Week 4: Deployment on Flask

Name: Zyad Al-Azazi

Batch Code: LISP01

Submission Date: 31/03/2021

Submitted to: Data Glacier

Creating the Model:

For more details refer to the Jupyter notebook on the repository: [zyadalazazi/healthcare-cost-prediction (github.com)](github.com).

1) Feature engineering, training and testing the model.

```
In [24]: x_train, x_test, y_train, y_test = train_test_split(df_health, charges, test_size = 0.2, random_state = 100)

         x_incls_cons = sm.add_constant(x_train)
         model = sm.OLS(y_train, x_incls_cons)
         results = model.fit()

         pd.DataFrame({'coef':results.params, 'p-value':round(results.pvalues, 9)})
```

Out[24]:

	coef	p-value
const	-11307.283746	0.000000
age	254.023936	0.000000
sex	13.432983	0.971837
bmi	308.710555	0.000000
children	474.015667	0.002857
smoker	23478.307798	0.000000
region	-385.005527	0.027306

```
In [25]: x_train = x_train.drop('sex', axis = 1)
         x_train = x_train.drop('region', axis = 1)

         x_test = x_test.drop('sex', axis = 1)
         x_test = x_test.drop('region', axis = 1)

         final_model = make_pipeline(StandardScaler(), LinearRegression())
         final_model.fit(x_train, y_train)
         final_model.score(x_test, y_test)
```

2) Using pickle to dump and load the model for the purpose of testing.

```
In [27]: import pickle

In [28]: with open('model_pickle', 'wb') as f:
             pickle.dump(final_model, f)

In [29]: with open('model_pickle', 'rb') as f:
             m = pickle.load(f)

In [33]: m.predict(pd.DataFrame([[20, 19.5, 0, 1]]))
Out[33]: array([22998.69308379])
```

Creating the Python Application and GUI:

1) Creating the .py application using Brackets text editor.

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

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

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

@app.route('/predict', methods = ['POST'])
def predict():

    age_value = request.form['age']
    bmi_value = request.form['bmi']
    children_number = request.form['children']
    smoker_value = request.form['smoker']
    query_df = pd.DataFrame([[age_value, bmi_value, children_number, smoker_value]])

    prediction = model.predict(query_df)
    output = round(prediction[0], 2)

    return render_template('webpage.html', prediction_text = 'Predicted Charge: {} USD'.format(output))


if __name__ == '__main__':
    app.run(port = 9566)
```

2) Creating the .html file for GUI.

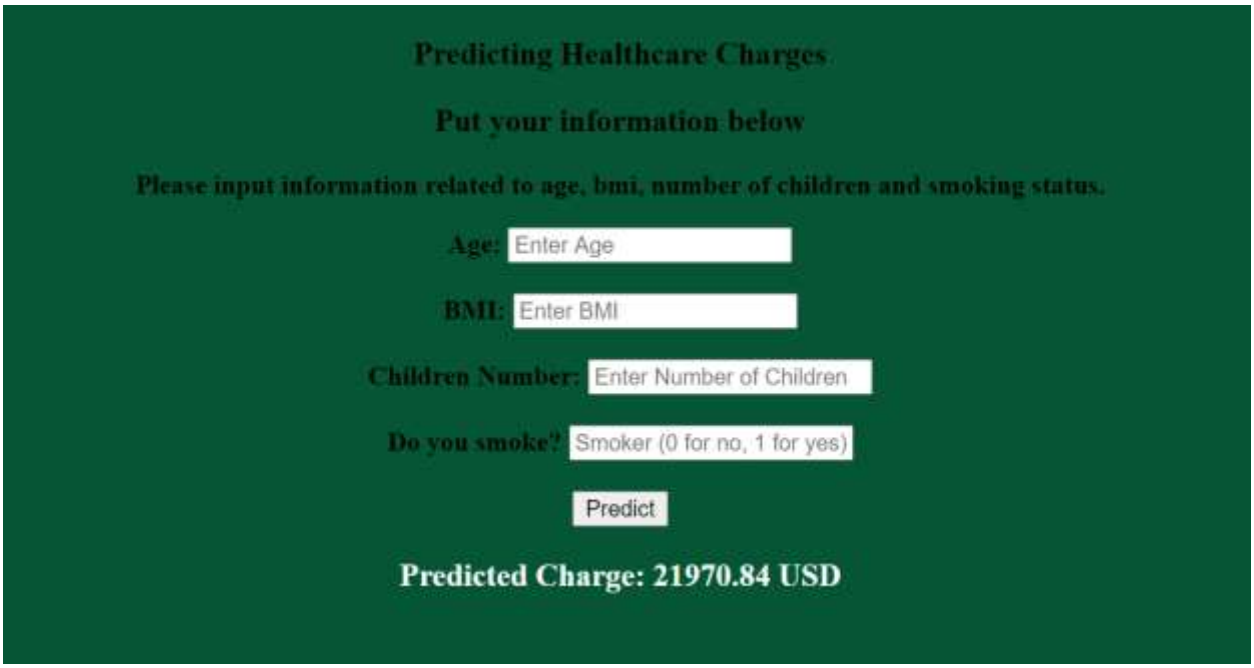
```html
<!DOCTYPE html>
<html>
<head>
<title>Predicting Healthcare Charges</title>
</head>
<body>
<center>
<h3><b><center>Predicting Healthcare Charges</center></b></h3>
<h3><b><center>Put your information below</center><b></h3>

<p><font color = "black">Please input information related to age, bmi, number of children and smoking status.</font></p>

<div class = "Prediction">
    <center>
    <form action = "{{ url_for('predict')}}", method="POST">
        <b> Age: <input type = "text" name = "age" placeholder = "Enter Age" required = "required" /><br><br>
        BMI: <input type = "text" name = "bmi" placeholder = "Enter BMI" required = "required" /><br><br>
        Children Number: <input type = "text" name = "children" placeholder = "Enter Number of Children" required = "required" /><br><br>
        Do you smoke? <input type = "text" name = "smoker" placeholder = "Smoker (0 for no, 1 for yes)" required = "required" /><br><br>

        <button type = "submit", class = "btn btn-primary btn-block btn-large">Predict</button>
    </form>
    </center>
        <h3><font color = 'white'>{{prediction_text}}</font></h3>
</div>
</center>
<body style="background-color:#065555;">

</body>
</html>
```

Running Flask and Local Testing of the Application:

1) Running Flask.

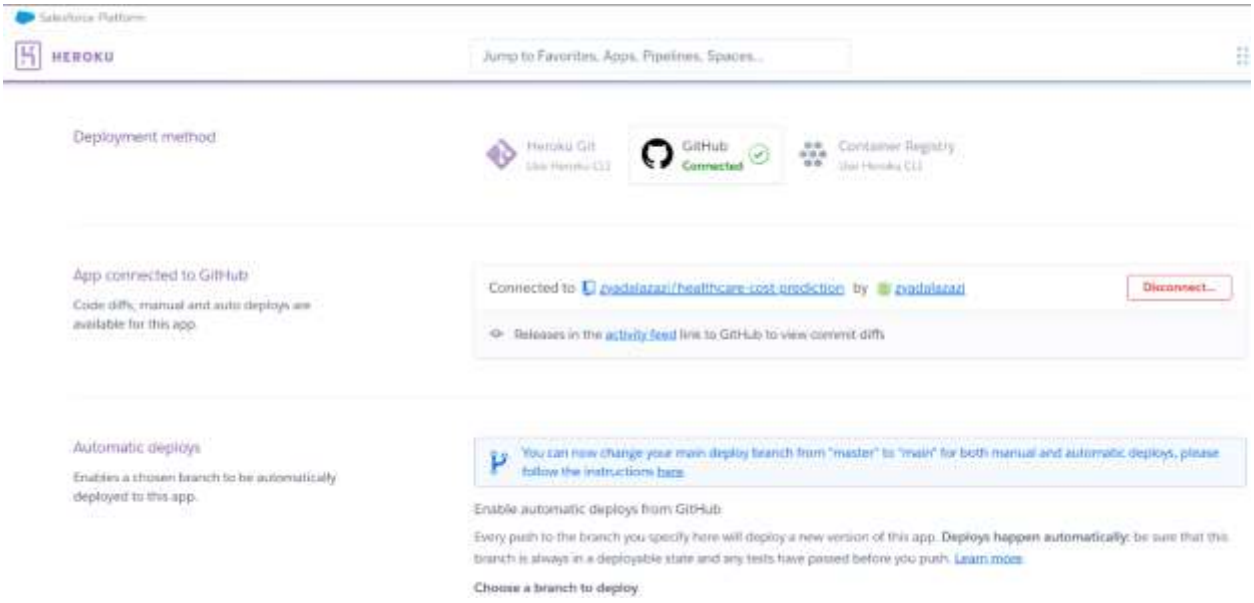
```
C:\Users\MEPI\Documents\Flask>env\Scripts\activate

(env) C:\Users\MEPI\Documents\Flask>FLASK_APP=Deployed_Model.py
'FLASK_APP' is not recognized as an internal or external command,
operable program or batch file.

(env) C:\Users\MEPI\Documents\Flask>set FLASK_APP=Deployed_Model.py

(env) C:\Users\MEPI\Documents\Flask>flask run
 * Serving Flask app "Deployed_Model.py"
 * 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)
127.0.0.1 - - [28/Mar/2021 16:28:41] "←[37mGET / HTTP/1.1←[0m" 200 -
127.0.0.1 - - [28/Mar/2021 16:28:56] "←[37mPOST /predict HTTP/1.1←[0m" 200 -
```

2) Result from the app.

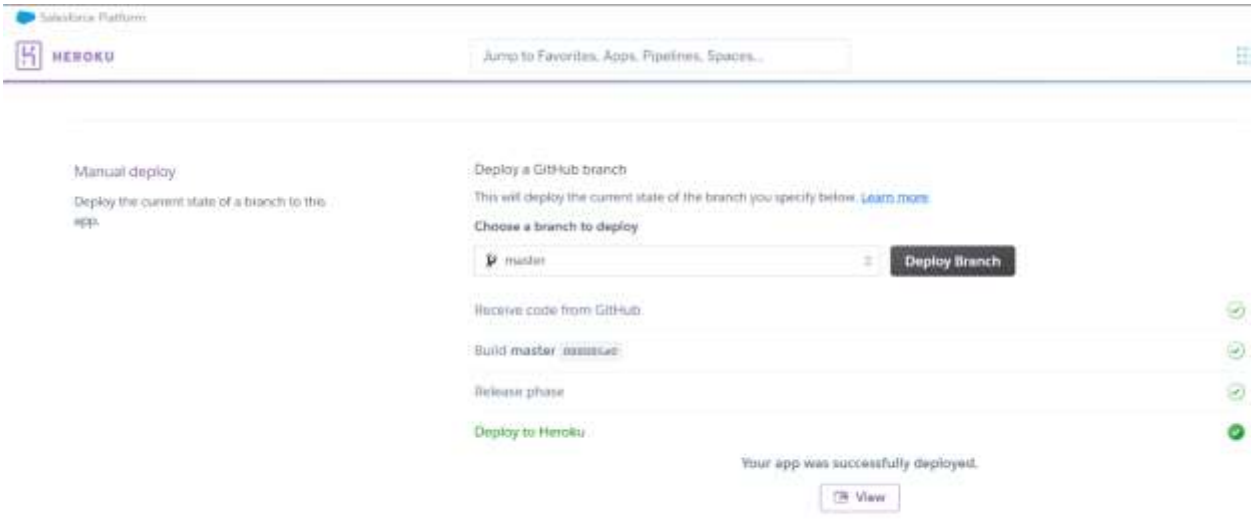


Deployment on Heroku:

1) Linking Heroku to the GitHub repository containing the files of the application.



2) Choosing the “Manual Deploy” option.



3) Checking the result of deployment.