Data Science Intern at Data Glacier

Week-4 : Deployment on Flask

Name : Shweta Singh

Batch Code : LISUM14

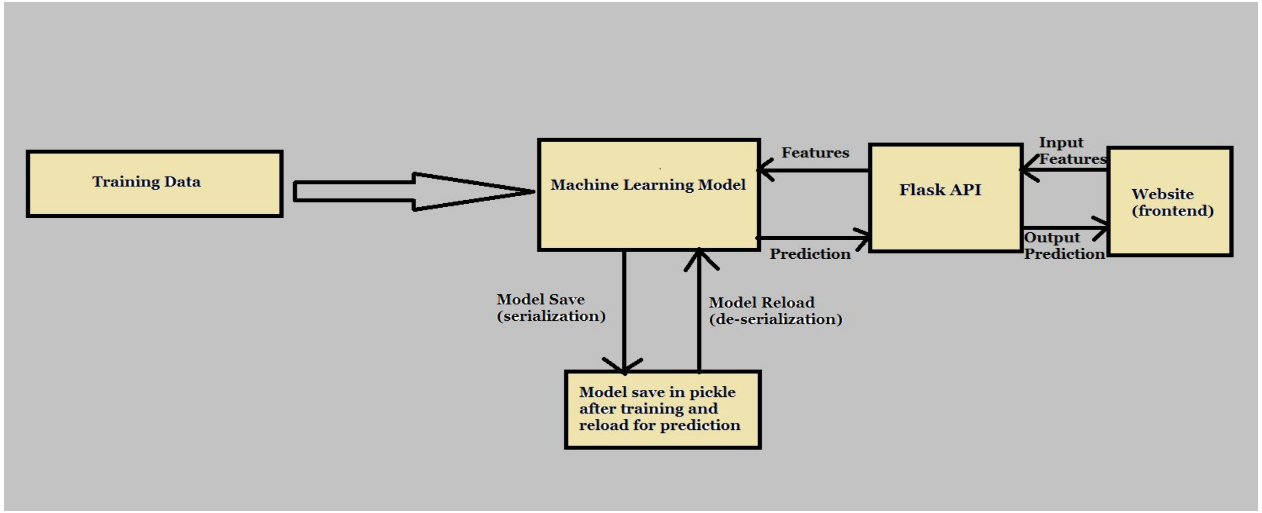
Submission Date : 29 October 2022

Submitted to : Data Glacier

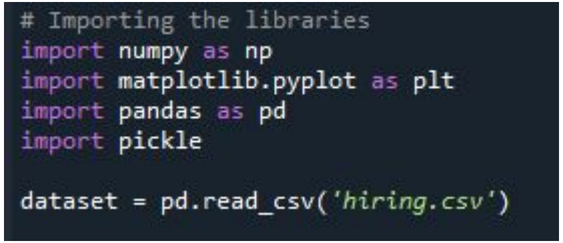
Flask

Flask is a a **web application framework written in python**, in simple terms it helps end users interact with your python code (in this case our ML models) directly from their web browser without needing any libraries,code files, etc.

Machine learning model deployment using Flask:


Training Data
Machine Learning Model
Features
Prediction
Flask API
Input Features
Output Prediction
Website (frontend)
Model Save (serialization)
Model Reload (de-serialization)
Model save in pickle after training and reload for prediction

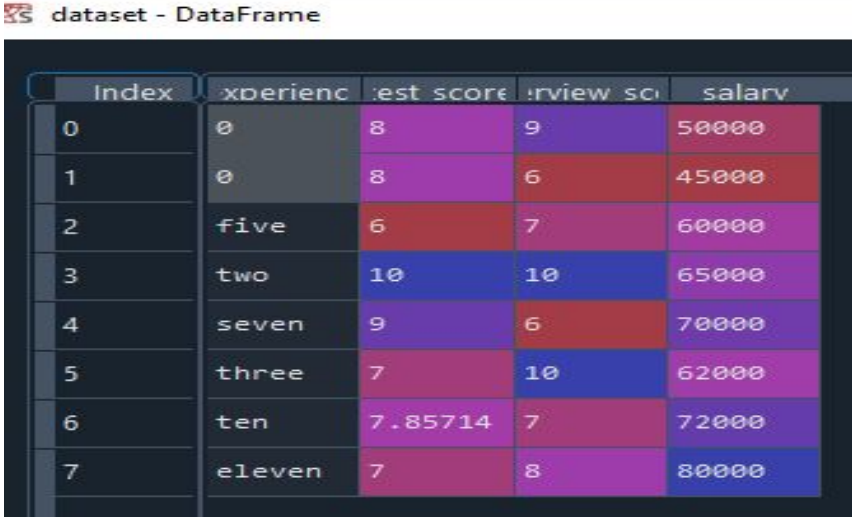
First we will import some libraries here and then read dataset—--



```python
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pickle

dataset = pd.read_csv('hiring.csv')
```

Dataset : (Hiring.csv)

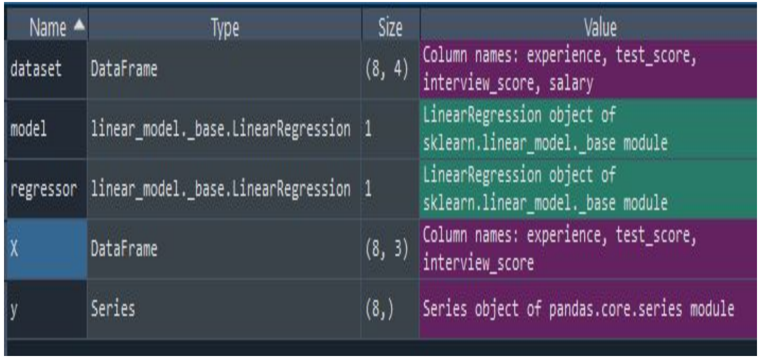
Field are:Experience, Test_Score,Interview_Score and Salary



dataset - DataFrame

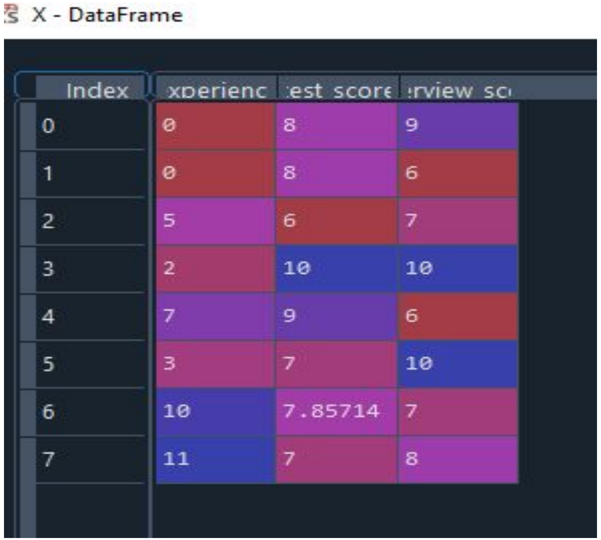
Index	xperienc	est score	rview sc	salary
0	0	8	9	50000
1	0	8	6	45000
2	five	6	7	60000
3	two	10	10	65000
4	seven	9	6	70000
5	three	7	10	62000
6	ten	7.85714	7	72000
7	eleven	7	8	80000

These are part of Data:



Name ▲	Type	Size	Value
dataset	DataFrame	(8, 4)	Column names: experience, test_score, interview_score, salary
model	linear_model._base.LinearRegression	1	LinearRegression object of sklearn.linear_model._base module
regressor	linear_model._base.LinearRegression	1	LinearRegression object of sklearn.linear_model._base module
X	DataFrame	(8, 3)	Column names: experience, test_score, interview_score
y	Series	(8,)	Series object of pandas.core.series module

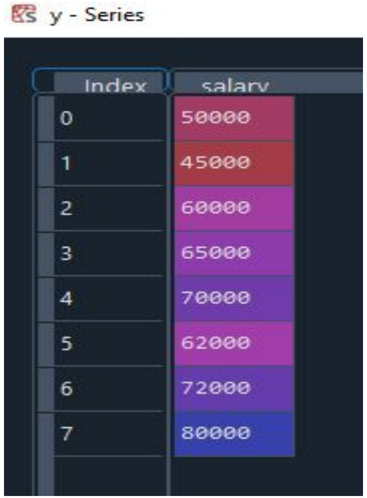
X-Dataframe



X - DataFrame

Index	xperienc	est score	rview sc
0	0	8	9
1	0	8	6
2	5	6	7
3	2	10	10
4	7	9	6
5	3	7	10
6	10	7.85714	7
7	11	7	8

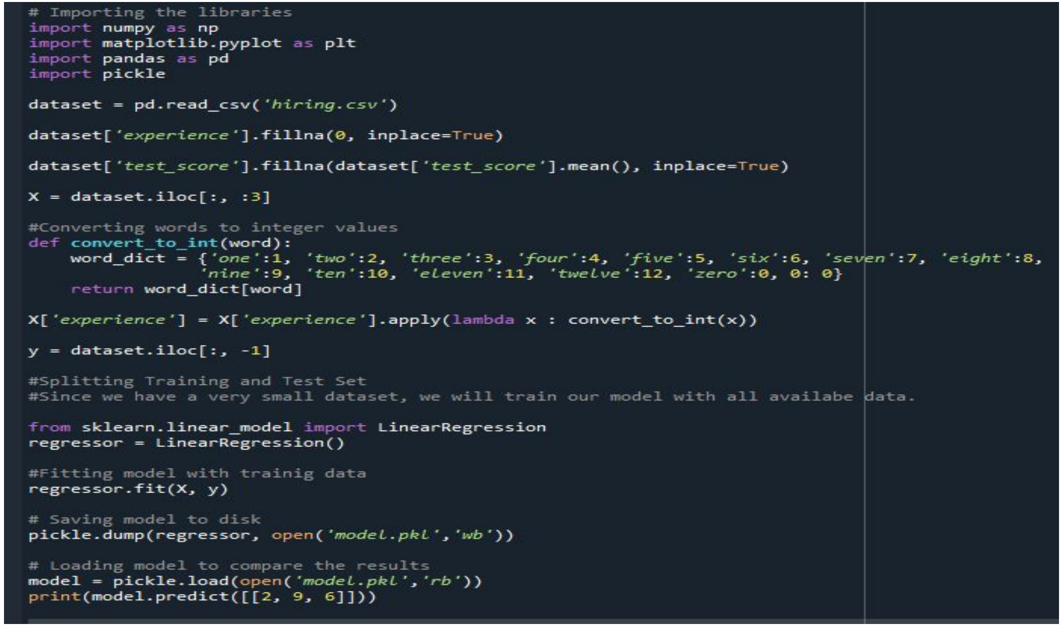
Here y is dependent Variable :



y - Series

Index	salary
0	50000
1	45000
2	60000
3	65000
4	70000
5	62000
6	72000
7	80000

First we will build model : **(model.py)**



```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pickle

dataset = pd.read_csv('hiring.csv')

dataset['experience'].fillna(0, inplace=True)

dataset['test_score'].fillna(dataset['test_score'].mean(), inplace=True)

X = dataset.iloc[:, :3]

#Converting words to integer values
def convert_to_int(word):
    word_dict = {'one':1, 'two':2, 'three':3, 'four':4, 'five':5, 'six':6, 'seven':7, 'eight':8,
                'nine':9, 'ten':10, 'eleven':11, 'twelve':12, 'zero':0, 0: 0}
    return word_dict[word]

X['experience'] = X['experience'].apply(lambda x : convert_to_int(x))

y = dataset.iloc[:, -1]

#Splitting Training and Test Set
#Since we have a very small dataset, we will train our model with all availabe data.

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()

#Fitting model with trainig data
regressor.fit(X, y)

# Saving model to disk
pickle.dump(regressor, open('model.pkl','wb'))

# Loading model to compare the results
model = pickle.load(open('model.pkl','rb'))
print(model.predict([[2, 9, 6]]))
```

#Splitting Training and Test Set

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()

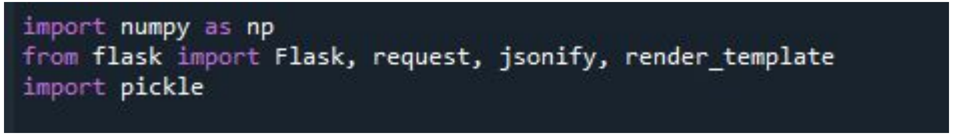
```
#Fitting model with training data

regressor.fit(X, y)
```

```
# Saving model to disk

pickle.dump(regressor,
open('model.pkl','wb'))
```

```
# Loading model to compare the results

model = pickle.load(open('model.pkl','rb'))
print(model.predict([[2, 9, 6]]))
```

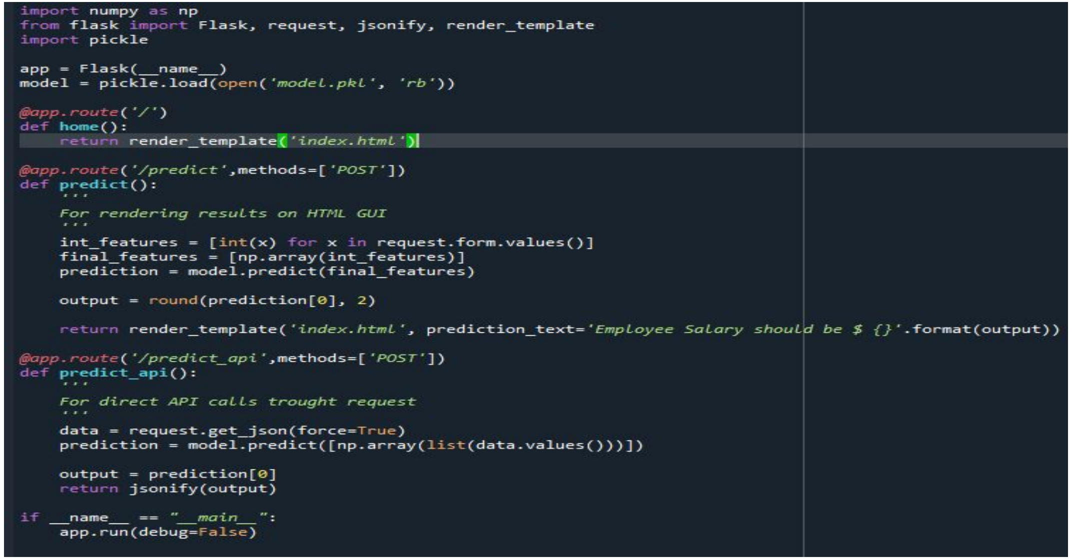
Now model.pkl file is ready

Now we will create **(app.py)**
import some libraries—



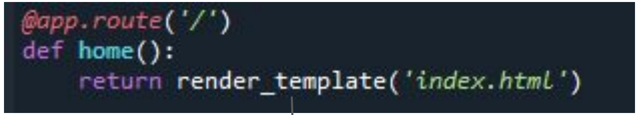
```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
```

Here i have used flask to post my model.
render_template is used for redirect to home page

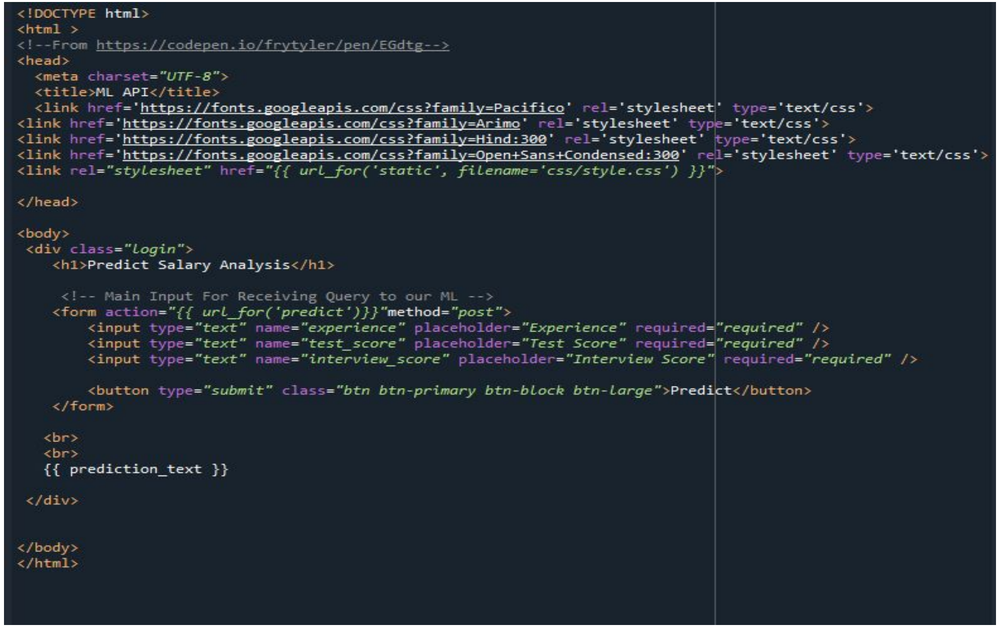


```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

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

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

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    int_features = [int(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    output = round(prediction[0], 2)

    return render_template('index.html', prediction_text='Employee Salary should be $ {}'.format(output))

@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls trought request
    '''
    data = request.get_json(force=True)
    prediction = model.predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)

if __name__ == "__main__":
    app.run(debug=False)
```

When we execute this code it will redirect to home page (index.html)



```python
@app.route('/')
def home():
    return render_template('index.html')
```

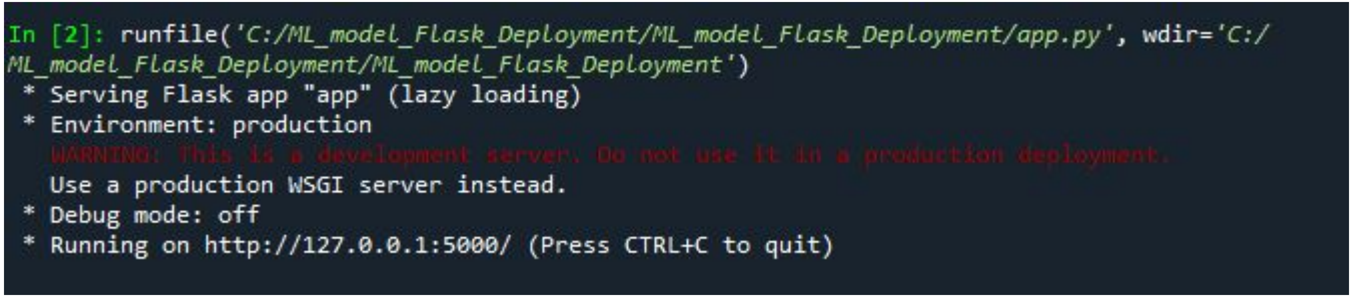


```html
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>ML API</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body>
 <div class="login">
    <h1>Predict Salary Analysis</h1>

     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
        <input type="text" name="experience" placeholder="Experience" required="required" />
        <input type="text" name="test_score" placeholder="Test Score" required="required" />
        <input type="text" name="interview_score" placeholder="Interview Score" required="required" />

        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>

   <br>
   <br>
   {{ prediction_text }}

 </div>


</body>
</html>
```

Home_page (index.html)

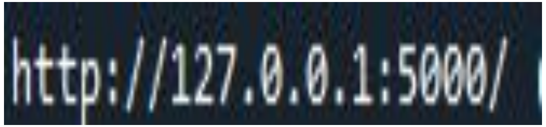
Deployment and Output

After executing **app.py** we will get this —



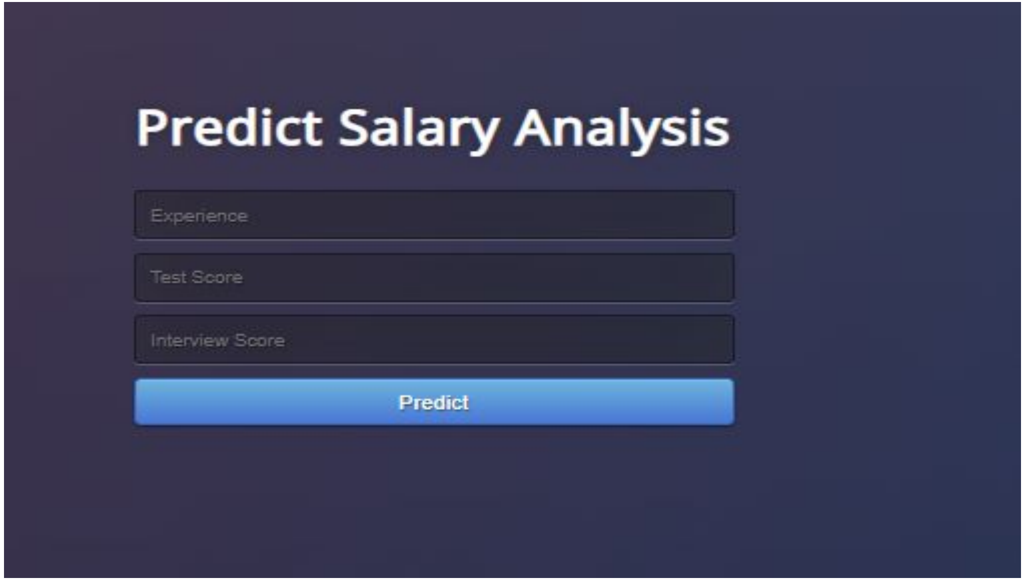
```
In [2]: runfile('C:/ML_model_Flask_Deployment/ML_model_Flask_Deployment/app.py', wdir='C:/
ML_model_Flask_Deployment/ML_model_Flask_Deployment')
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Then we will copy this path in the browser



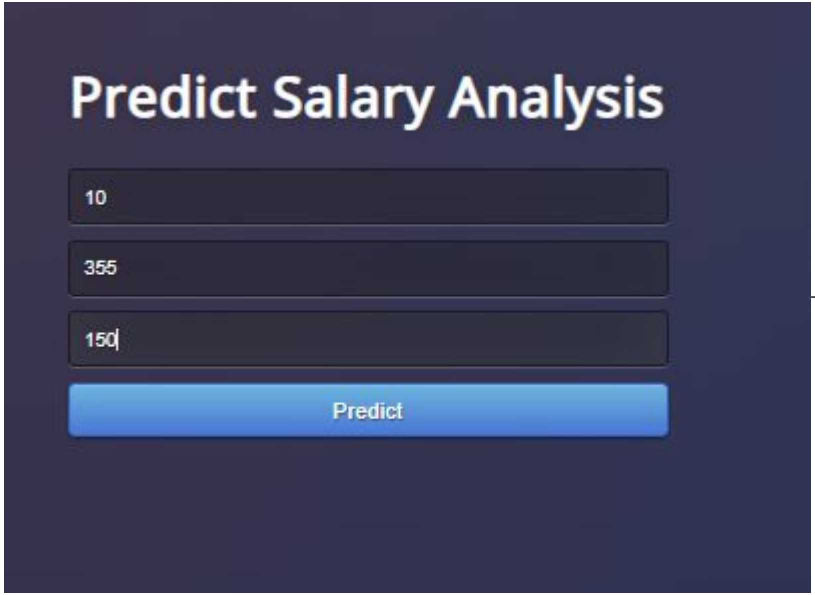
```
http://127.0.0.1:5000/
```

This is our **Home_Page**



Predict Salary Analysis

After providing required input :



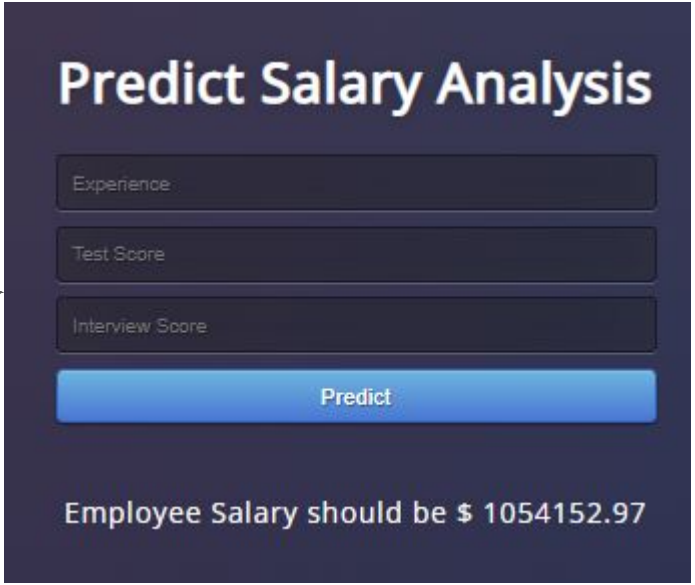
Predict Salary Analysis

10

355

150

Here comes the output



Predict Salary Analysis

Employee Salary should be $ 1054152.97