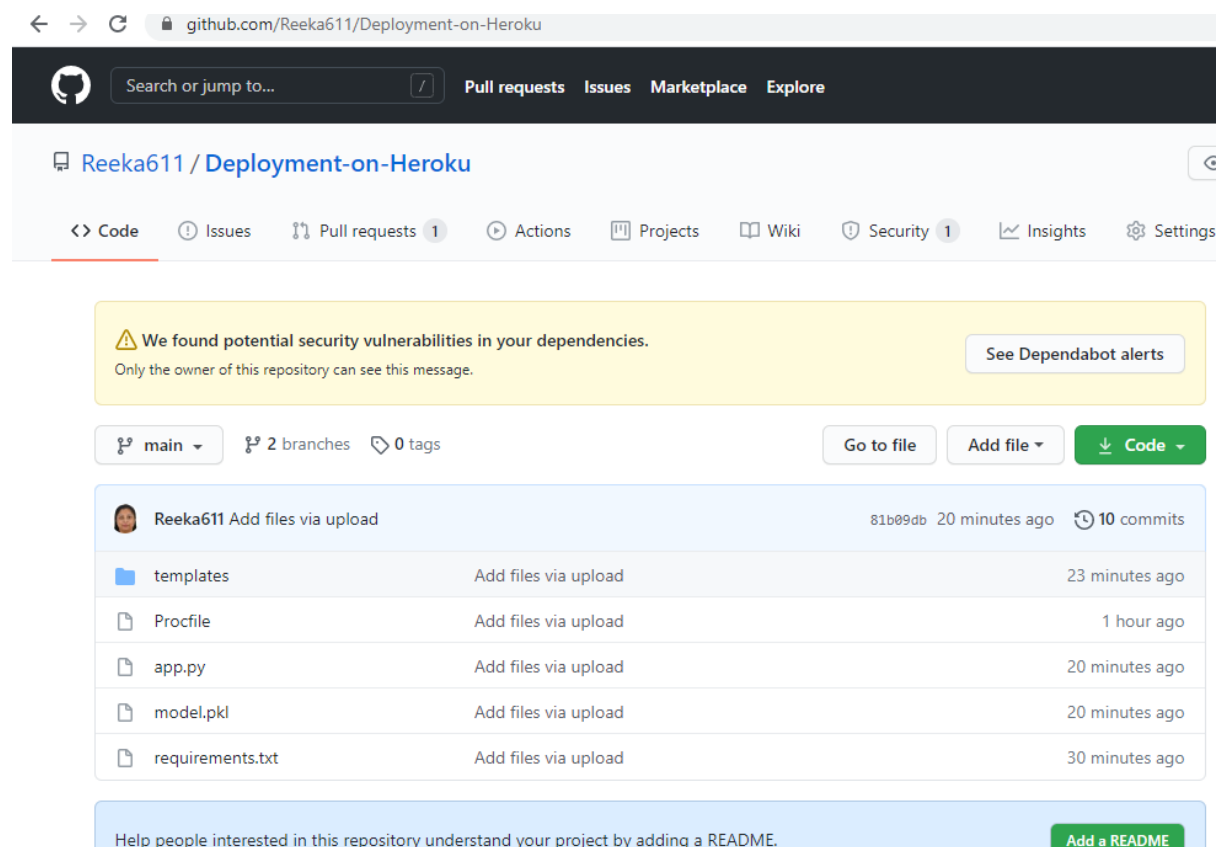
Create **Procfile** file which specifies the commands that are executed by a Heroku app on startup. `web: gunicorn app:app.`

Running the command `pip freeze > requirements.txt` in CMD for creating **requirements.txt** file which will contain all of the dependencies for the flask app.

Step 7:

Create a repository in Github and Commit the code

Link: <https://github.com/Reeka611/Deployment-on-Heroku>

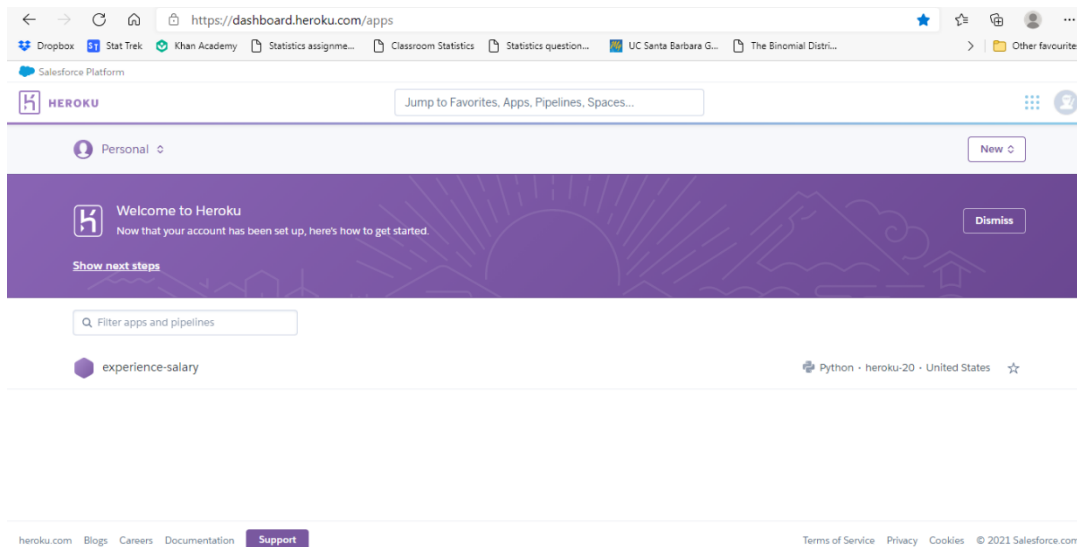


The screenshot shows the GitHub repository page for 'Reeka611/Deployment-on-Heroku'. At the top, there's a navigation bar with the GitHub logo, a search bar, and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below this, the repository name 'Reeka611 / Deployment-on-Heroku' is displayed. A secondary navigation bar includes links for 'Code', 'Issues', 'Pull requests' (with a badge '1'), 'Actions', 'Projects', 'Wiki', 'Security' (with a badge '1'), 'Insights', and 'Settings'. A yellow warning box states: 'We found potential security vulnerabilities in your dependencies. Only the owner of this repository can see this message.' with a button 'See Dependabot alerts'. Below the warning, repository statistics show 'main' branch, '2 branches', and '0 tags'. Action buttons include 'Go to file', 'Add file', and 'Code'. A table lists recent file uploads:

File Name	Action	Time
templates	Add files via upload	23 minutes ago
Procfile	Add files via upload	1 hour ago
app.py	Add files via upload	20 minutes ago
model.pkl	Add files via upload	20 minutes ago
requirements.txt	Add files via upload	30 minutes ago

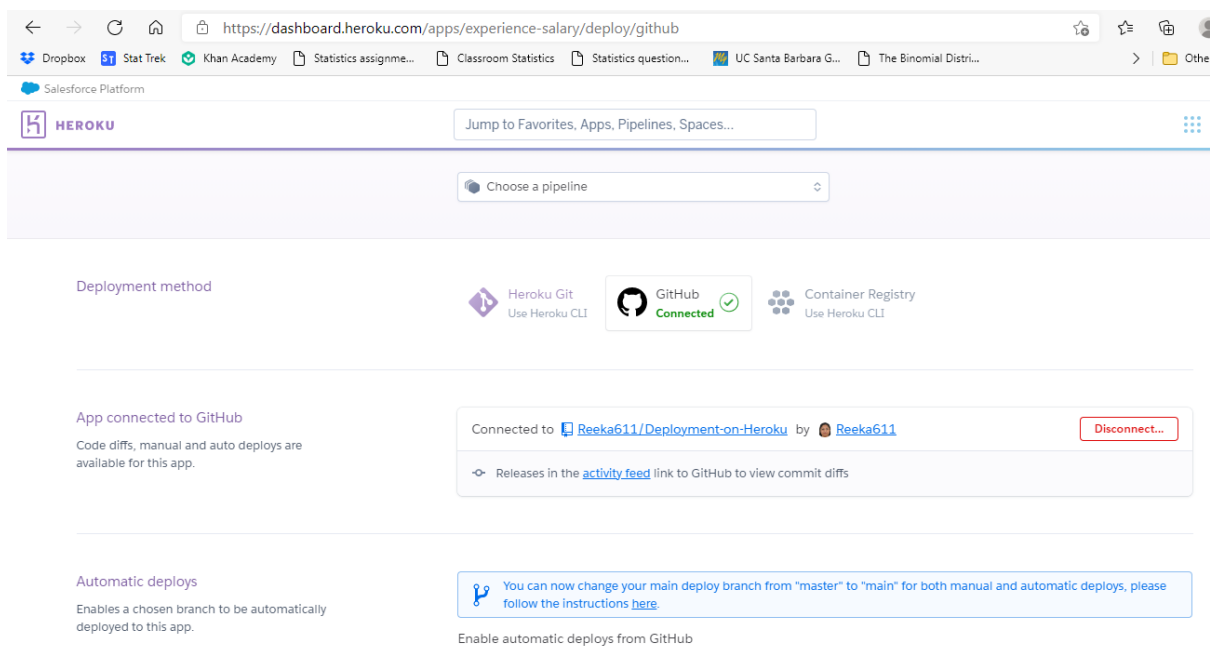
At the bottom, a blue box encourages adding a README with the text: 'Help people interested in this repository understand your project by adding a README.' and a button 'Add a README'.

Step 8: Create an Account in Heroku and then create an app.



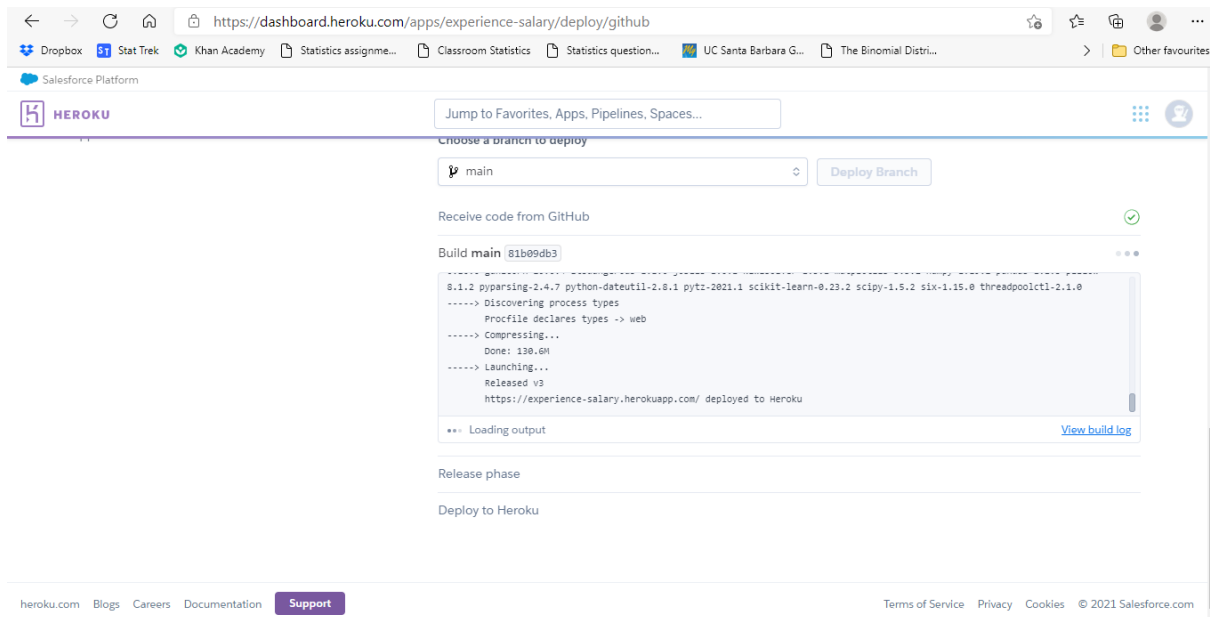
Step 9:

Link Github Account with Heroku app



Step 10:

Deploy the Model on Heroku



Step 11:

App successfully deployed

[ML API \(experience-salary.herokuapp.com/\)](https://experience-salary.herokuapp.com/)

