DATA SCIENCE INTERN AT DATA GLACIER

Week 4: Deployment on Flask

Name: AMRAOUI FATIMA EZZAHRA

Batch Code: LISUM13

Date: 27 SEPTEMBER 2022

Submitted to: DATA GLACIER

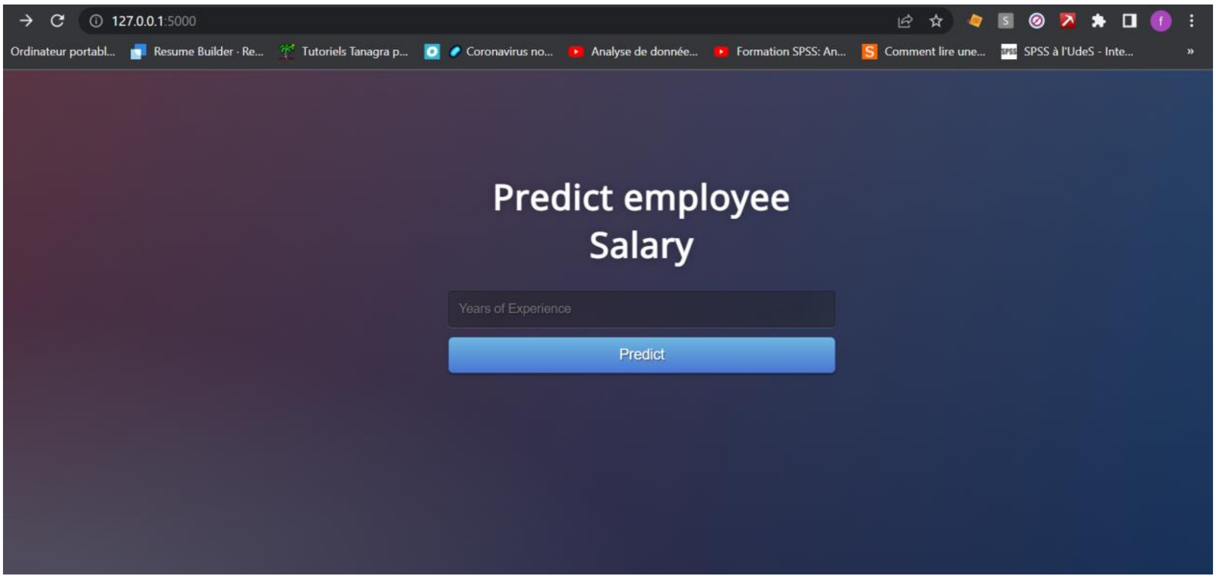
Table of Contents

I. Introduction:

The purpose of this project is the deployment of a machine learning model using Flask framework. The model aims to predict the employee salary based on its years of experience, hence it's a simple linear regression problem.

First, we create our machine learning model and save it using pickle module. Then create the API using Flask.

This is how our application looks like :



II. Dataset description:

The dataset was downloaded from Kaggle https://www.kaggle.com/datasets/suyog17/salary-prediction it has a size of **47.87 KB** and contains one predictor variable (years of experience) and the target variable(salary).

The shape of this dataset is (5000, 2) with years of experience between 0 & 29 and salary range is from 1036 $ to 99980 $.

The dataset is clear and simple as required in the assignment.

III. Model building & saving:

First let's import the necessary packages:

```
#Simple linear regresion
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.stats import pearsonr
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score,accuracy_score
from sklearn.metrics import mean_squared_error

import statsmodels.api as sm
import statsmodels.formula.api as smf
import pickle
```

This is how the dataset looks like:

```
dataset = pd.read_csv('/content/drive/MyDrive/Sal_vs_Exp.csv')
print(dataset)
x = dataset[['Years of Experience']]
y = dataset['Salary']
```

```
      Years of Experience  Salary
0                       0    1036
1                       0    1041
2                       0    1054
3                       0    1069
4                       0    1110
...                   ...     ...
4995                   29   99841
4996                   29   99934
4997                   29   99940
4998                   29   99941
4999                   29   99980

[5000 rows x 2 columns]
```

In the scatter plot below, we notice that we have a strong linear relationship between years of experience and salary so linear regression model will be a good fit to our problem:

```
fig, ax = plt.subplots(figsize=(6, 3.84))

dataset.plot(
    x     = 'Years of Experience',
    y     = 'Salary',
    c     = 'firebrick',
    kind = "scatter",
    ax    = ax)

ax.set_title('Salary distribution');
plt.show()
```


```
corr_test = pearsonr(x = dataset['Years of Experience'], y =  dataset['Salary'])
print("pearson's coeficient correlation: ", corr_test[0])
print("P-value: ", corr_test[1])
```

```
pearson's coeficient correlation:  0.9992745685977932
P-value:  0.0
```

We have run a Pearson test to gauge the degree of correlation and we get r = 0.999 which is almost equal to 1, which means that age and years of experience form an absolute linear relationship.

Now let's split the data into two subsets trainset and test set:

```
[8] x_train, x_test, y_train, y_test = train_test_split(x.values.reshape(-1,1),
                                                        y.values.reshape(-1,1),
                                                        test_size = 1/3,
                                                        random_state= 1234,
                                                        shuffle=True)


    model = LinearRegression()
    model.fit(X= x_train.reshape(-1,1), y= y_train)
```

```
LinearRegression()
```

We dedicated almost 30% of data to test and 70% for training set

```
[9] print(x_train.shape)
    print(y_train.shape)
```

```
(3333, 1)
(3333, 1)
```

```
print('intercept: ', model.intercept_)
print('coeficiente: ', list(zip(x.columns,model.coef_.flatten(),)))
print('R^2: ',model.score(x,y))
print("=================================")
# model test error
# ==============================================================================
pred = model.predict(X = x_test)
print(pred[0:3,])
print("=================================")
rmse = mean_squared_error(
        y_true  = y_test,
        y_pred  = pred)
print("")
print(f"the (rmse) test error is: {rmse}")
```

```
intercept:  [1928.24104939]
coeficiente:  [('Years of Experience', 3343.4598406882665)]
R^2:  0.9985486995191352
=================================
[[55423.5985004 ]
 [48736.67881903]
 [25332.45993421]]
=================================

the (rmse) test error is: 1223068.0310047166
```

R^2 = 99% (goodness of fit of the model) which indicates that the model is powerful.

```
X_train = sm.add_constant(x_train, prepend=True)
model2 = sm.OLS(endog=y_train, exog=X_train)
model2 = model2.fit()
print(model2.summary())
```

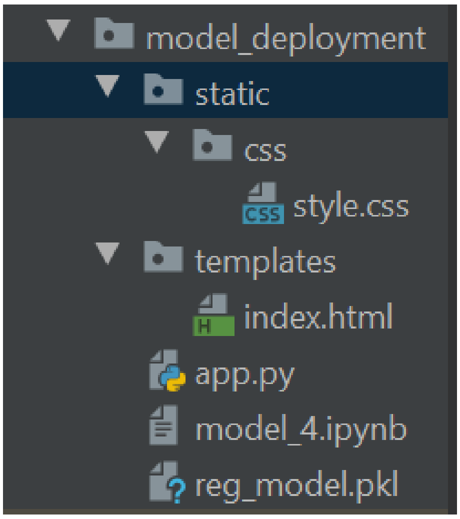
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.999
Method:                 Least Squares   F-statistic:                 2.287e+06
Date:                Tue, 27 Sep 2022   Prob (F-statistic):               0.00
Time:                        15:37:07   Log-Likelihood:                -28064.
No. Observations:                3333   AIC:                         5.613e+04
Df Residuals:                    3331   BIC:                         5.614e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       1928.2410     37.272     51.735      0.000    1855.163    2001.319
x1          3343.4598      2.211   1512.261      0.000    3339.125    3347.795
==============================================================================
Omnibus:                      182.542   Durbin-Watson:                   1.960
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               70.101
Skew:                          -0.013   Prob(JB):                     5.99e-16
Kurtosis:                       2.290   Cond. No.                         33.1
==============================================================================

Notes:
```

Now we save the model as reg_model using pickle:

```
linear_reg = open("reg_model.pkl","wb")
pickle.dump(model,linear_reg)
linear_reg.close()
```

IV. Model deployment using Flask:

We develop a web application that consists of a simple web page with a form field that lets us enter the years of experience. After submitting the number of years to the web application, it will render the result of the predicted salary. First, we create a folder for this project called model_deployment, this is the directory tree inside the folder:



1. app.py:

The app.py file contains the main code that will be executed by the Python interpreter to run the Flask web application.

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

app = Flask(__name__)
model = pickle.load(open('reg_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 = np.round(prediction[0], 2)

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


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

We ran our application as a single module; thus, we initialized a new Flask instance with the argument __name__ to let Flask know that it can find the HTML template folder (templates) in the same directory where it is located. Next, we used the route decorator (@app.route('/')) to specify the URL that should trigger the execution of the home function.

Our home function simply rendered the index.html HTML file, which is located in the templates folder.

Inside the predict function, we access the input entered by the user and use our model to make a prediction for its input.

Lastly, we used the run function to only run the application on the server when this script is directly executed by the Python interpreter, which we ensured using the if statement with __name__ == '__main__'.

2. Index.html:

The following are the contents of the Index.html file that will render a text form where a user can enter a number of years of experience.

```
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>model deployment</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 employee Salary</h1>
     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
        <input type="text" name="Years of Experience" placeholder="Years of Experience" required="required" />
        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>
   <br>
   <br>
   {{ prediction_text }}
 </div>
</body>
</html>
```

3. Style.css:

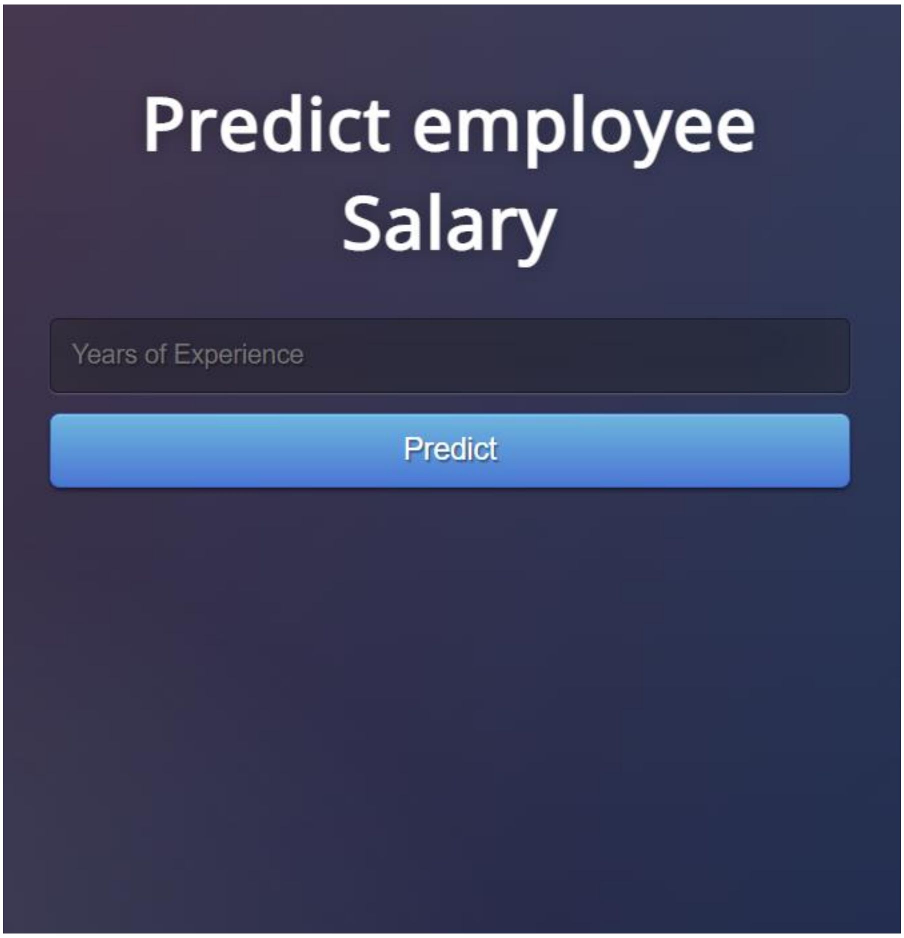
CSS is to determine how the look and feel of HTML documents.The capture of code won't be provided here since it's relatively long and optional.

4. App execution:

Now we can execute the app.py file as shown below :

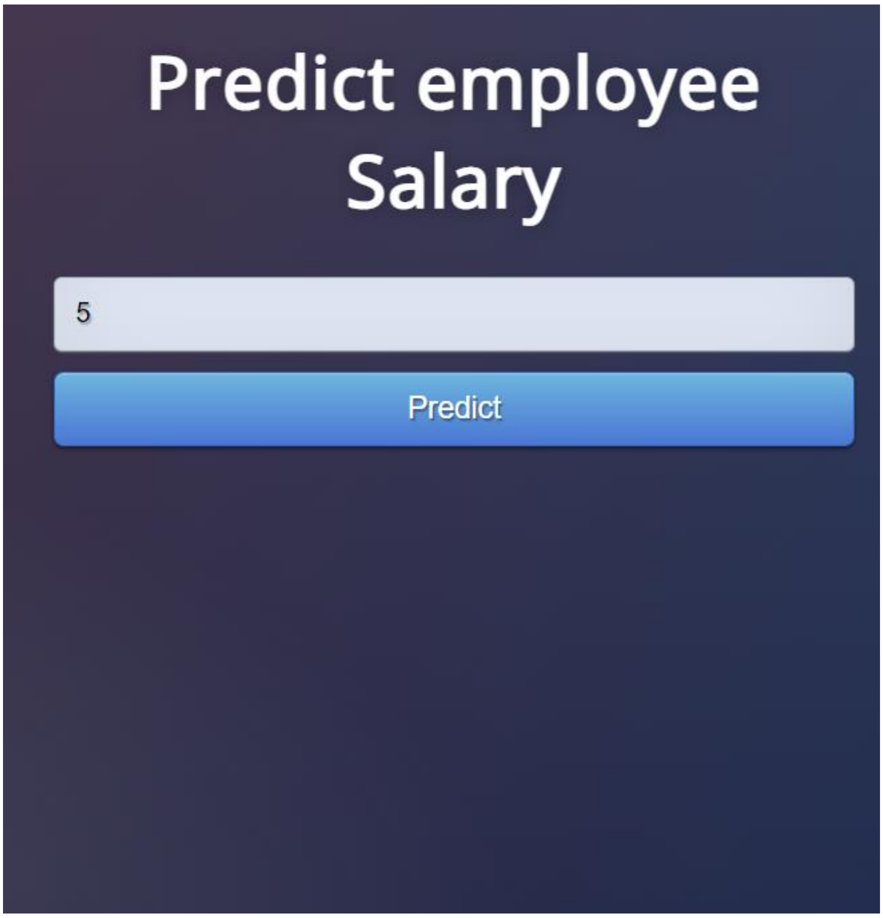
```
(venv) C:\Users\user\PycharmProjects\salary_app_deployment\model_deployment>ls
app.py  model_4.ipynb  reg_model.pkl  static  templates

(venv) C:\Users\user\PycharmProjects\salary_app_deployment\model_deployment>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 814-204-614
```

We can access to the web API from our web browser using *http://127.0.0.1:5000* :

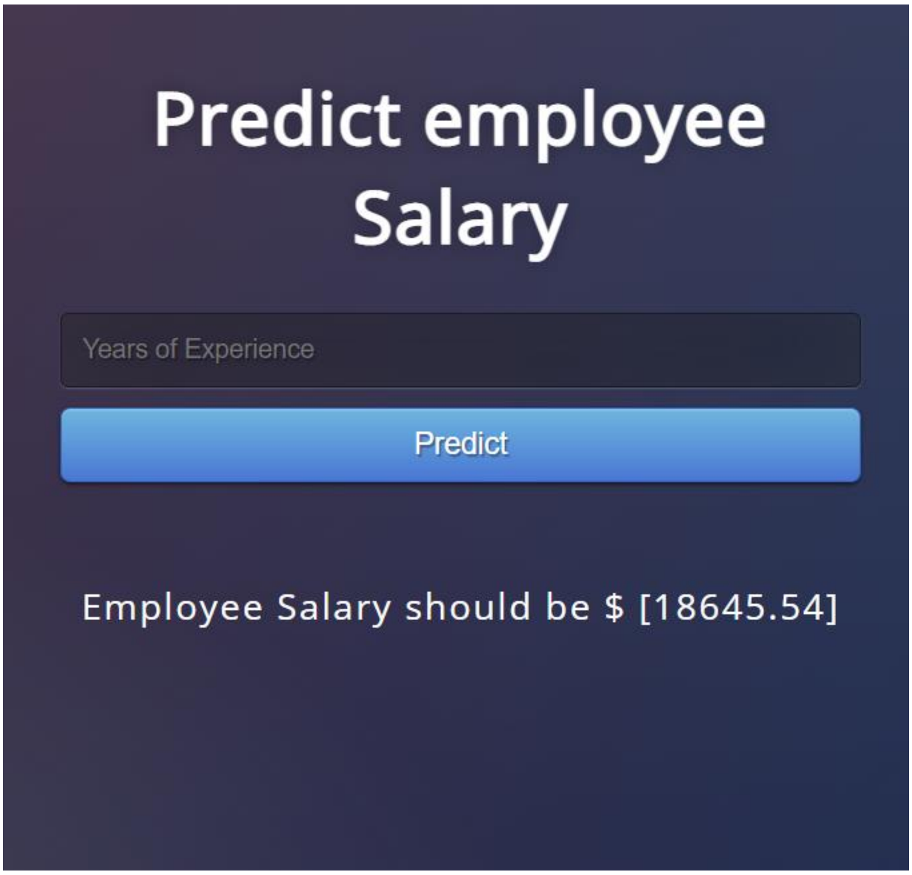


Now we can see this simple form where the user may enter any number of experience and predict the salary based on its input.

Let’s try the app functioning by entering 5 as number of years of experience:



Now by pressing the predict button we get :


Predict employee Salary
Years of Experience
Predict
Employee Salary should be $ [18645.54]