

Flask Deployment

Name: Shwetha Basavanagowda

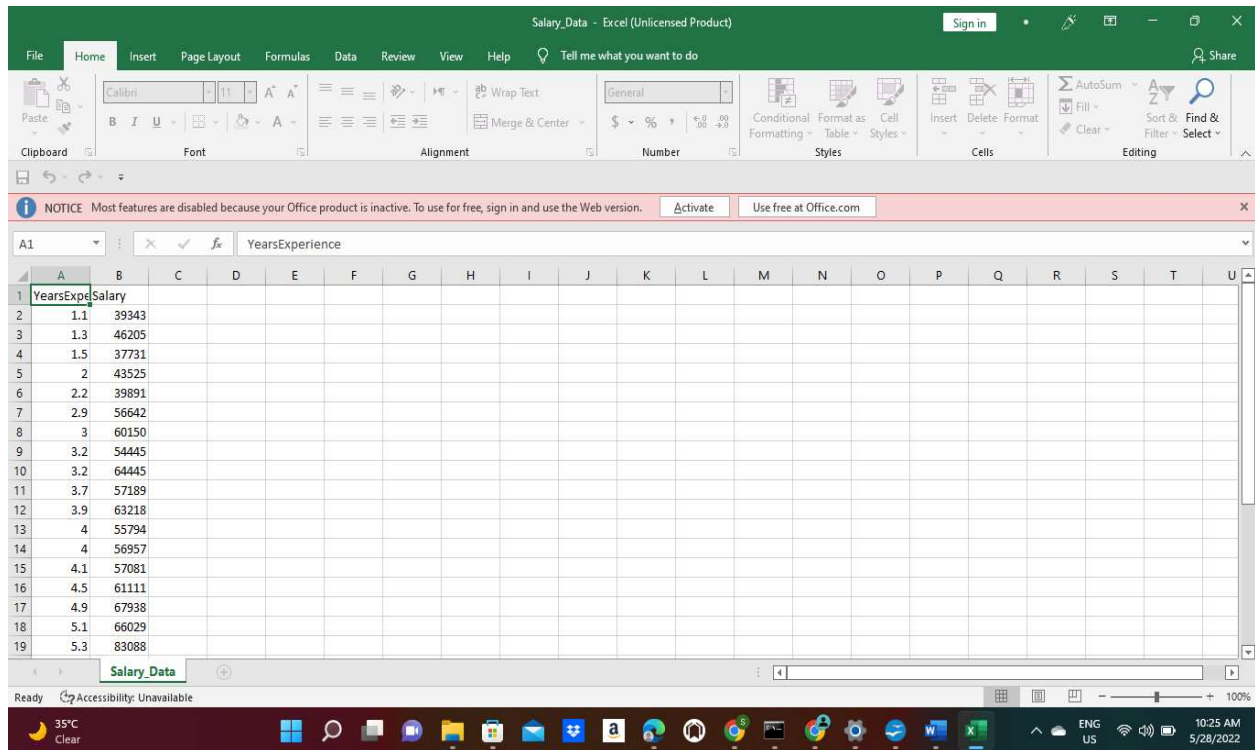
Batch Code: LISUM09

Submission Date: 28/05/2022

Submitted To: Data Glacier Team

Steps and Screenshots of Deployment

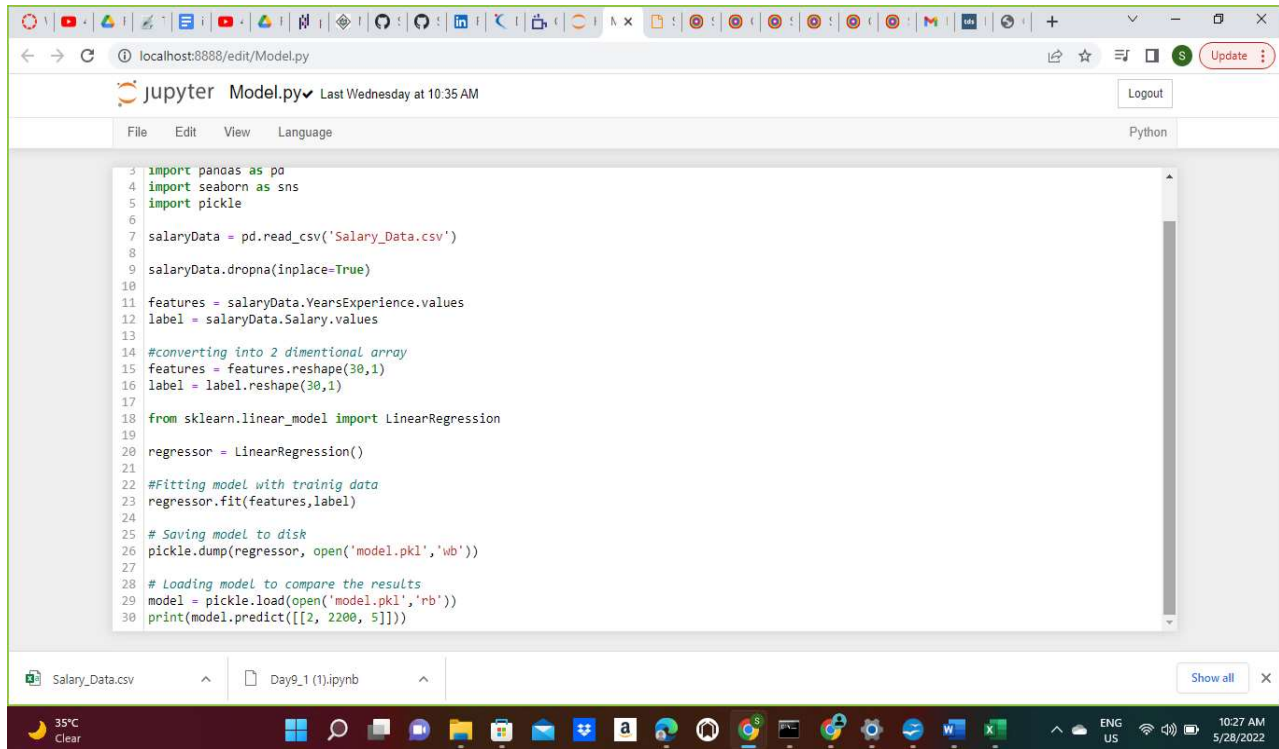
1. The Toy data I have taken for building a prediction model and deployment in flask is a Salary_Data which has Number of years of experience and corresponding salary data which looks as below.



The screenshot shows an Excel spreadsheet with the following data:

	YearsExperience	Salary
1	1.1	39343
2	1.3	46205
3	1.5	37731
4	2	43525
5	2.2	39891
6	2.9	56642
7	3	60150
8	3.2	54445
9	3.2	64445
10	3.7	57189
11	3.9	63218
12	4	55794
13	4	56957
14	4.1	57081
15	4.5	61111
16	4.9	67938
17	5.1	66029
18	5.3	83088

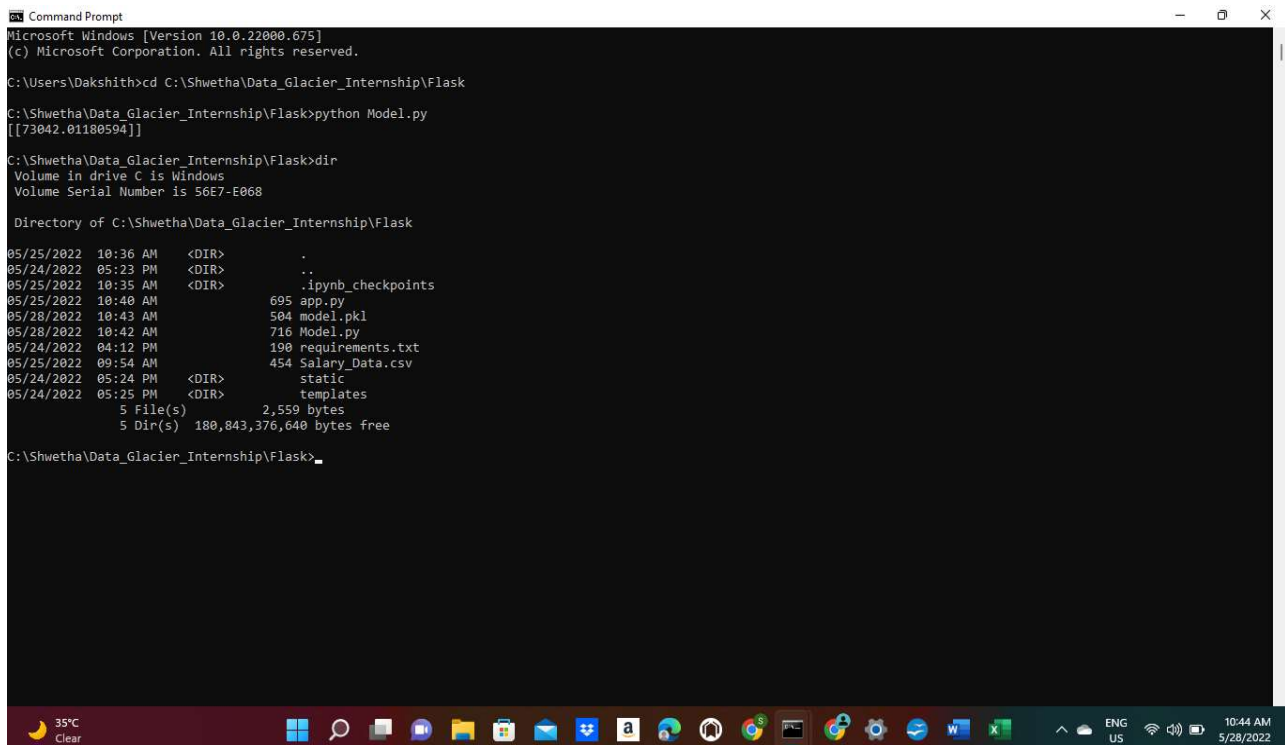
2. Build a model to predict the Salary depending on the Number of Years of experience and save the model.



The screenshot shows a Jupyter Notebook interface in a web browser. The notebook is titled "Model.py" and shows a Python script for building a linear regression model and saving it as a pickle file. The script includes imports for pandas, seaborn, and pickle, reads a CSV file, drops missing values, reshapes the data, and uses sklearn's LinearRegression to fit the model. It then saves the model to a file named "model.pkl" and loads it back to compare results.

```
3 import pandas as pd
4 import seaborn as sns
5 import pickle
6
7 salaryData = pd.read_csv('Salary_Data.csv')
8
9 salaryData.dropna(inplace=True)
10
11 features = salaryData.YearsExperience.values
12 label = salaryData.Salary.values
13
14 #converting into 2 dimensional array
15 features = features.reshape(30,1)
16 label = label.reshape(30,1)
17
18 from sklearn.linear_model import LinearRegression
19
20 regressor = LinearRegression()
21
22 #Fitting model with training data
23 regressor.fit(features,label)
24
25 # Saving model to disk
26 pickle.dump(regressor, open('model.pkl','wb'))
27
28 # Loading model to compare the results
29 model = pickle.load(open('model.pkl','rb'))
30 print(model.predict([[2, 2200, 5]]))
```

3. Generate the Pickle file Model.pkl by running the above file Model.py



The screenshot shows a Windows Command Prompt window. The user has navigated to the directory C:\Shwetha\Data_Glacier_Internship\Flask and executed the command python Model.py. The output shows the directory listing for C:\Shwetha\Data_Glacier_Internship\Flask, which includes files like app.py, model.pkl, Model.py, requirements.txt, Salary_Data.csv, static, and templates. The directory listing also shows the file sizes and the free space on the drive.

```
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dakshith>cd C:\Shwetha\Data_Glacier_Internship\Flask

C:\Shwetha\Data_Glacier_Internship\Flask>python Model.py
[[73042.01180594]]

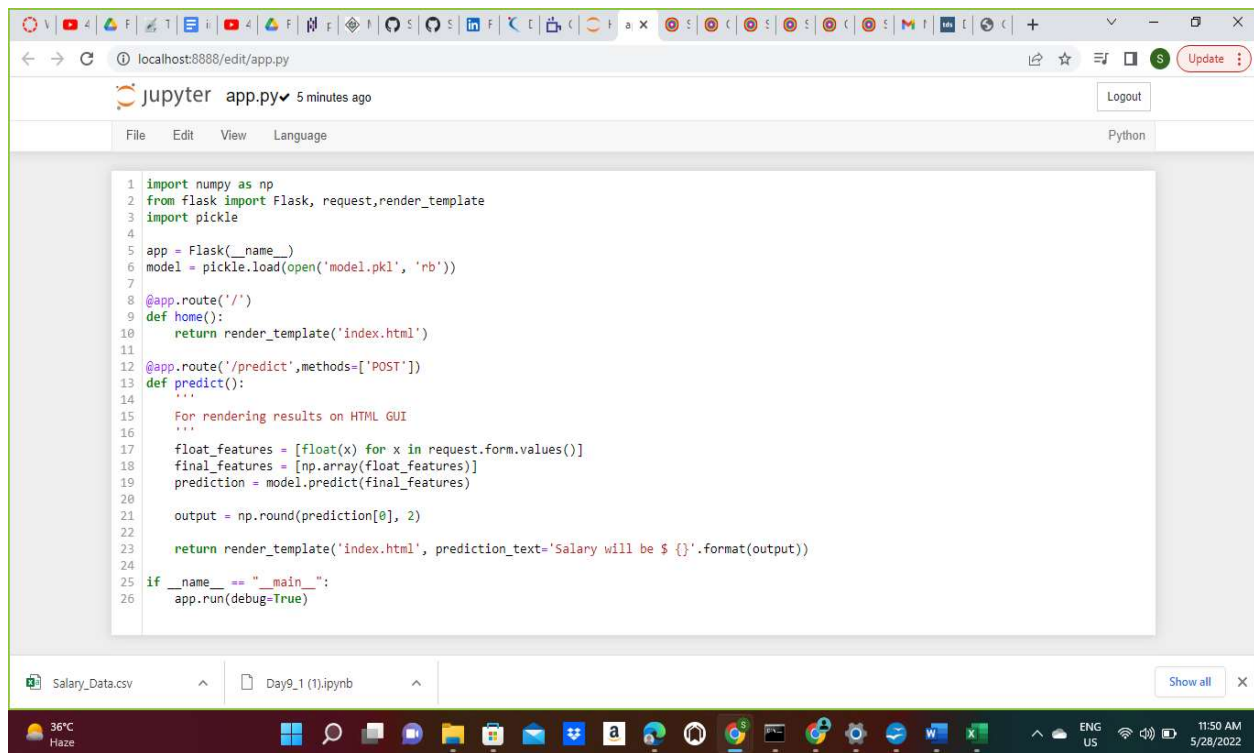
C:\Shwetha\Data_Glacier_Internship\Flask>dir
Volume in drive C is Windows
Volume Serial Number is 56E7-E068

Directory of C:\Shwetha\Data_Glacier_Internship\Flask

05/25/2022  10:36 AM    <DIR>        .
05/24/2022  05:23 PM    <DIR>        ..
05/25/2022  10:35 AM    <DIR>        .ipynb_checkpoints
05/25/2022  10:40 AM             695 app.py
05/28/2022  10:43 AM             504 model.pkl
05/28/2022  10:42 AM             716 Model.py
05/24/2022  04:12 PM             190 requirements.txt
05/25/2022  09:54 AM             454 Salary_Data.csv
05/24/2022  05:24 PM    <DIR>        static
05/24/2022  05:25 PM    <DIR>        templates
               5 File(s)          2,559 bytes
               5 Dir(s)  180,843,376,640 bytes free

C:\Shwetha\Data_Glacier_Internship\Flask>
```

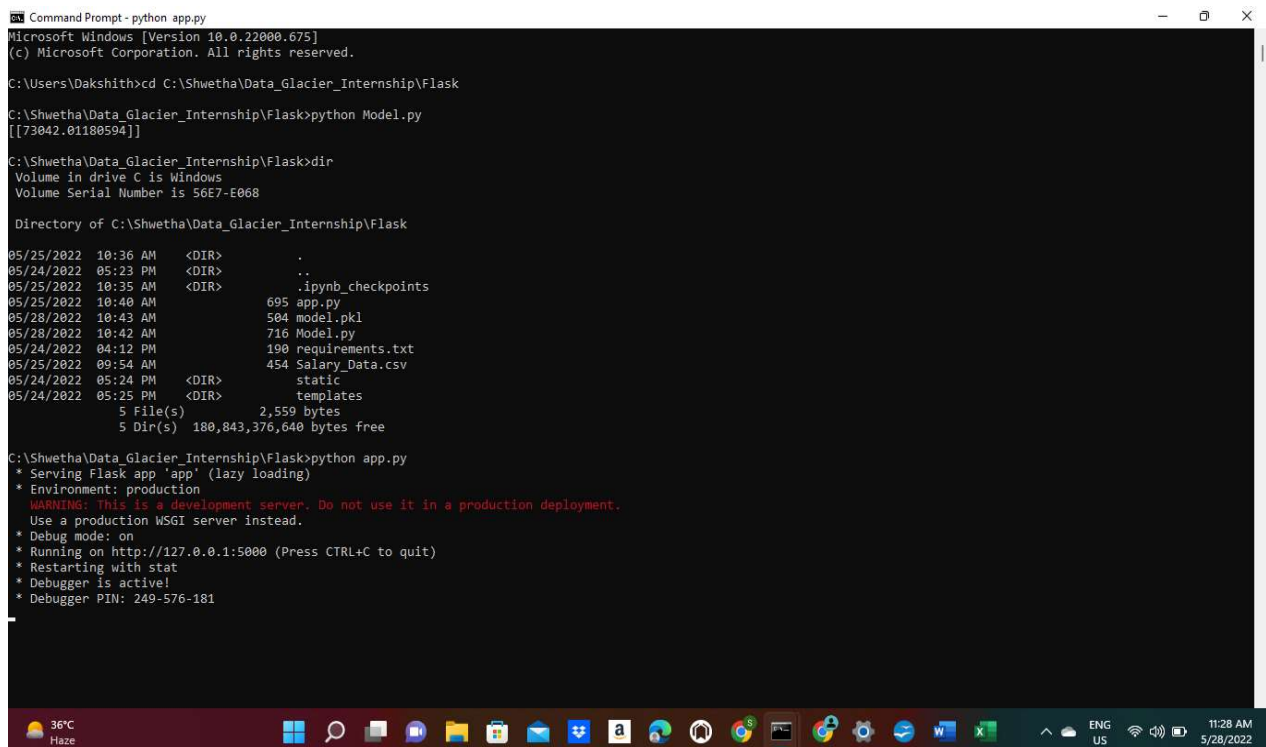
4. Then Create a app.py file to deploy the Model in flask as below.



The screenshot shows a Jupyter Notebook interface with a file named 'app.py' open. The code is a Flask application that loads a pre-trained model and provides a web interface for predictions. The code is as follows:

```
1 import numpy as np
2 from flask import Flask, request, render_template
3 import pickle
4
5 app = Flask(__name__)
6 model = pickle.load(open('model.pkl', 'rb'))
7
8 @app.route('/')
9 def home():
10     return render_template('index.html')
11
12 @app.route('/predict', methods=['POST'])
13 def predict():
14     """
15     For rendering results on HTML GUI
16     """
17     float_features = [float(x) for x in request.form.values()]
18     final_features = [np.array(float_features)]
19     prediction = model.predict(final_features)
20
21     output = np.round(prediction[0], 2)
22
23     return render_template('index.html', prediction_text='Salary will be ${}'.format(output))
24
25 if __name__ == "__main__":
26     app.run(debug=True)
```

5. Go to folder and run the app.py file as below.



The screenshot shows a Windows Command Prompt window where the user has navigated to the directory 'C:\Shwetha\Data_Glacier_Internship\Flask' and run the command 'python app.py'. The output shows the directory contents and the Flask application running on http://127.0.0.1:5000.

```
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dakshith>cd C:\Shwetha\Data_Glacier_Internship\Flask
C:\Shwetha\Data_Glacier_Internship\Flask>python Model.py
[[73042.01180594]]

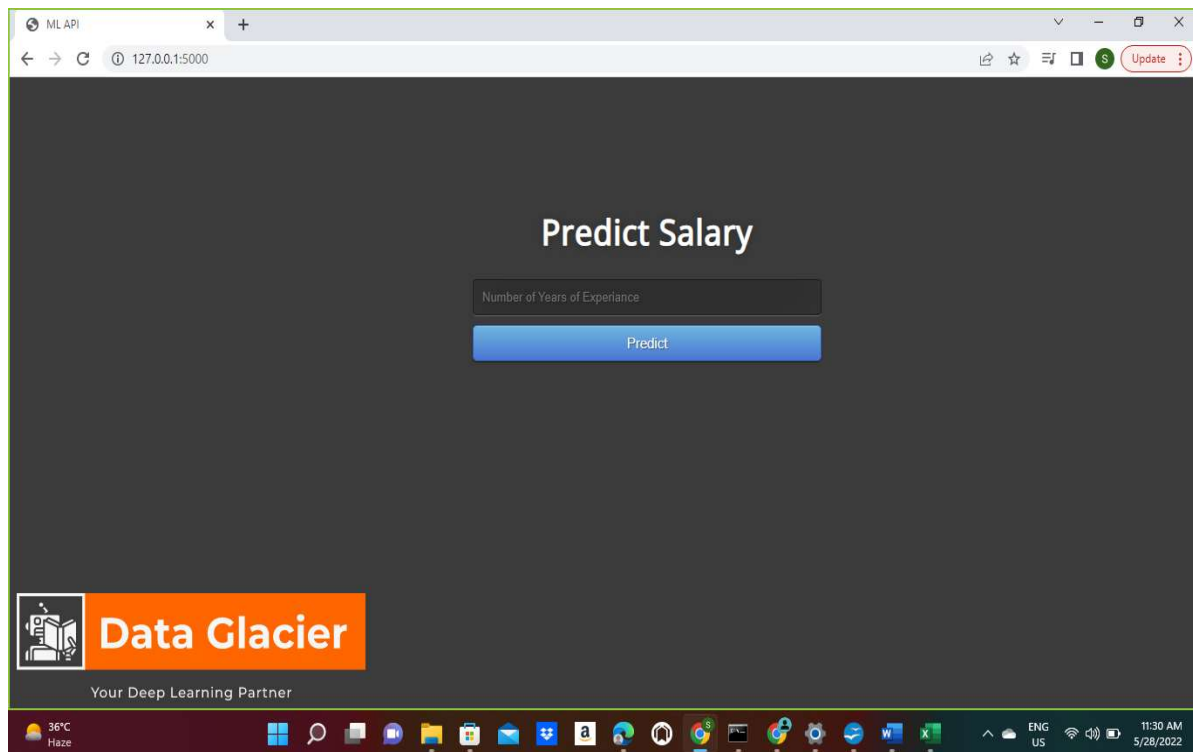
C:\Shwetha\Data_Glacier_Internship\Flask>dir
Volume in drive C is Windows
Volume Serial Number is 56E7-E068

Directory of C:\Shwetha\Data_Glacier_Internship\Flask

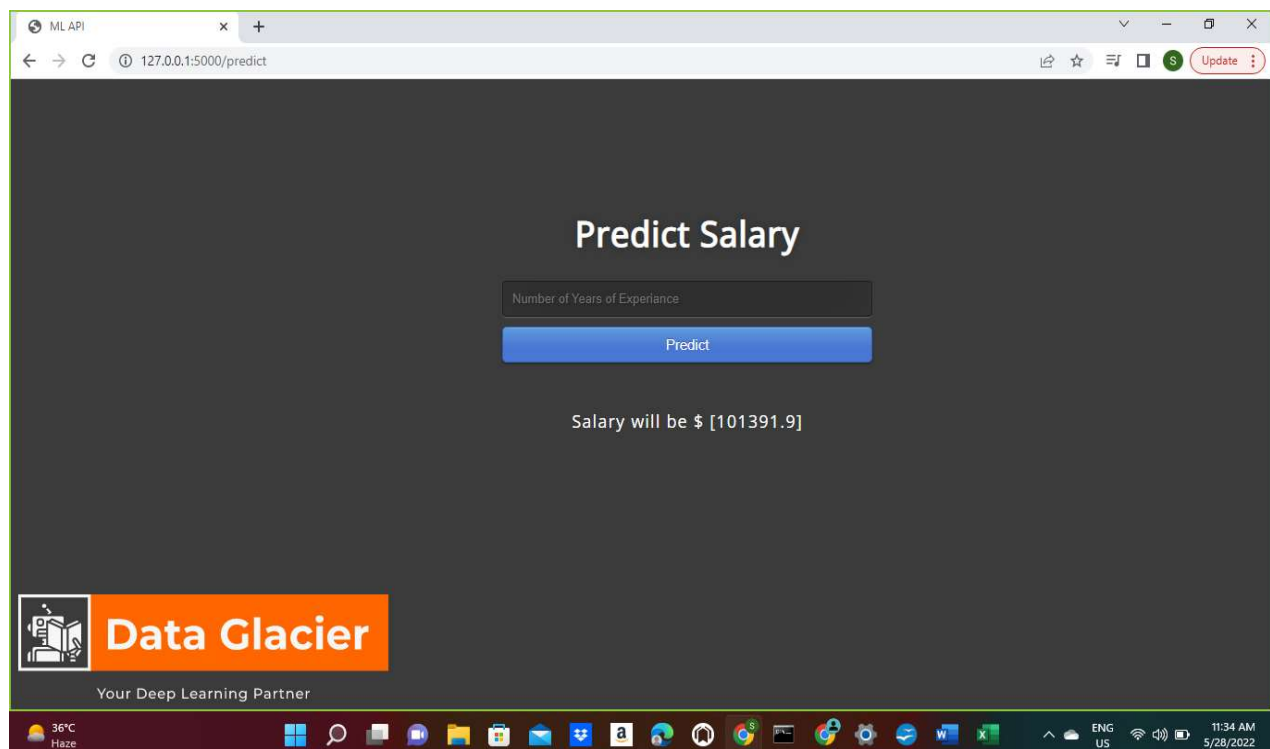
05/25/2022 10:36 AM <DIR>      .
05/24/2022 05:23 PM <DIR>      ..
05/25/2022 10:35 AM <DIR>      .ipynb_checkpoints
05/25/2022 10:40 AM          695 app.py
05/28/2022 10:43 AM          504 model.pkl
05/28/2022 10:42 AM          716 Model.py
05/24/2022 04:12 PM          190 requirements.txt
05/25/2022 09:54 AM          454 Salary_Data.csv
05/24/2022 05:24 PM <DIR>      static
05/24/2022 05:25 PM <DIR>      templates
                    5 File(s)      2,559 bytes
                    5 Dir(s)  180,843,376,640 bytes free

C:\Shwetha\Data_Glacier_Internship\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
* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 249-576-181
```

6. Copy paste the URL <http://127.0.0.1:5000> in any browser, then you will be prompted for asking the number of years of experience to predict salary as below



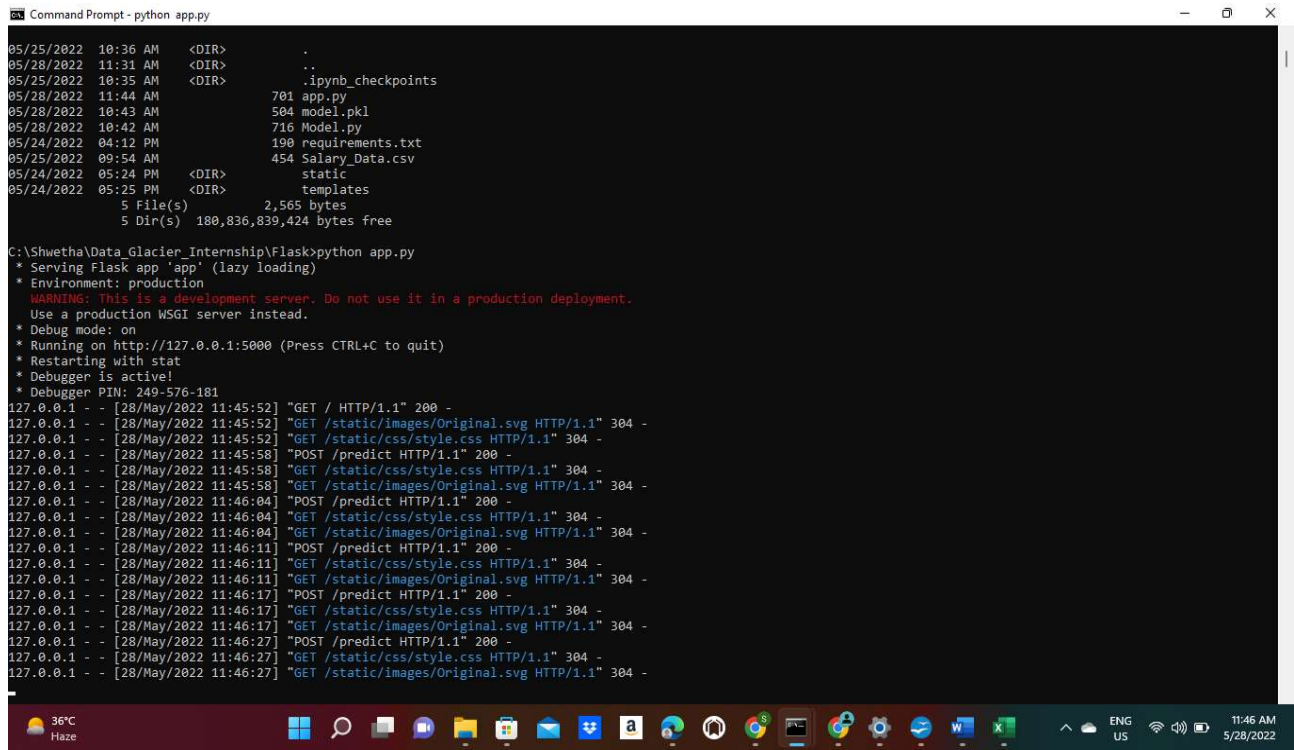
7. Enter the Number of Years of experience and click on submit button it will show the predicted salary as below.



This is how we can predict the salary by looking at there Number of experience by saving a model and deploying it in Flask.

#HTML and image templates have to saved in the same folder where have Model.py and app.py

Below is the screenshot of POST and GET operations



```
Command Prompt - python app.py

05/25/2022 10:36 AM <DIR> .
05/28/2022 11:31 AM <DIR> ..
05/25/2022 10:35 AM <DIR> .ipynb_checkpoints
05/28/2022 11:44 AM      701 app.py
05/28/2022 10:43 AM      504 model.pkl
05/28/2022 10:42 AM      716 Model.py
05/24/2022 04:12 PM      190 requirements.txt
05/25/2022 09:54 AM      454 Salary_Data.csv
05/24/2022 05:24 PM <DIR> static
05/24/2022 05:25 PM <DIR> templates
      5 File(s)      2,565 bytes
      5 Dir(s) 180,836,839 bytes free

C:\Shwetha\Data_Glacier_Internship\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
* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 249-576-181
127.0.0.1 - - [28/May/2022 11:45:52] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [28/May/2022 11:45:52] "GET /static/images/Original.svg HTTP/1.1" 304 -
127.0.0.1 - - [28/May/2022 11:45:52] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [28/May/2022 11:45:56] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [28/May/2022 11:45:58] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [28/May/2022 11:45:58] "GET /static/images/Original.svg HTTP/1.1" 304 -
127.0.0.1 - - [28/May/2022 11:46:04] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [28/May/2022 11:46:04] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [28/May/2022 11:46:04] "GET /static/images/Original.svg HTTP/1.1" 304 -
127.0.0.1 - - [28/May/2022 11:46:11] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [28/May/2022 11:46:11] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [28/May/2022 11:46:11] "GET /static/images/Original.svg HTTP/1.1" 304 -
127.0.0.1 - - [28/May/2022 11:46:17] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [28/May/2022 11:46:17] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [28/May/2022 11:46:17] "GET /static/images/Original.svg HTTP/1.1" 304 -
127.0.0.1 - - [28/May/2022 11:46:27] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [28/May/2022 11:46:27] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [28/May/2022 11:46:27] "GET /static/images/Original.svg HTTP/1.1" 304 -
```