

# ML web app deployment using Flask

Name: Juan Carlos

Batch: LISP01

Submission date: 23-March-2021

Submitted to: Data Glacier

Repository: [https://github.com/jaycee-ds/Salary\\_ModelDeployment\\_Flask](https://github.com/jaycee-ds/Salary_ModelDeployment_Flask)

The goal of the assignment is to build a Machine Learning model and deploy it in a web app using the framework Flask. In this case, I used a very simple dataset and a simple linear regression algorithm that takes years of experience as input and returns the expected salary. It is my first web app. These are the steps I have followed:

Step 1: Collect data and build a model. I used a Kaggle dataset with salaries and their corresponding years of experience (it can be found here: <https://www.kaggle.com/karthickveerakumar/salary-data-simple-linear-regression>).

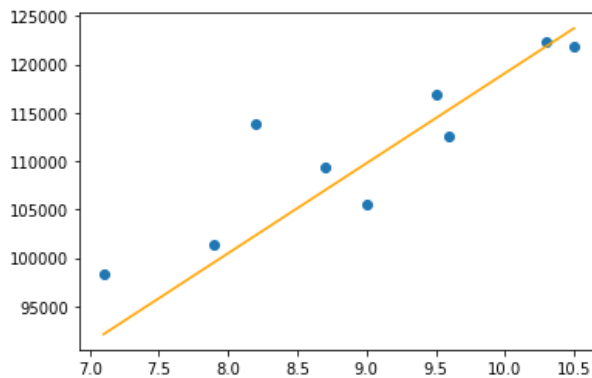
Imported the relevant libraries, loaded the data, plotted it, split the data into train and validation sets and set the target and explanatory variables, say salary and years of experience respectively.

```
1  import numpy as np
2  import pandas as pd
3  import matplotlib.pyplot as plt
4  import statsmodels.api as sm
5  from sklearn.metrics import mean_absolute_error, mean_squared_error
6  import pickle
7
8  data = pd.read_csv('Salary_Data.csv')
9  data
10
11  plt.scatter(data.YearsExperience, data.Salary)
12  plt.show()
13
14  # Split data into train and validation
15  train = data[:int(0.7*len(data))]
16  valid = data[int(0.7*len(data)):]
17
18  # Assigning feature vectors to target and explanatory variable
19  Y_train = train.Salary
20  X_train = train.YearsExperience
21  X_train = sm.add_constant(X_train)
22
23  Y_valid = valid.Salary
24  X_valid = valid.YearsExperience
25  X_valid = sm.add_constant(X_valid)
```

Step 2: Build a model using a simple linear regression algorithm, predict using the validation dataset and see how it performs. These are the results:

The MAE is 3745.563327835247

The RMSE is 4879.788496122362

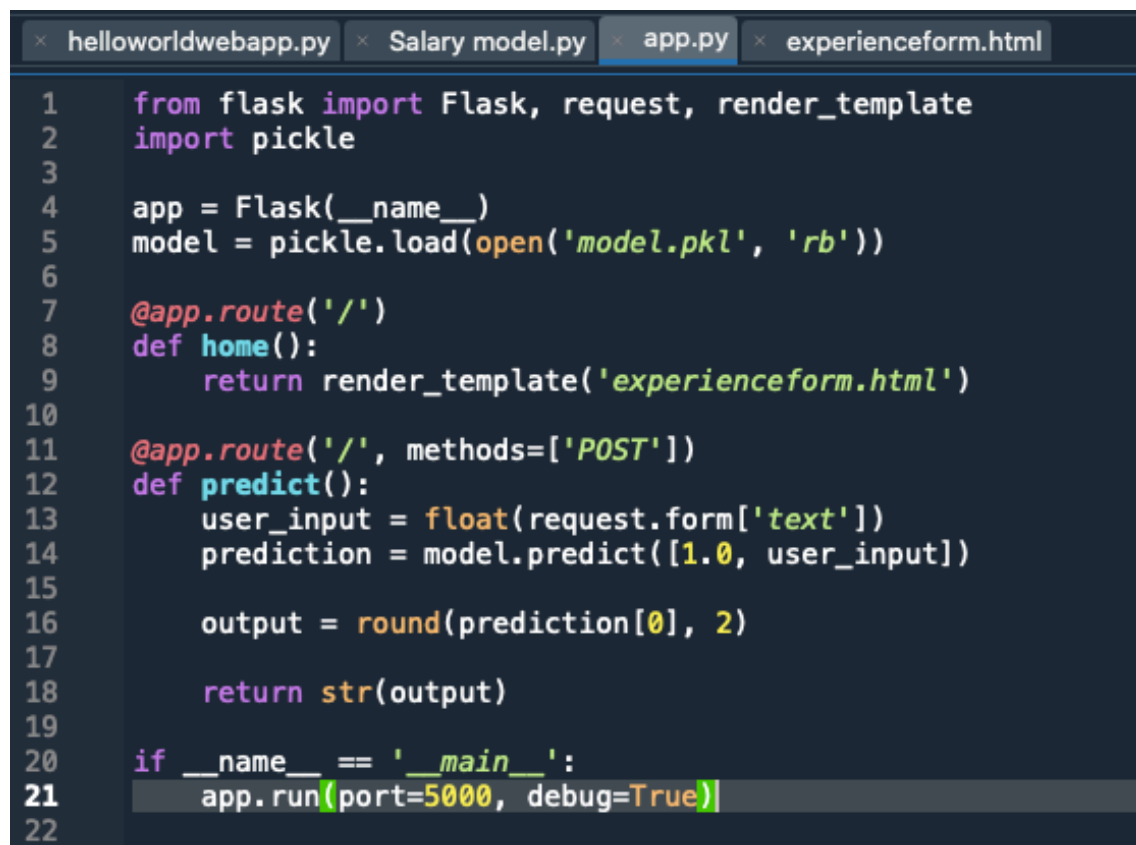


Below is the code for that. Last, I pickled the model object in a binary file so that I can use it later on.

```
26
27 # Building a Simple Linear Regression model
28 model = sm.OLS(Y_train, X_train)
29 results = model.fit()
30 results.params
31
32 # Predict with validation data
33 predicted = results.predict(X_valid)
34
35 # Model performance
36 print(f'The MAE is {mean_absolute_error(Y_valid, predicted)}')
37 print(f'The RMSE is {np.sqrt(mean_squared_error(Y_valid, predicted))}')
38
39 # Visualizing predictions and actual values
40 plt.scatter(Y_valid, predicted)
41 plt.show()
42
43 # Visualizing the model
44 plt.scatter(X_valid.YearsExperience, Y_valid)
45 plt.plot(X_valid.YearsExperience, predicted, color='orange')
46 plt.show()
47
48 with open('model.pkl', 'wb') as f:
49     pickle.dump(results, f)
50     print('Pickling completed')
```

Step 3: I created the Python script and HTML files to build the web app. I imported the relevant libraries (including Flask as we can see). So first we create the Flask object assigned to the variable “app” and unpickled the model object that was created previously assigning it to the variable “model”.

The @app.route decorator lets us indicate the path as well as the allowed methods (HTTP verbs). The first function simply renders the HTML file which shows a simple form where you can enter a specific number for “years of experience” (the explanatory variable of our model). The second function takes the number from the previous form and inputs it as a parameter of our OLS model. Last, it formats the output to returns it. That is what we see in our web app.

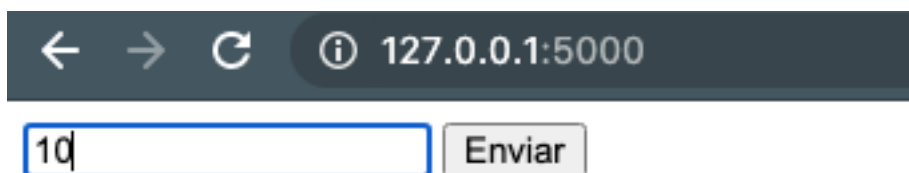


```
1  from flask import Flask, request, render_template
2  import pickle
3
4  app = Flask(__name__)
5  model = pickle.load(open('model.pkl', 'rb'))
6
7  @app.route('/')
8  def home():
9      return render_template('experienceform.html')
10
11 @app.route('/', methods=['POST'])
12 def predict():
13     user_input = float(request.form['text'])
14     prediction = model.predict([1.0, user_input])
15
16     output = round(prediction[0], 2)
17
18     return str(output)
19
20 if __name__ == '__main__':
21     app.run(port=5000, debug=True)
22
```

This is how our HTML file looks like. It is a very simple form.

```
× helloworldwebapp.py × Salary model.py × app.py × experienceform.html
1 <form method="POST">
2   <input name="text">
3   <input type="submit">
4 </form>
5
```

The first page we encounter in our app is a box to enter the desired number of years of experience and a submit button.



← → ↻ ⓘ 127.0.0.1:5000

10 Enviar

The second page showcases the result. Simply, it returns the predicted salary according to the years of experience.



← → ↻ ⓘ 127.0.0.1:5000

119083.34