



Flask app deployment Project

Name :Refka Mejri

Email : Refka.mejri@enit.utm.tn

Country:Tunisia

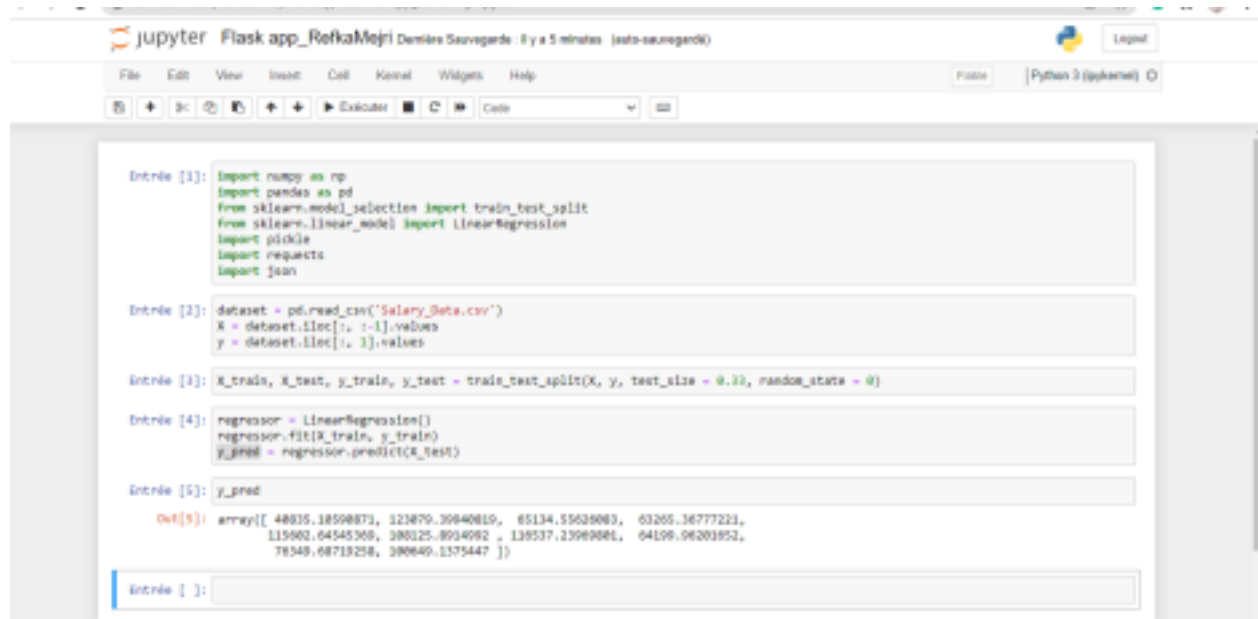
College: National Engineering School of Tunis

Specialization :Data Science



First step:

Model prediction :



```
Entrée [1]: import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import pickle
import requests
import json

Entrée [2]: dataset = pd.read_csv('Salary_Data.csv')
X = dataset.iloc[:, 1:2].values
y = dataset.iloc[:, 2].values

Entrée [3]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 0)

Entrée [4]: regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)

Entrée [5]: y_pred

Out[5]: array([[ 48835.18598871, 121879.39840819, 65134.55625083, 63265.16777221,
115602.64545369, 380325.8934992 , 139537.23989081, 64199.98203852,
78349.68719258, 380649.1375447 ]])

Entrée [ ]:
```

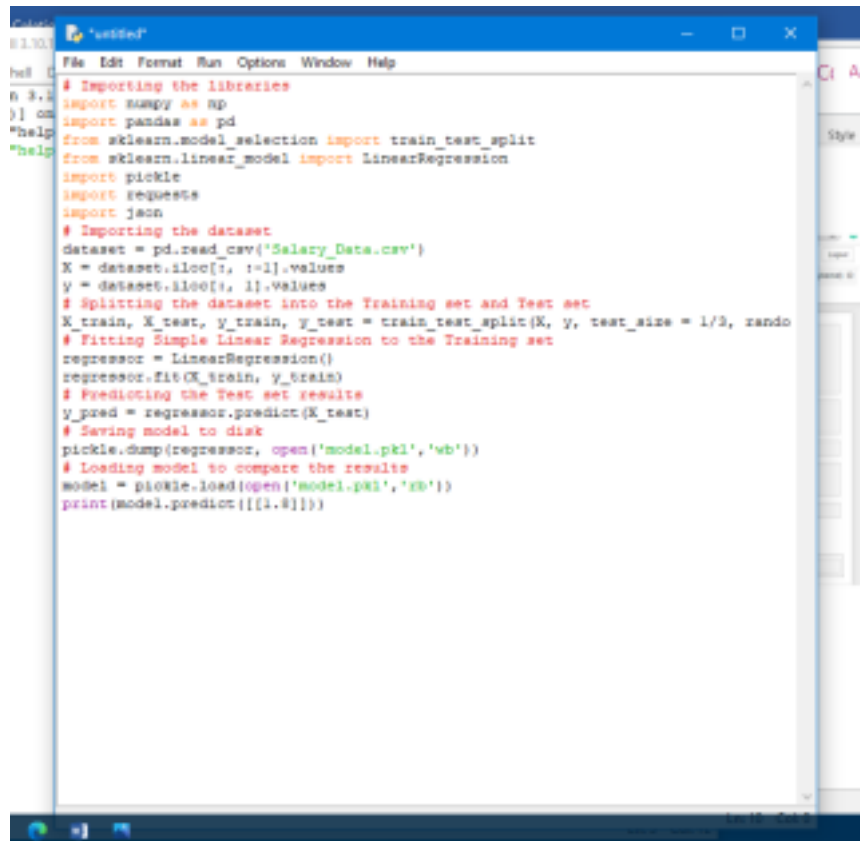
Second step :

After prediction, I save my model using pickle library.

Pickle is used for serialization and deserialization of python object structure.

The object regressor is saved as a file named model.pkl

Then I prepare model.py file for training and saving my model:



```
File Edit Format Run Options Window Help
# Importing the libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import pickle
import requests
import json

# Importing the dataset
dataset = pd.read_csv('Salary_Data.csv')
X = dataset.iloc[:, 1:2].values
y = dataset.iloc[:, 2].values

# Splitting the dataset into the Training set and Test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)

# Fitting Simple Linear Regression to the Training set
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = regressor.predict(X_test)

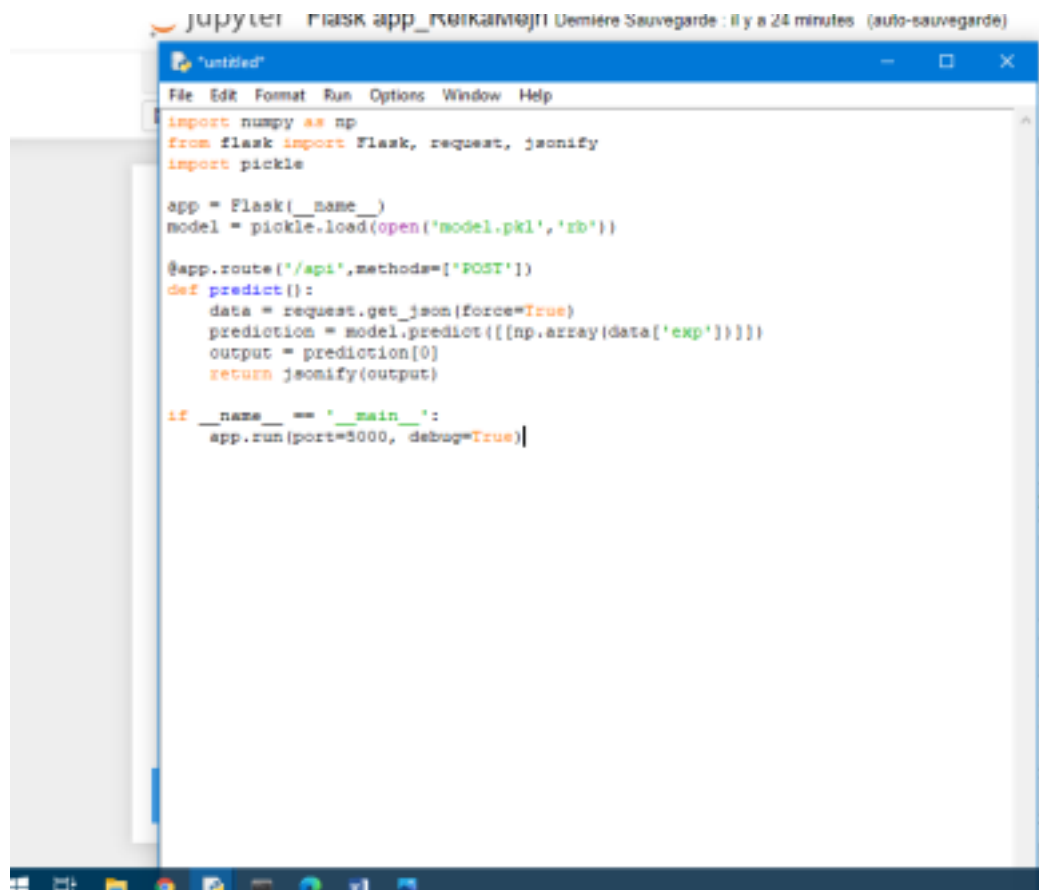
# Saving model to disk
pickle.dump(regressor, open('model.pkl', 'wb'))

# Loading model to compare the results
model = pickle.load(open('model.pkl', 'rb'))
print(model.predict([[1.5]]))
```

Step3 :

I prepare server.py file where I will use flask web framework to deal with the post request that we will get from request.py

p. 3



The screenshot shows a JupyterLab window with a file named 'untitled' open. The code is a Flask application that loads a pre-trained model from a file named 'model.pkl'. It defines a POST endpoint at '/api' that takes a JSON request, processes it using the loaded model, and returns the prediction as a JSON response. The application is configured to run on port 3000 with debug mode enabled.

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

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

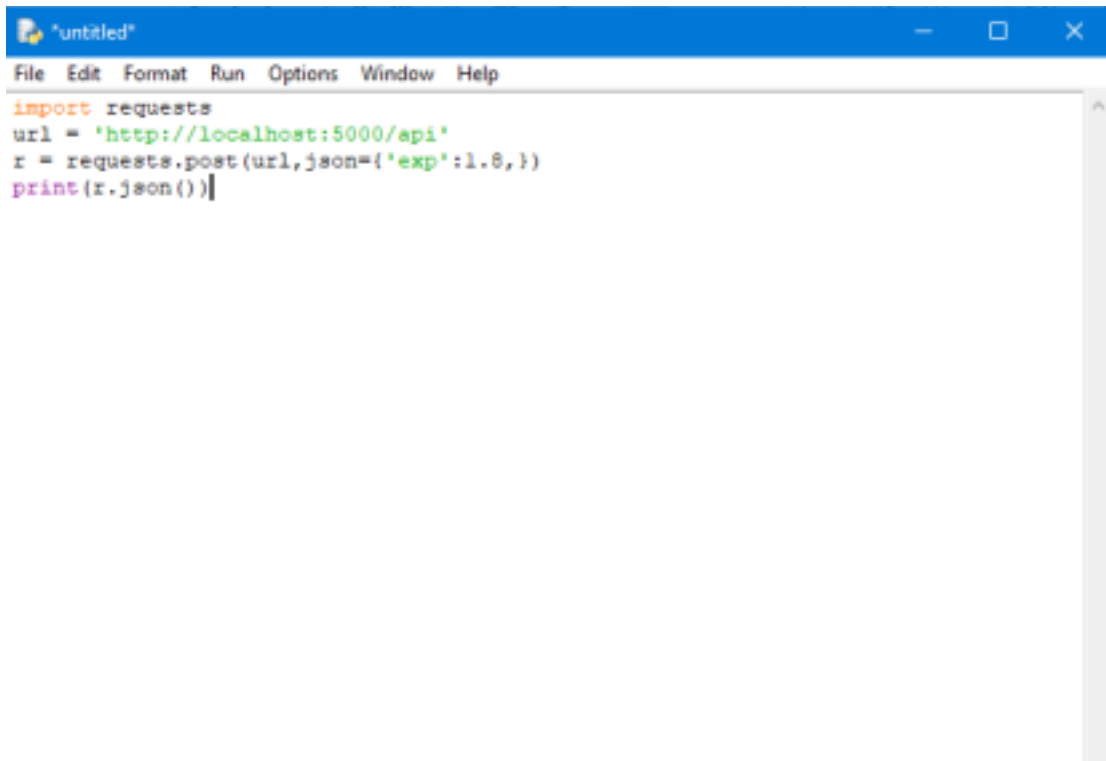
@app.route('/api', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    prediction = model.predict([np.array(data['exp'])])
    output = prediction[0]
    return jsonify(output)

if __name__ == '__main__':
    app.run(port=3000, debug=True)
```

Now our server is ready, let's add request.py file :

p. 4

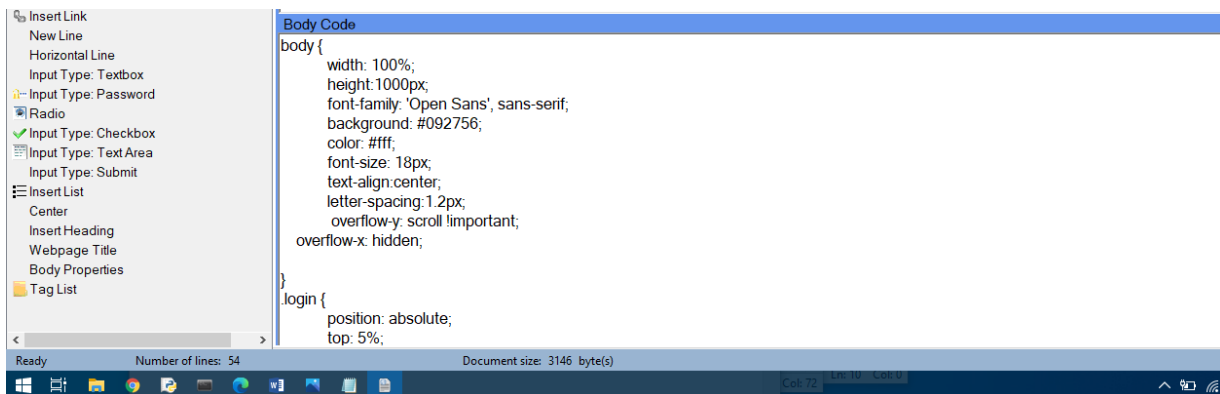




A screenshot of a Python script in an IDE window titled "untitled". The script uses the 'requests' library to perform a POST request to 'http://localhost:5000/api' with a JSON body {'exp': 1.8}. The response is printed as JSON. The IDE has a menu bar with File, Edit, Format, Run, Options, Window, and Help.

```
import requests
url = 'http://localhost:5000/api'
r = requests.post(url,json={'exp':1.8,})
print(r.json())
```

After preparing html and css file for web app.I tested my app using python app.py:

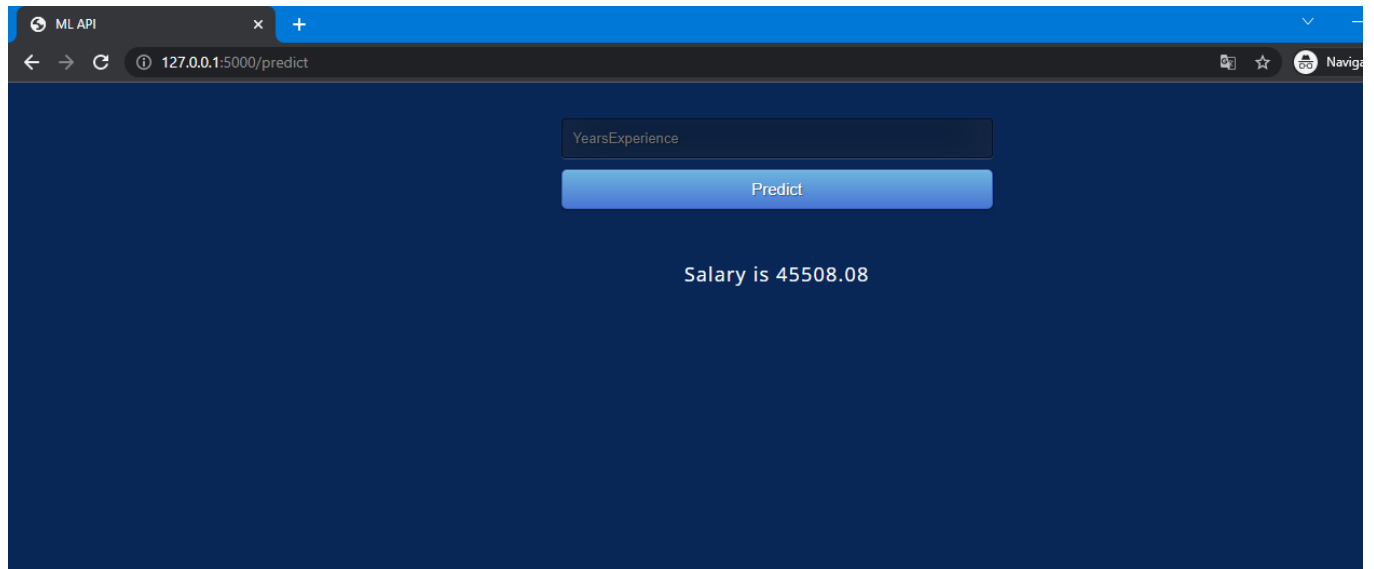


A screenshot of a web browser's developer tools. The 'Body Code' panel is active, showing CSS for the body and a .login class. The body styles include width, height, font-family, background-color, color, font-size, text-align, letter-spacing, overflow-y, and overflow-x. The .login class has position and top properties. The status bar at the bottom shows 'Ready', 'Number of lines: 54', and 'Document size: 3146 byte(s)'.

```
body {
  width: 100%;
  height: 1000px;
  font-family: 'Open Sans', sans-serif;
  background: #092756;
  color: #fff;
  font-size: 18px;
  text-align: center;
  letter-spacing: 1.2px;
  overflow-y: scroll !important;
  overflow-x: hidden;
}
.login {
  position: absolute;
  top: 5%;
}
```

p. 5

I obtain the url for the app : Running on http://127.0.0.1:5000/



All file are saved in the same file named : Flask-app_Salary prediction: