



Data Science Intern at Data Glacier

Week 4: Deployment on Flask

Name: Okeoma Ihunwo

Batch Code: LISUM26

Submission Date: 3rd November 2023

Submitted to: Data Glacier

Table of Contents:

1. Introduction	3
2. Data Information.....	3
Attribute Information	3
3. Build Machine Learning Model	4
4. Turning Model into Flask Framework.....	5
predict.html.....	5
Style.css	6
App.py	6
Running the App.....	7

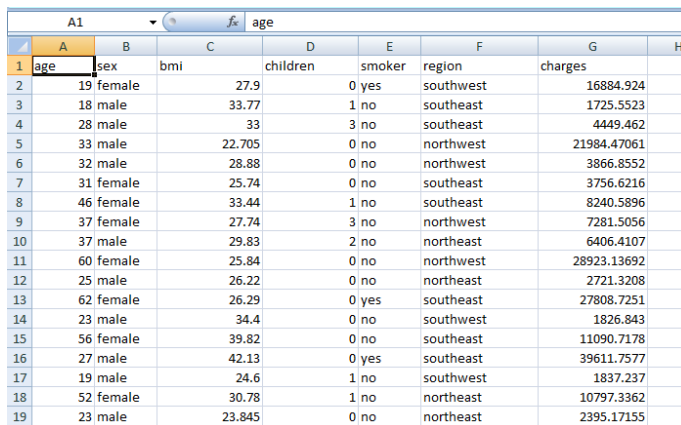
1. Introduction

This project involves the deployment of a web app using the Python and Flask framework. I used a dummy dataset (insurance.csv) along with a trained model (RandomForestRegressor from sklearn.ensemble) saved as model.pkl, which is to be used to predict Premium Health Insurance charges.

The necessary files and directories, including the Flask app (app.py), HTML templates, CSS file, was placed in their respective folders.

2. Data Information

The dataset in CSV (Viewed in Excel) showing the column headers and the first few rows are as follows:



	A	B	C	D	E	F	G	H
1	age	sex	bmi	children	smoker	region	charges	
2	19	female	27.9	0	yes	southwest	16884.924	
3	18	male	33.77	1	no	southeast	1725.5523	
4	28	male	33	3	no	southeast	4449.462	
5	33	male	22.705	0	no	northwest	21984.47061	
6	32	male	28.88	0	no	northwest	3866.8552	
7	31	female	25.74	0	no	southeast	3756.6216	
8	46	female	33.44	1	no	southeast	8240.5896	
9	37	female	27.74	3	no	northwest	7281.5056	
10	37	male	29.83	2	no	northeast	6406.4107	
11	60	female	25.84	0	no	northwest	28923.13692	
12	25	male	26.22	0	no	northeast	2721.3208	
13	62	female	26.29	0	yes	southeast	27808.7251	
14	23	male	34.4	0	no	southwest	1826.843	
15	56	female	39.82	0	no	southeast	11090.7178	
16	27	male	42.13	0	yes	southeast	39611.7577	
17	19	male	24.6	1	no	southwest	1837.237	
18	52	female	30.78	1	no	northeast	10797.3362	
19	23	male	23.845	0	no	northeast	2395.17155	

Figure 1.1: Dataset used for the App deployment in Excel Table (insurance.csv)

2.1. Attribute Information

The CSV consists of the following columns: 'age', 'sex', 'bmi', 'children', 'smoker', 'region' and 'charges'. There are six(6) feature variables and one(1) target variables. The below table describes the field properties and what it should be converted to before being passed to the Machine Learning model:

Table 2.2: Attribute Information (Feature variables)

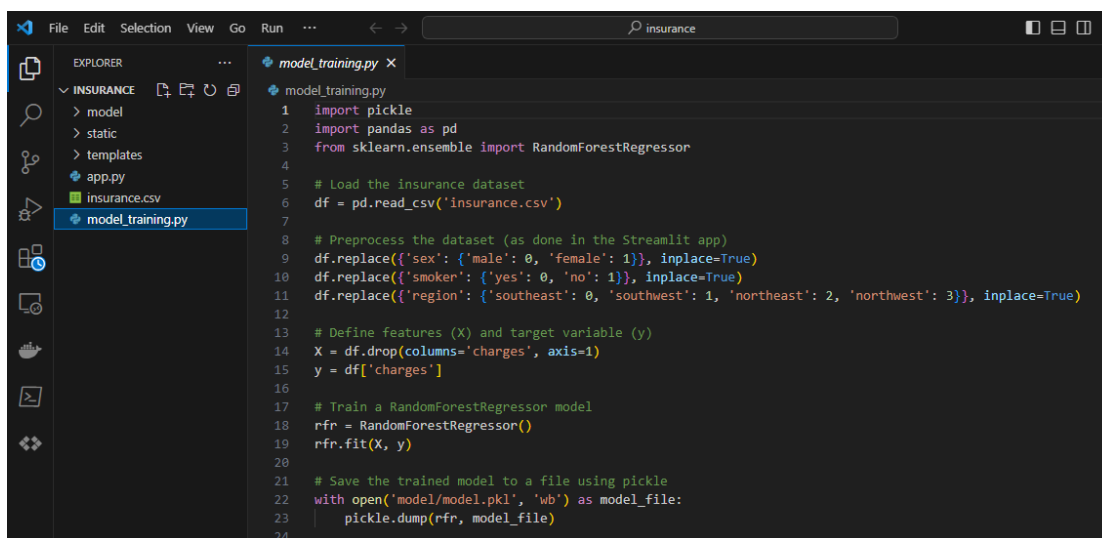
Attributes	Data-type
age	INT
sex	To be converted to INT – 'male':0, 'female'1
bmi	FLOAT
children	INT

smoker	To be converted to INT – ‘yes’:0, ‘No’:1
region	To be converted to INT – ‘southeast’:0, ‘southwest’:1, ‘northeast’:2, ‘northwest’:3

The Outcome variable (Target) is the ‘**charges**’ column which will be used to predict the insurance premium charges of the particular person.

3. Building the Model

The model was trained and saved into path (‘model/model.pkl’). A Python program was created and saved with the file named model_training.py to handle the model training and saving. Here is the code for the file.



```

1  import pickle
2  import pandas as pd
3  from sklearn.ensemble import RandomForestRegressor
4
5  # Load the insurance dataset
6  df = pd.read_csv('insurance.csv')
7
8  # Preprocess the dataset (as done in the Streamlit app)
9  df.replace({'sex': {'male': 0, 'female': 1}}, inplace=True)
10 df.replace({'smoker': {'yes': 0, 'no': 1}}, inplace=True)
11 df.replace({'region': {'southeast': 0, 'southwest': 1, 'northeast': 2, 'northwest': 3}}, inplace=True)
12
13 # Define features (X) and target variable (y)
14 X = df.drop(columns='charges', axis=1)
15 y = df['charges']
16
17 # Train a RandomForestRegressor model
18 rfr = RandomForestRegressor()
19 rfr.fit(X, y)
20
21 # Save the trained model to a file using pickle
22 with open('model/model.pkl', 'wb') as model_file:
23     pickle.dump(rfr, model_file)
24

```

Figure 3.1: model_training.py file used for training and saving the model

4. Turning Model into Flask Framework

4.1. Predict.html

I created an HTML template for the web app. Here's a simple "predict.html" template that was created and saved in the template folder.

```
Cloud Code - Release Notes  predict.html X
templates > predict.html > ...
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">
5  </head>
6  <body>
7      <h1>Health Insurance Premium Predictor</h1>
8      
9
10     <form method="POST" action="/predict">
11         <label for="age">Age:</label>
12         <input type="number" id="age" name="age" required>
13
14         <label for="sex">Sex:</label>
15         <select id="sex" name="sex">
16             <option value="0">Male</option>
17             <option value="1">Female</option>
18         </select>
19
20         <label for="bmi">BMI:</label>
21         <input type="number" id="bmi" name="bmi" step="0.01" min="0" required>
22
23         <label for="children">Number of Children:</label>
24         <input type="number" id="children" name="children" step="1" min="0" required>
25
26         <label for="smoker">Do you smoke?</label>
27         <select id="smoker" name="smoker">
28             <option value="0">Yes</option>
29             <option value="1">No</option>
30         </select>
31
32         <label for="region">Region:</label>
33         <select id="region" name="region">
34             <option value="0">SouthEast</option>
35             <option value="1">SouthWest</option>
36             <option value="2">NorthEast</option>
37             <option value="3">NorthWest</option>
38         </select>
39
40         <button type="submit">Predict</button>
41     </form>
```

Figure 4.1: predict.html Template for interfacing with the user

4.2. Styles.css

I created a CSS file named style.css in a "static" folder. Here, I defined the styling for the HTML template in this file.

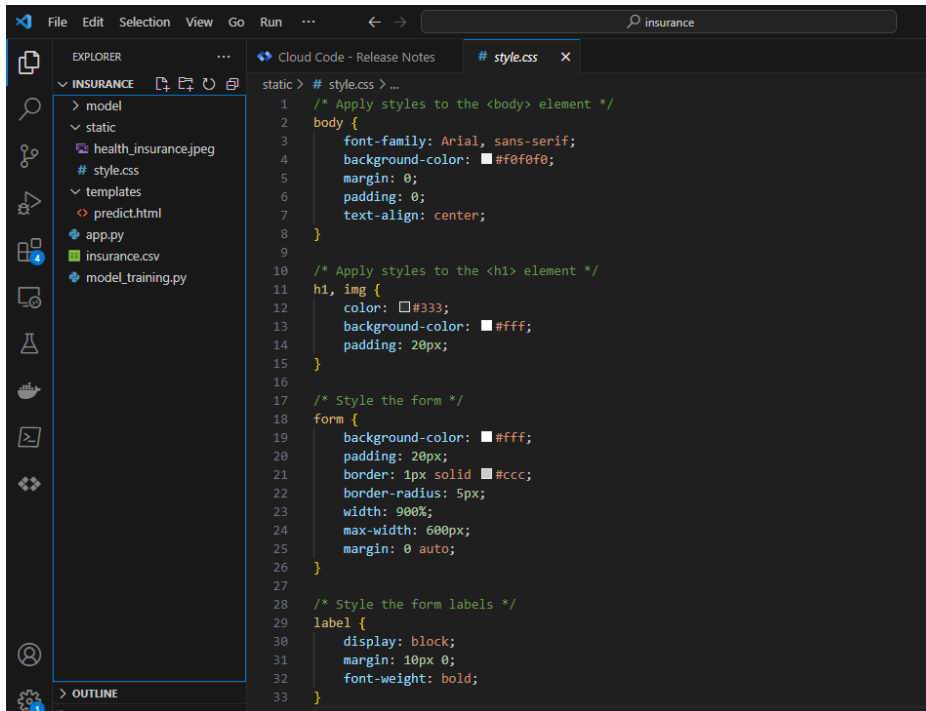


Figure 4.2: style.css used for styling the HTML template

4.3. App.py

I created a file named app.py for the Flask application, which ties everything together

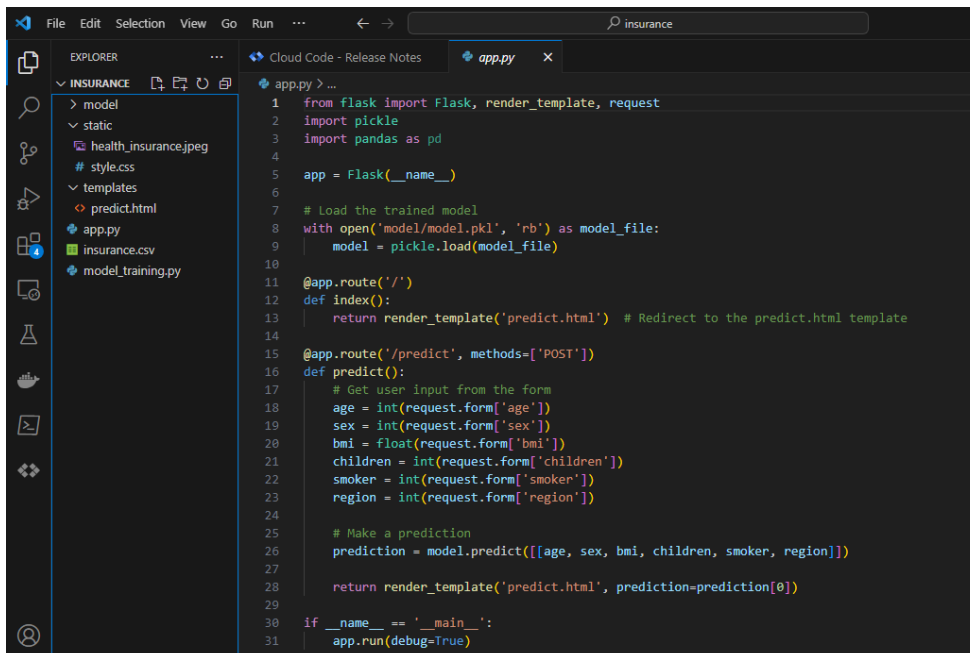


Figure 4.3: style.css used for styling the HTML template

4.4. Running the App

To run the Flask app, make sure you have Flask installed (pip install Flask). Then, run the app.py file. You can access the app in your web browser

```
C:\Windows\System32\cmd.exe - python app.py
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user\Documents\Okeoma\Internship program\Data Glacier Virtual Internship\Github\Week4\insurance>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 watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 132-453-836
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figure 4.4: Running the App via Command prompt



Figure 4.5: App running locally in the browser

The image shows a web application interface for predicting insurance premium. It features a central form with several input fields and a 'Predict' button. Below the form, the predicted premium is displayed. The background is a light gray with a faint 'Activate Windows' watermark.

Age:
25

Sex:
Male

BMI:
35.6

Number of Children:
3

Do you smoke?
Yes

Region:
NorthEast

Predict

Predicted Premium: \$7448.702104199998

Activate Windows
Go to Settings to activate Windows.

Figure 4.6: Showing the predicted results derived from the model