

**Name: Reeka Hazarika**

**Batch code: LISP01**

**Submission date: 22-MAR-2021**

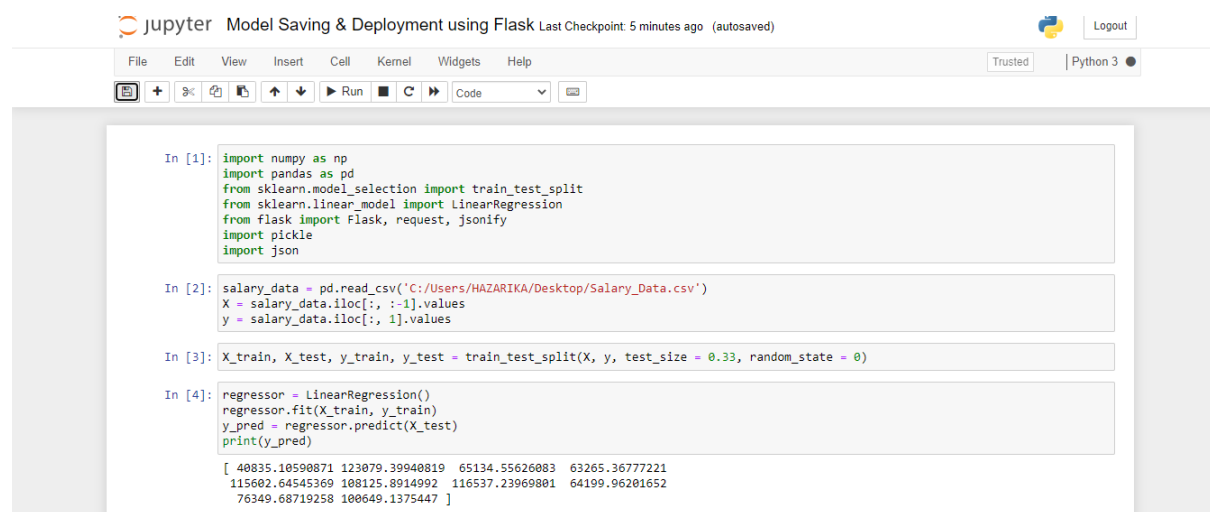
**Submitted to: Data Glacier**

# Deployment on Flask

## Step 1:

Develop ML model:

Predict the salary of an employee based on experience using Linear Regression Model.



The screenshot shows a Jupyter Notebook titled "Model Saving & Deployment using Flask". The notebook contains four code cells. The first cell imports necessary libraries: numpy, pandas, sklearn (train\_test\_split, LinearRegression), flask (Flask, request, jsonify), pickle, and json. The second cell reads a CSV file 'Salary\_Data.csv' and splits the data into features (X) and target (y). The third cell uses train\_test\_split to create training and testing sets. The fourth cell creates a LinearRegression model, fits it to the training data, and prints the predictions for the test data.

```
In [1]: import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from flask import Flask, request, jsonify
import pickle
import json

In [2]: salary_data = pd.read_csv('C:/Users/HAZARIKA/Desktop/Salary_Data.csv')
X = salary_data.iloc[:, :-1].values
y = salary_data.iloc[:, 1].values

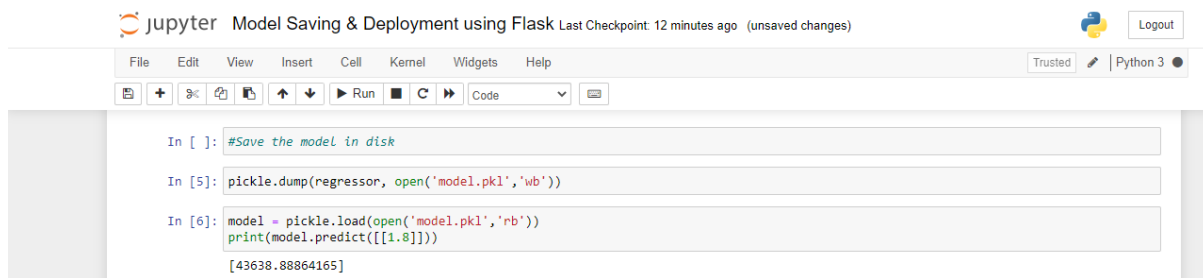
In [3]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 0)

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

[ 40835.10590871 123079.39940819  65134.55626083  63265.36777221
 115602.64545369 108125.8914992  116537.23969801  64199.96201652
 76349.68719258 100649.1375447 ]
```

## Step 2:

Saving the trained model to the disk using the *pickle* library.



A Jupyter Notebook interface titled "Model Saving & Deployment using Flask". The notebook shows three code cells. The first cell contains a comment "#Save the model in disk". The second cell contains the code to save a regressor model to a file named "model.pkl" using pickle.dump. The third cell contains the code to load the model back from "model.pkl" using pickle.load and then print the prediction for the input [1.8]. The output of the third cell is [43638.88864165].

```
In [ ]: #Save the model in disk


In [5]: pickle.dump(regressor, open('model.pkl','wb'))

In [6]: model = pickle.load(open('model.pkl','rb'))
        print(model.predict([[1.8]]))

[43638.88864165]
```

## Step 3:

### Deployment of Model



A Jupyter Notebook interface titled "Model Saving & Deployment using Flask". The notebook shows three code cells. The first cell contains a comment "#Deployment of Model". The second cell contains the code to create a Flask app and load the model from "model.pkl". The third cell contains the code to define a predict endpoint that takes JSON input, converts it to a numpy array, uses the model to predict, and returns the result as JSON. The fourth cell contains the code to run the Flask app on port 5000. The output of the fourth cell is a message indicating the app is running on http://127.0.0.1:5000/.

```
In [ ]: #Deployment of Model

In [7]: app = Flask(__name__)
        model = pickle.load(open('model.pkl','rb'))

In [8]: @app.route('/', methods=['POST'])
        def predict():
            data = request.get_json(force=True)
            prediction = model.predict([np.array(data['exp'])])
            output = prediction[0]
            return jsonify(output)

In [*]: if __name__ == '__main__':
        app.run(port=5000)

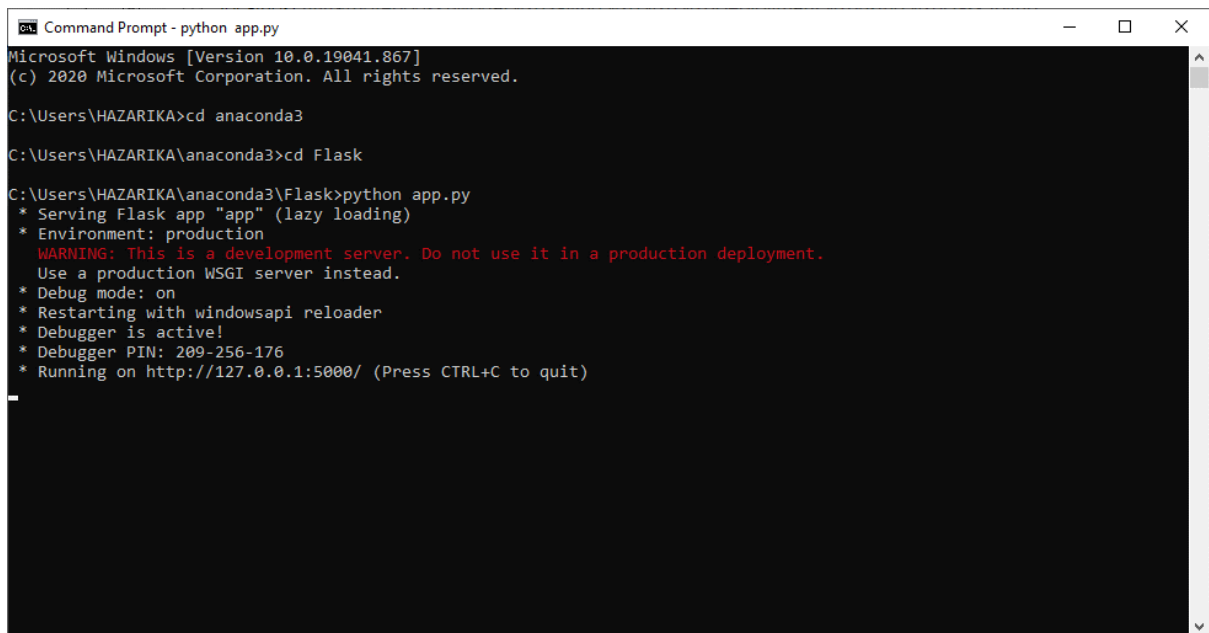
* Serving Flask app "__main__" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

- Created the instance of the *Flask()* and loaded the model.
- Bounded “/” with the method *predict()* in which predict method gets the data from the json passed by the requestor.
- *model.predict()* method takes input from the json and converts it into 2D *numpy array* the results are stored into the variable named *output*.
- Return this variable after converting it into the json object using flask's *jsonify()* method.
- Run our server by following above code section and using port 5000.

## Step 4:

Checking app.py file in CMD



```
Command Prompt - python app.py
Microsoft Windows [Version 10.0.19041.867]
(c) 2020 Microsoft Corporation. All rights reserved.

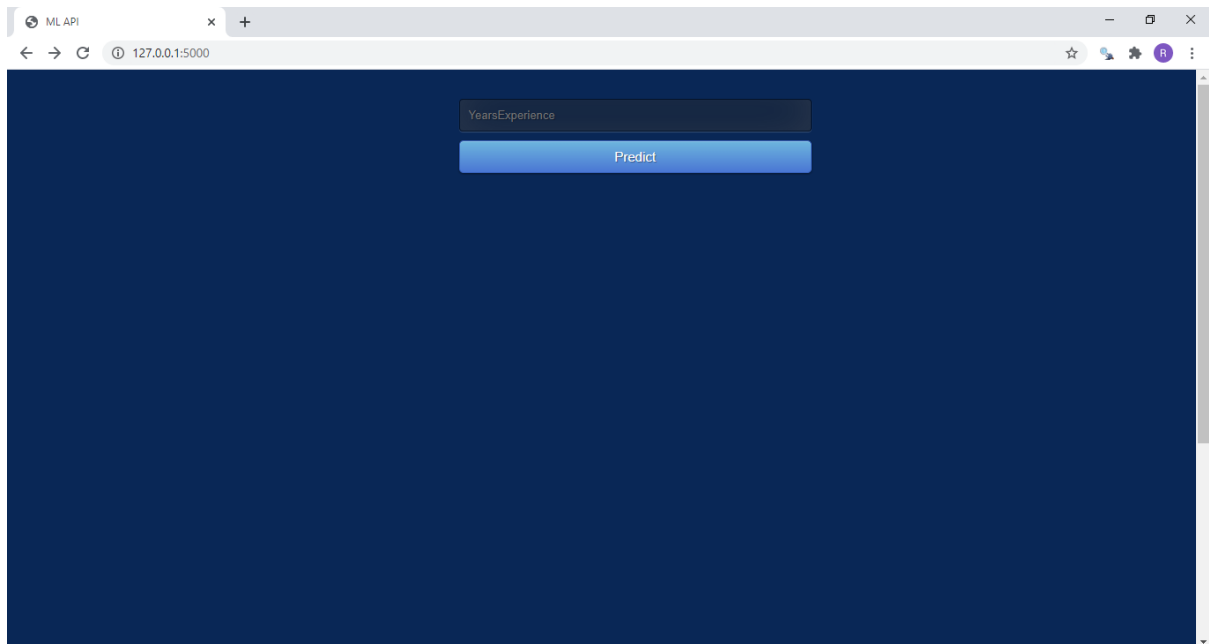
C:\Users\HAZARIKA>cd anaconda3

C:\Users\HAZARIKA\anaconda3>cd Flask

C:\Users\HAZARIKA\anaconda3\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
* Restarting with windowsapi reloader
* Debugger is active!
* Debugger PIN: 209-256-176
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

## Step 5:

Creating the Web App by typing the URL in the browser



**Thank You**

