Data Glacier

Data Science Intern at Data Glacier

Week 5: Deployment of Flask App to AWS Cloud

Name: Okeoma Ihunwo

Batch Code: LISUM26

Submission Date: 10th November 2023

Submitted to: Data Glacier

Table of Contents:

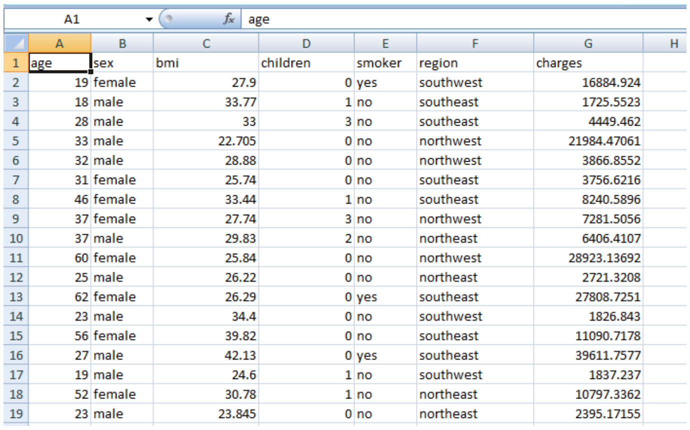
1. Introduction

This project involves the deployment of a web app to AWS cloud 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:



A1 | age

	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)

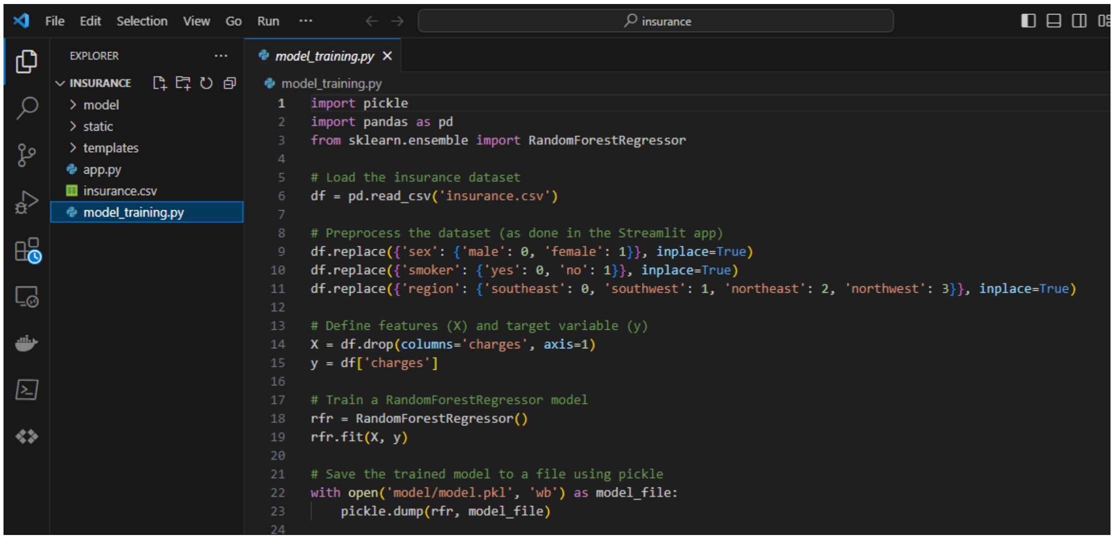
Attributes	Data-type
age	INT
sex	To be converted to INT – ‘male’:0, ‘female’1
bmi	FLOAT
children	INT

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.



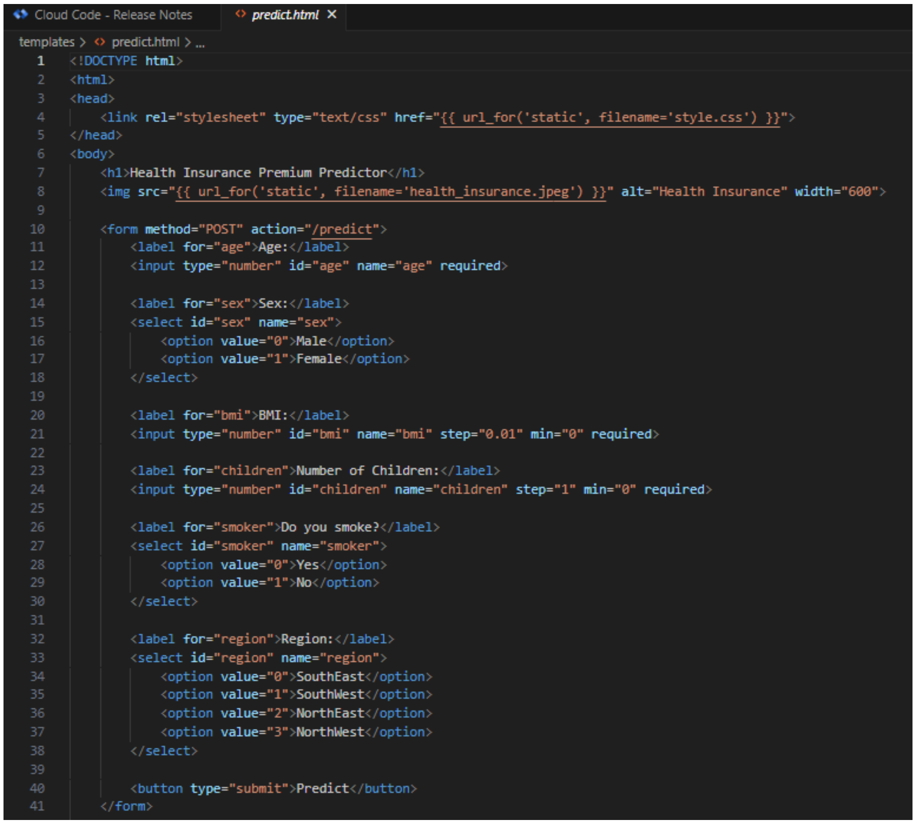
```python
import pickle
import pandas as pd
from sklearn.ensemble import RandomForestRegressor

# Load the insurance dataset
df = pd.read_csv('insurance.csv')

# Preprocess the dataset (as done in the Streamlit app)
df.replace({'sex': {'male': 0, 'female': 1}}, inplace=True)
df.replace({'smoker': {'yes': 0, 'no': 1}}, inplace=True)
df.replace({'region': {'southeast': 0, 'southwest': 1, 'northeast': 2, 'northwest': 3}}, inplace=True)

# Define features (X) and target variable (y)
X = df.drop(columns='charges', axis=1)
y = df['charges']

# Train a RandomForestRegressor model
rfr = RandomForestRegressor()
rfr.fit(X, y)

# Save the trained model to a file using pickle
with open('model/model.pkl', 'wb') as model_file:
    pickle.dump(rfr, model_file)
```

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

4. Create Flask App

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.

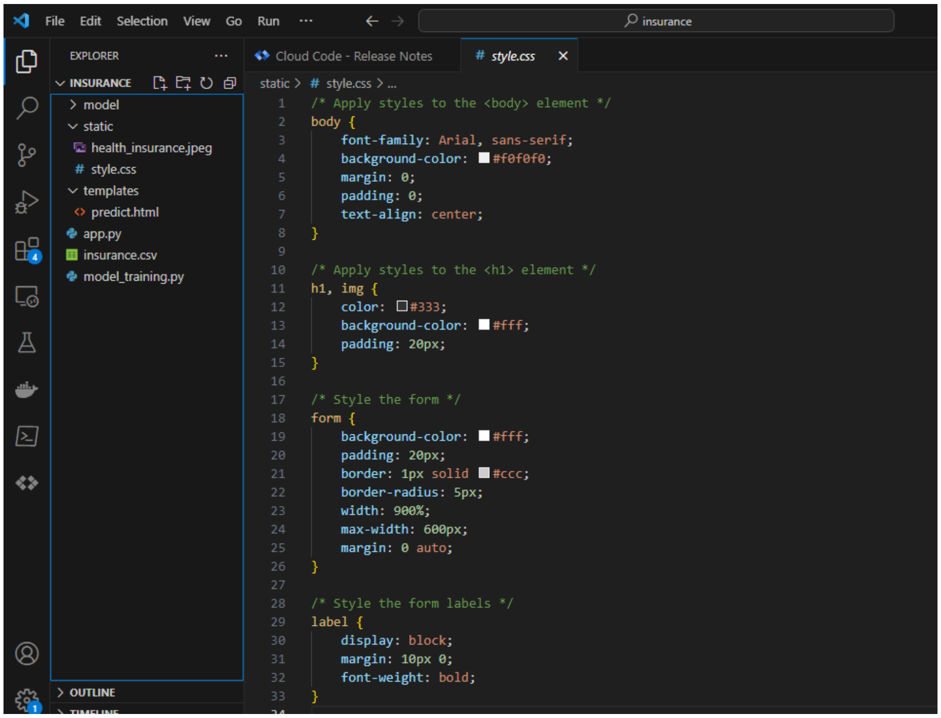


```html
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <h1>Health Insurance Premium Predictor</h1>
    <img src="{{ url_for('static', filename='health_insurance.jpeg') }}" alt="Health Insurance" width="600">

    <form method="POST" action="/predict">
        <label for="age">Age:</label>
        <input type="number" id="age" name="age" required>

        <label for="sex">Sex:</label>
        <select id="sex" name="sex">
            <option value="0">Male</option>
            <option value="1">Female</option>
        </select>

        <label for="bmi">BMI:</label>
        <input type="number" id="bmi" name="bmi" step="0.01" min="0" required>

        <label for="children">Number of Children:</label>
        <input type="number" id="children" name="children" step="1" min="0" required>

        <label for="smoker">Do you smoke?</label>
        <select id="smoker" name="smoker">
            <option value="0">Yes</option>
            <option value="1">No</option>
        </select>

        <label for="region">Region:</label>
        <select id="region" name="region">
            <option value="0">SouthEast</option>
            <option value="1">SouthWest</option>
            <option value="2">NorthEast</option>
            <option value="3">NorthWest</option>
        </select>

        <button type="submit">Predict</button>
    </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.

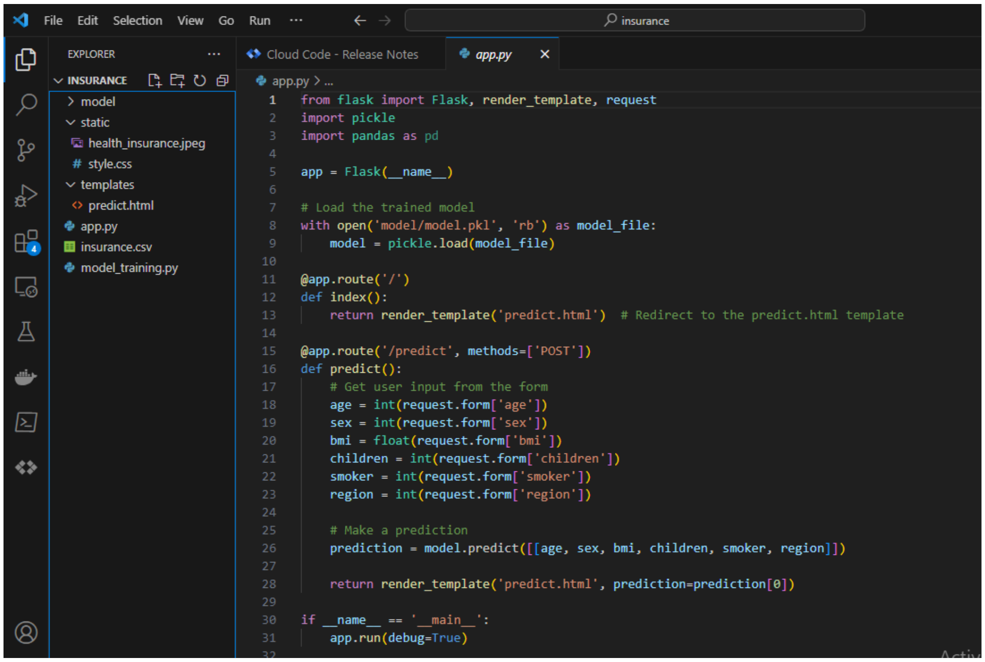


```css
/* Apply styles to the <body> element */
body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    margin: 0;
    padding: 0;
    text-align: center;
}

/* Apply styles to the <h1> element */
h1, img {
    color: #333;
    background-color: #fff;
    padding: 20px;
}

/* Style the form */
form {
    background-color: #fff;
    padding: 20px;
    border: 1px solid #ccc;
    border-radius: 5px;
    width: 900%;
    max-width: 600px;
    margin: 0 auto;
}

/* Style the form labels */
label {
    display: block;
    margin: 10px 0;
    font-weight: bold;
}
```

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

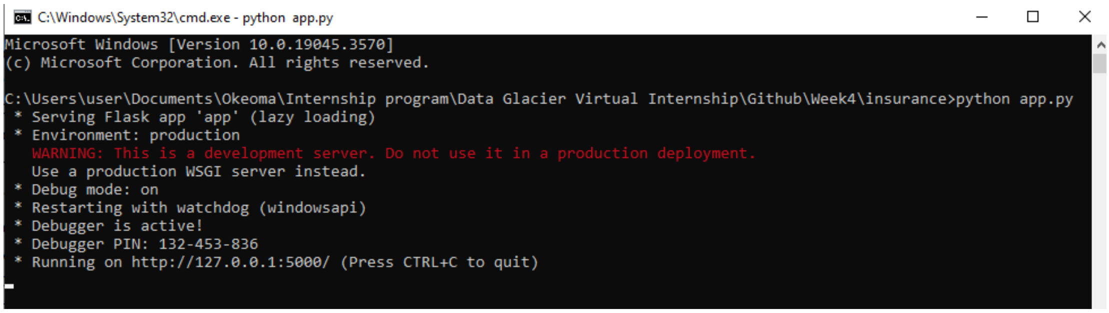


```python
from flask import Flask, render_template, request
import pickle
import pandas as pd

app = Flask(__name__)

# Load the trained model
with open('model/model.pkl', 'rb') as model_file:
    model = pickle.load(model_file)

@app.route('/')
def index():
    return render_template('predict.html')  # Redirect to the predict.html template

@app.route('/predict', methods=['POST'])
def predict():
    # Get user input from the form
    age = int(request.form['age'])
    sex = int(request.form['sex'])
    bmi = float(request.form['bmi'])
    children = int(request.form['children'])
    smoker = int(request.form['smoker'])
    region = int(request.form['region'])

    # Make a prediction
    prediction = model.predict([[age, sex, bmi, children, smoker, region]])

    return render_template('predict.html', prediction=prediction[0])

if __name__ == '__main__':
    app.run(debug=True)
```

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

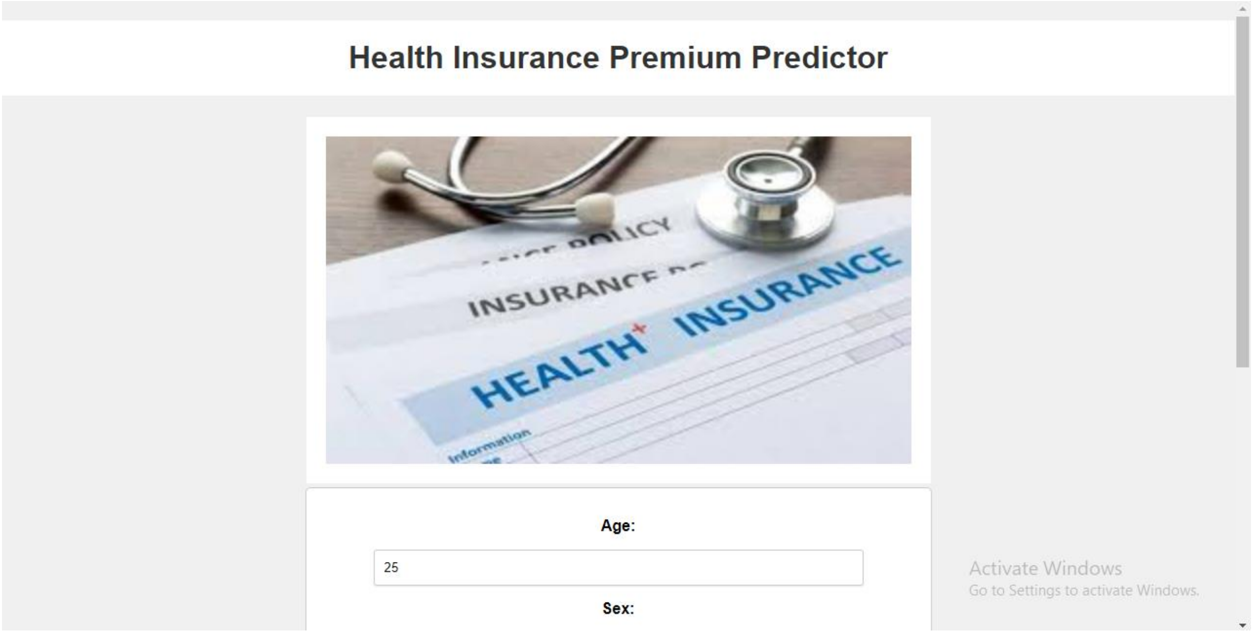
4.4. Running the App locally

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



```
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



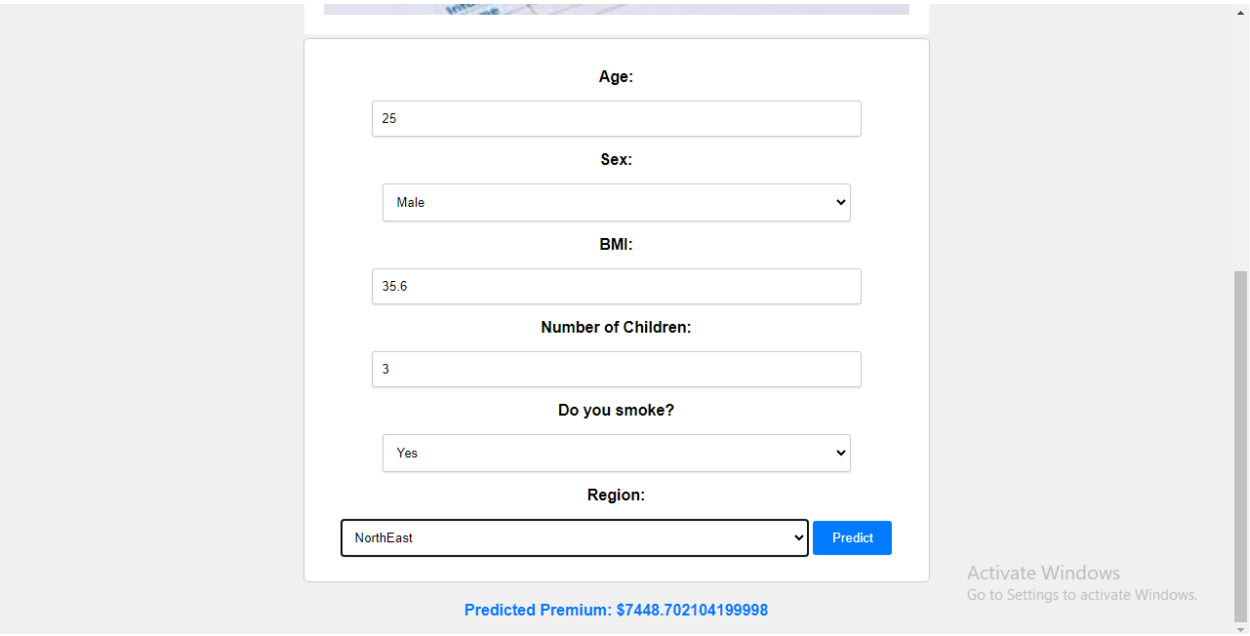
Health Insurance Premium Predictor

Age:

25

Sex:

Figure 4.5: App running locally in the browser



Age:

25

Sex:

Male

BMI:

35.6

Number of Children:

3

Do you smoke?

Yes

Region:

NorthEast

Predict

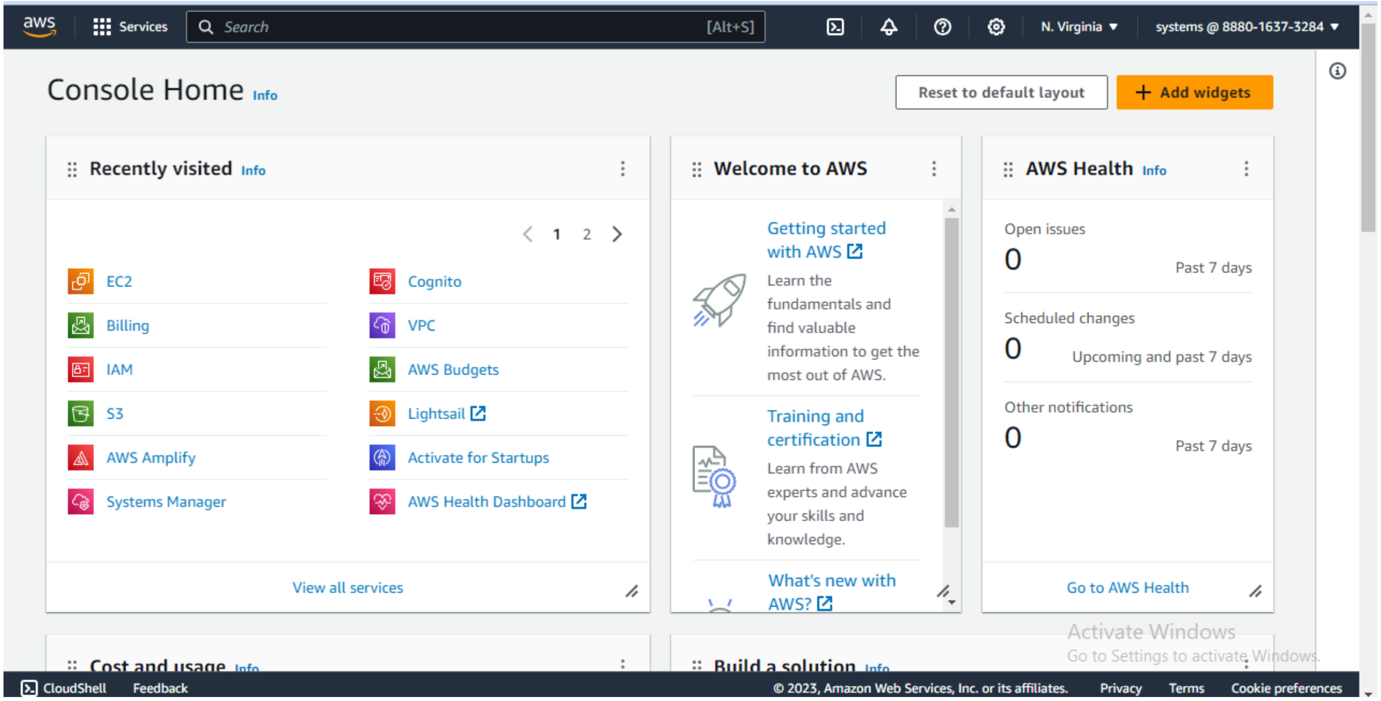
Predicted Premium: $7448.702104199998

Figure 4.6: Showing the predicted results derived from the model

5. Create & Connect to AWS EC2

5.1. Log-in to AWS

Create or Log-in to AWS account.



Console Home Info

Reset to default layout

+ Add widgets

Recently visited Info

EC2	Cognito
Billing	VPC
IAM	AWS Budgets
S3	Lightsail
AWS Amplify	Activate for Startups
Systems Manager	AWS Health Dashboard

View all services

Welcome to AWS

Getting started with AWS

Learn the fundamentals and find valuable information to get the most out of AWS.

Training and certification

Learn from AWS experts and advance your skills and knowledge.

What's new with AWS?

AWS Health Info

Open issues

0 Past 7 days

Scheduled changes

0 Upcoming and past 7 days

Other notifications

0 Past 7 days

Go to AWS Health

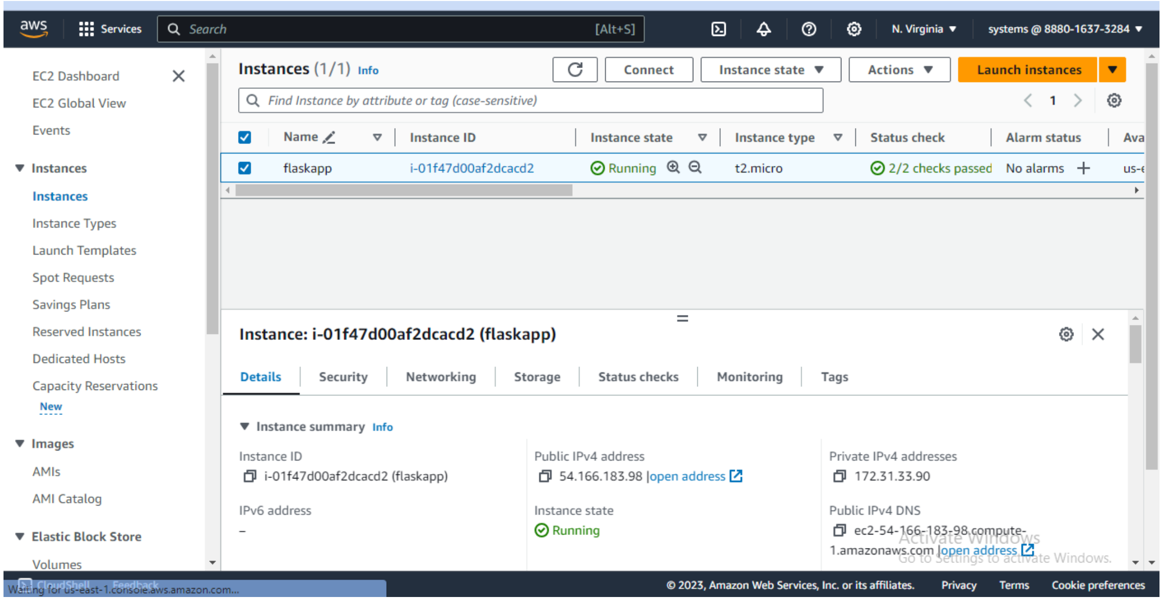
Cost and usage Info

Build a solution Info

Figure 5.1: AWS Console

5.2. Create EC2 Instance

Create a free tier EC2 (Ubuntu Server 20.04 LTS (HVM), SSD Volume Type) instance and download the key pair for accessing the server remotely using Putty via windows system.



Instances (1/1) Info

☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Ava
☑	flaskapp	i-01f47d00af2dcacd2	✓ Running	t2.micro	✓ 2/2 checks passed	No alarms	us-e

Instance: i-01f47d00af2dcacd2 (flaskapp)

Instance summary Info

Instance ID
i-01f47d00af2dcacd2 (flaskapp)

Public IPv4 address
54.166.183.98 |open address

Private IPv4 addresses
172.31.33.90

IPv6 address
–

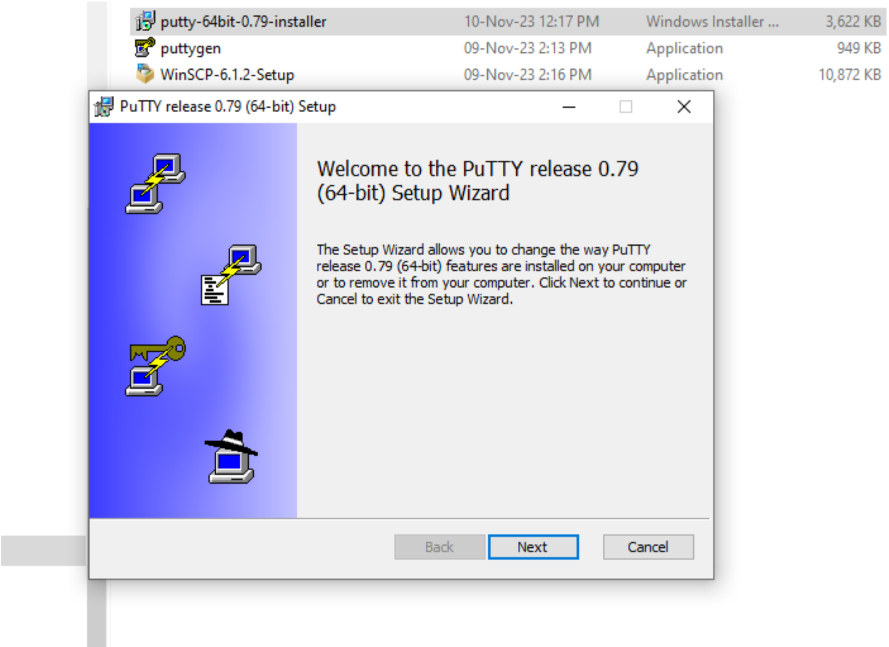
Instance state
Running

Public IPv4 DNS
ec2-54-166-183-98.compute-1.amazonaws.com |open address

Figure 5.2: New AWS EC2 Instance

5.3. Download & Install Puttygen, Wincp & Putty

Download and install the updated version of Puttygen, Wincp and Putty.



putty-64bit-0.79-installer	10-Nov-23 12:17 PM	Windows Installer ...	3,622 KB
puttygen	09-Nov-23 2:13 PM	Application	949 KB
WinSCP-6.1.2-Setup	09-Nov-23 2:16 PM	Application	10,872 KB

PuTTY release 0.79 (64-bit) Setup

Welcome to the PuTTY release 0.79 (64-bit) Setup Wizard

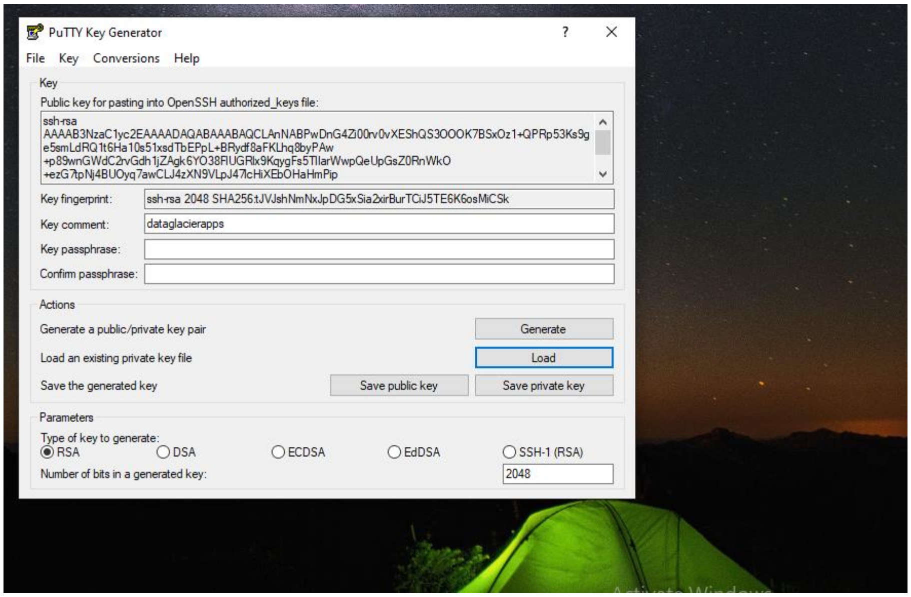
The Setup Wizard allows you to change the way PuTTY release 0.79 (64-bit) features are installed on your computer or to remove it from your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Figure 5.3: Installation of Puttygen, Wincp and Putty

5.4. Connect to EC2 Instance

You can connect to the EC2 instance by following the below steps:

5.4.1. Puttygen -> generate private key



Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCLAnNABPwDnG4Zi00rv0vXEShQS3OOOK7BSxOz1+QPRp53Ks9g
e5smLdRQ1t6Ha10s51xsdTbEPpL+BRydf8aFKLhq8byPAw
+p89wnGWdC2rvGdh1jZAgk6YO38FlUGRlx9KqygFs5TIlarWwpQeUpGsZ0RnWkO
+ezG7tpNj4BUOyq7awCLJ4zXN9VLpJ47lcHiXEbOHaHmPip
```

Key fingerprint: ssh-rsa 2048 SHA256:tJVJshNmNxJpDG5xSia2xirBurTCiJ5TE6K6osMiCSk

Key comment: dataglacierapps

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair

Load an existing private key file

Save the generated key

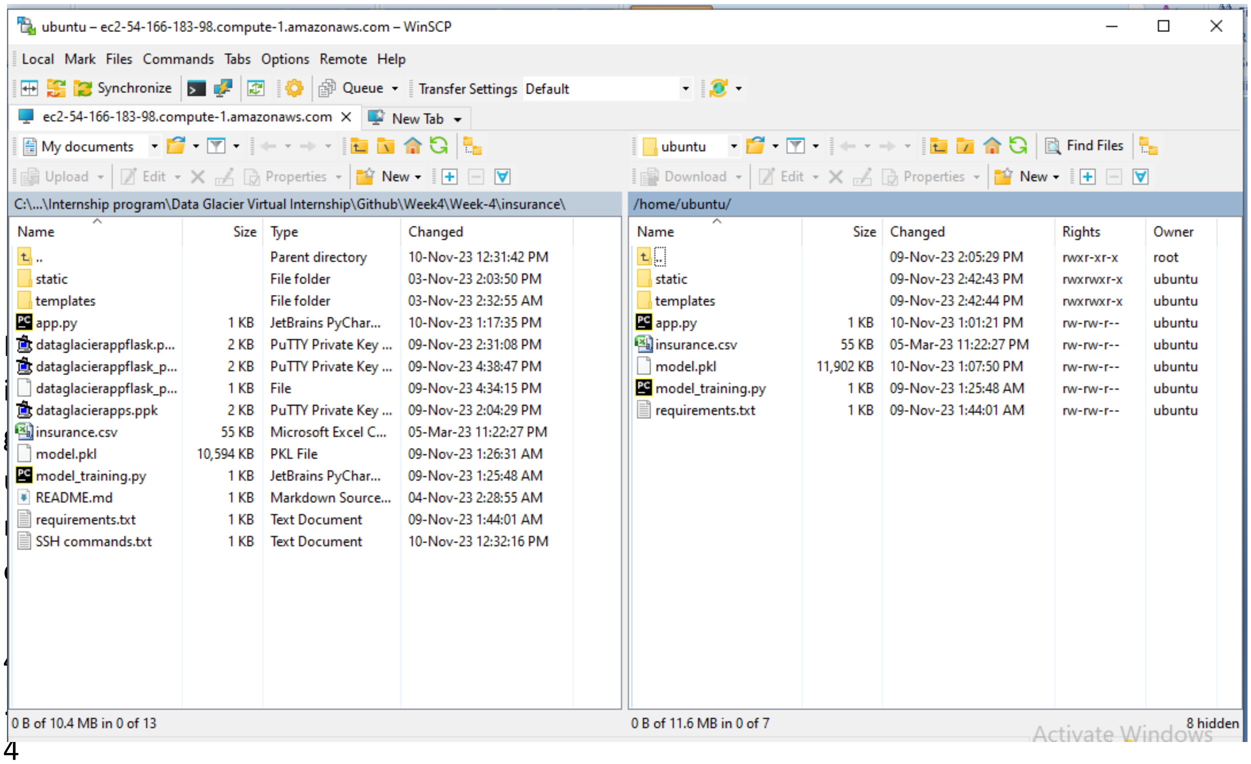
Parameters

Type of key to generate:
● RSA ○ DSA ○ ECDSA ○ EdDSA ○ SSH-1 (RSA)

Number of bits in a generated key: 2048

Figure 5.4.1: Generating Private key with Puttygen

5.4.2. wincp -> copy all files to Ubuntu server


ubuntu – ec2-54-166-183-98.compute-1.amazonaws.com – WinSCP
Local Mark Files Commands Tabs Options Remote Help
Synchronize
Queue
Transfer Settings Default
ec2-54-166-183-98.compute-1.amazonaws.com
New Tab
My documents
Upload
Edit
Properties
New
ubuntu
Find Files
Download
Edit
Properties
New
Activate Windows

C:\...\Internship program\Data Glacier Virtual Internship\Github\Week4\Week-4\insurance\

Name	Size	Type	Changed
..		Parent directory	10-Nov-23 12:31:42 PM
static		File folder	03-Nov-23 2:03:50 PM
templates		File folder	03-Nov-23 2:32:55 AM
app.py	1 KB	JetBrains PyChar...	10-Nov-23 1:17:35 PM
dataglacierappflask.p...	2 KB	PuTTY Private Key ...	09-Nov-23 2:31:08 PM
dataglacierappflask_p...	2 KB	PuTTY Private Key ...	09-Nov-23 4:38:47 PM
dataglacierappflask_p...	1 KB	File	09-Nov-23 4:34:15 PM
dataglacierapps.ppk	2 KB	PuTTY Private Key ...	09-Nov-23 2:04:29 PM
insurance.csv	55 KB	Microsoft Excel C...	05-Mar-23 11:22:27 PM
model.pkl	10,594 KB	PKL File	09-Nov-23 1:26:31 AM
model_training.py	1 KB	JetBrains PyChar...	09-Nov-23 1:25:48 AM
README.md	1 KB	Markdown Source...	04-Nov-23 2:28:55 AM
requirements.txt	1 KB	Text Document	09-Nov-23 1:44:01 AM
SSH commands.txt	1 KB	Text Document	10-Nov-23 12:32:16 PM

0 B of 10.4 MB in 0 of 13

/home/ubuntu/

Name	Size	Changed	Rights	Owner
..		09-Nov-23 2:05:29 PM	rwxr-xr-x	root
static		09-Nov-23 2:42:43 PM	rwxrwxr-x	ubuntu
templates		09-Nov-23 2:42:44 PM	rwxrwxr-x	ubuntu
app.py	1 KB	10-Nov-23 1:01:21 PM	rw-rw-r--	ubuntu
insurance.csv	55 KB	05-Mar-23 11:22:27 PM	rw-rw-r--	ubuntu
model.pkl	11,902 KB	10-Nov-23 1:07:50 PM	rw-rw-r--	ubuntu
model_training.py	1 KB	09-Nov-23 1:25:48 AM	rw-rw-r--	ubuntu
requirements.txt	1 KB	09-Nov-23 1:44:01 AM	rw-rw-r--	ubuntu

0 B of 11.6 MB in 0 of 7

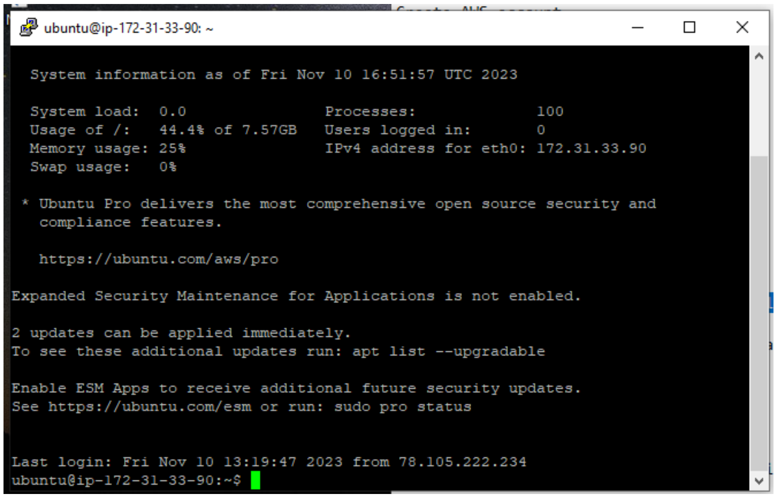
8 hidden

4

Figure 5.4.2: Copying Flacks app files into Ubuntu server

5.4.3. Putty --> connect through Putty

Connect via SSH into the EC2 instance by authenticating with the Private key using Putty.



```
  System information as of Fri Nov 10 16:51:57 UTC 2023

  System load:  0.0               Processes:             100
  Usage of /:   44.4% of 7.57GB   Users logged in:       0
  Memory usage: 25%               IPv4 address for eth0: 172.31.33.90
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

2 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Fri Nov 10 13:19:47 2023 from 78.105.222.234
ubuntu@ip-172-31-33-90:~$
```

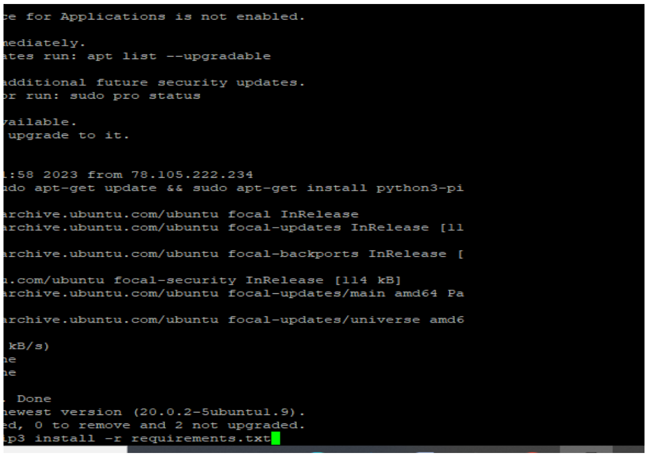
Figure 5.4.3: Connect via SSH into the EC2 instance using Putty

6. Deploy Flask App on AWS EC2

6.1. Update & Install Required Libraries

After you must have logged in to the EC2 instance, it is required to update PIP and install the required libraries using the following commands:

Command 1: sudo apt-get update && sudo apt-get install python3-pip and **Command 2**: pip3 install -r requirements.txt

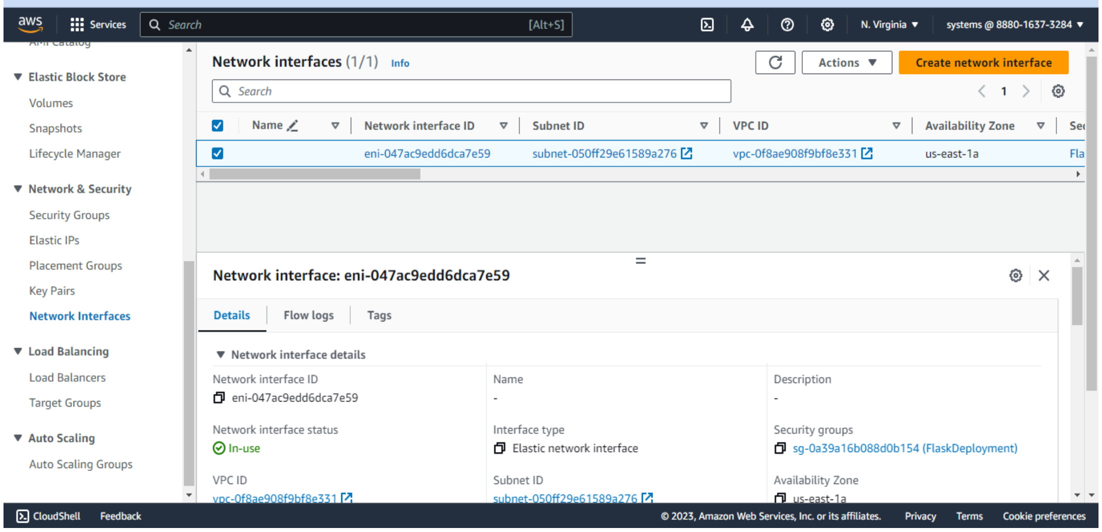


```
ce for Applications is not enabled.

mediately.
ates run: apt list --upgradable

additional future security updates.
or run: sudo pro status

vailable.
 upgrade to it.


1:58 2023 from 78.105.222.234
udo apt-get update && sudo apt-get install python3-pi

archive.ubuntu.com/ubuntu focal InRelease
archive.ubuntu.com/ubuntu focal-updates InRelease [11

archive.ubuntu.com/ubuntu focal-backports InRelease [

1.com/ubuntu focal-security InRelease [114 kB]
archive.ubuntu.com/ubuntu focal-updates/main amd64 Pa

archive.ubuntu.com/ubuntu focal-updates/universe amd6

 kB/s)
ne
ne

. Done
newest version (20.0.2-5ubuntu1.9).
ed, 0 to remove and 2 not upgraded.
ip3 install -r requirements.txt
```

Figure 6.1: Showing the commands for updating and installing libraries

6.2. Create Security Group & Assign to EC2 Instance

A new security group (FlaskDeployment) was created to allow all users to connect to the Flask app via the internet



Network interfaces (1/1) Info

☑	Name	Network interface ID	Subnet ID	VPC ID	Availability Zone	Se
☑		eni-047ac9edd6dca7e59	subnet-050ff29e61589a276	vpc-0f8ae908f9bf8e331	us-east-1a	Fla

Network interface: eni-047ac9edd6dca7e59

▼ Network interface details

Network interface ID eni-047ac9edd6dca7e59	Name -	Description -
Network interface status In-use	Interface type Elastic network interface	Security groups sg-0a39a16b088d0b154 (FlaskDeployment)
VPC ID vpc-0f8ae908f9bf8e331	Subnet ID subnet-050ff29e61589a276	Availability Zone us-east-1a

Figure 6.2: Security Group created and attached to the EC2 instance

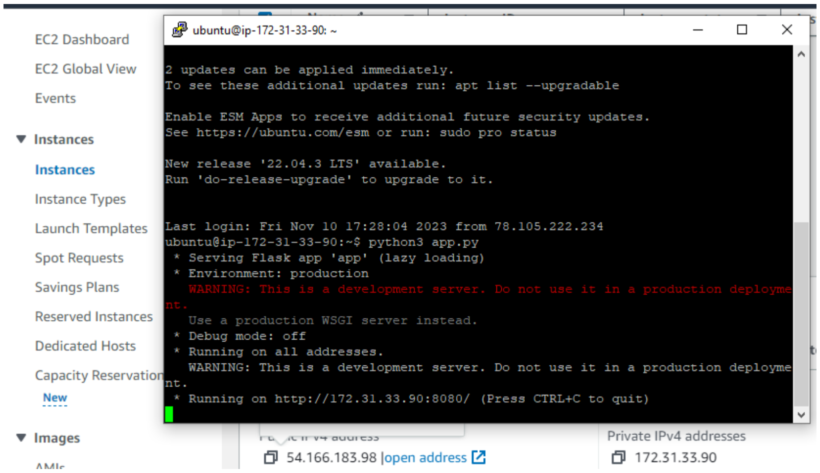
6.3. Edit Host & Port in Flask App

To enable the connection via internet, the host and port numbers was added to the app.py file while the app.run(debug=True) code was commented:

```
app.run(host="0.0.0.0", port=8080)
#app.run(debug=True)
```

6.4. Running the App in AWS

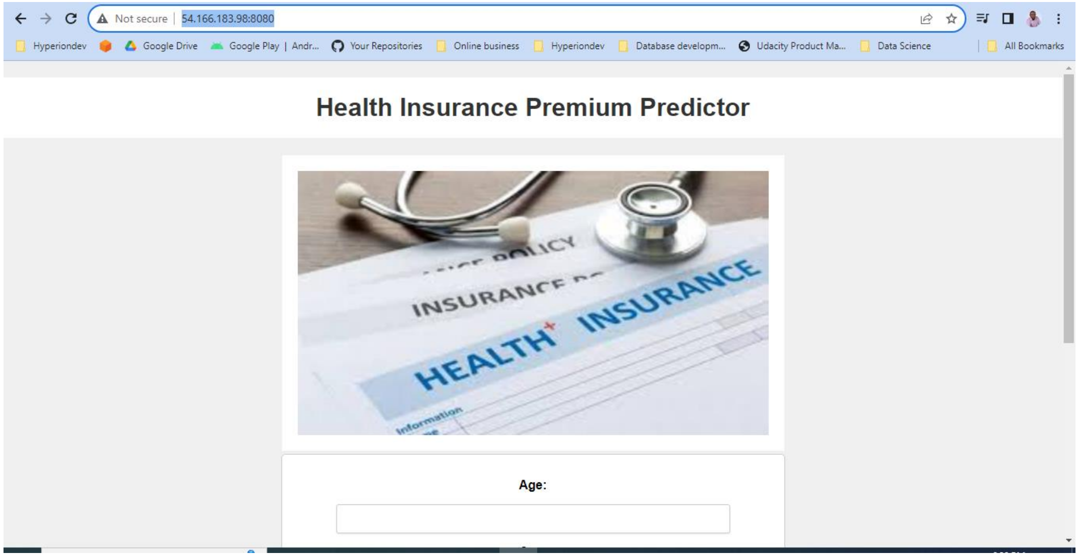
Finally, run the app in AWS using the command “python3 app.py”



```
2 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Fri Nov 10 17:28:04 2023 from 78.105.222.234
ubuntu@ip-172-31-33-90:~$ python3 app.py
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployme
nt.
 * Running on http://172.31.33.90:8080/ (Press CTRL+C to quit)
```

Figure 6.3: Running the Flask App from in AWS using Putty

Open the browser and paste the EC2 instance Public IP address in the browser with colon (“:”) and port “8080” following, ie: http://54.166.183.98:8080/



Health Insurance Premium Predictor

Age:

Figure 6.4: Accessing the deployed Flask App from a browser