Deployment on Flask

Name: Nonhlanhla Luphade

Batch Code: LISP01

Submission Date: 3/16/21

Submitted to: Data Glacier

Steps

1. Select toy data.
2. Create a simple model and save it.
3. Create a web app using Flask.
4. Deploy the model on Flask.
5. Web app results.

Step 1: The Data

	total_bill	sex	smoker	day	time	size	tip
0	12.16	1.0	Yes	Friday	Lunch	2	2.20
1	21.50	1.0	No	Sunday	Dinner	4	3.50
2	10.33	0.0	No	Thursday	Lunch	2	2.00
3	14.78	1.0	No	Sunday	Dinner	2	3.23
4	18.04	1.0	No	Sunday	Dinner	2	3.00
..	...	...	...	...	...	...	...
194	18.28	1.0	No	Thursday	Lunch	2	4.00
195	17.29	1.0	No	Thursday	Lunch	2	2.71
196	18.43	1.0	No	Sunday	Dinner	4	3.00
197	18.78	0.0	No	Thursday	Dinner	2	3.00
198	15.98	0.0	No	Friday	Lunch	3	3.00

[199 rows x 7 columns]

- The tip data was used for this process.
- Only 4 columns were considered to keep the model simple (total_bill, sex, size and tip)

Step 1: The Model

- A linear model was used and saved for the deployment purpose.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pickle

df = pd.read_csv('my_tipdata.csv')
pd.DataFrame(df,columns=['total_bill','sex','smoker','day','time','size','tip'])
df.sex = df.sex.astype('category')

X = df[['total_bill','sex','size']]
Y = df['tip']

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X, Y)
pickle.dump(regressor, open('model.pkl','wb'))
model = pickle.load(open('model.pkl','rb'))
```

Step 2: The html and CSS

```
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>Deployment Tutorial with Flask</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=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body style="background: #000;">
 <div class="login">
        <h1>Tip Forecasting</h1>

    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
        <input type="text" name="Total Bill" placeholder="Total bill" required="required" />
        <input type="text" name="Gender" placeholder="0 - Female 1 - Male" required="required" />
        <input type="text" name="Size" placeholder="Size of people" required="required" />
        <button type="submit" class="btn btn-primary btn-block btn-large">Predict tip </button>
    </form>

   <br>
   <br>
   {{ prediction_text }}

 </div>
</body>
</html>
```

```
html { width: 100%; height:100%; overflow:hidden; }

body {
    width: 100%;
    height:100%;
    font-family: 'Helvetica';
    background: #000;
    color: #fff;
    font-size: 24px;
    text-align:center;
    letter-spacing:1.4px;

}
.login {
    position: absolute;
    top: 40%;
    left: 50%;
    margin: -150px 0 0 -150px;
    width:400px;
    height:400px;
}

.login h1 { color: #fff; text-shadow: 0 0 10px rgba(0,0,0,0.3); letter-spacing:1px; text-align:center; }

input {
    width: 100%;
    margin-bottom: 10px;
    background: rgba(0,0,0,0.3);
    border: none;
    outline: none;
    padding: 10px;
    font-size: 13px;
    color: #fff;
    text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
    border: 1px solid rgba(0,0,0,0.3);
```

Step 3: The App

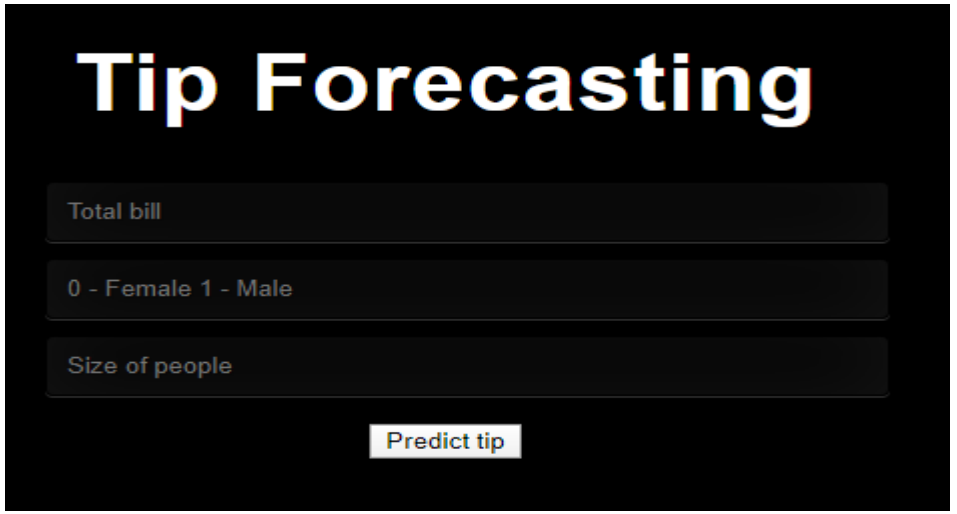
```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

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():

    int_features = [float(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='Tip should be $ {}'.format(output))


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

This is used to create a web app using Flask.

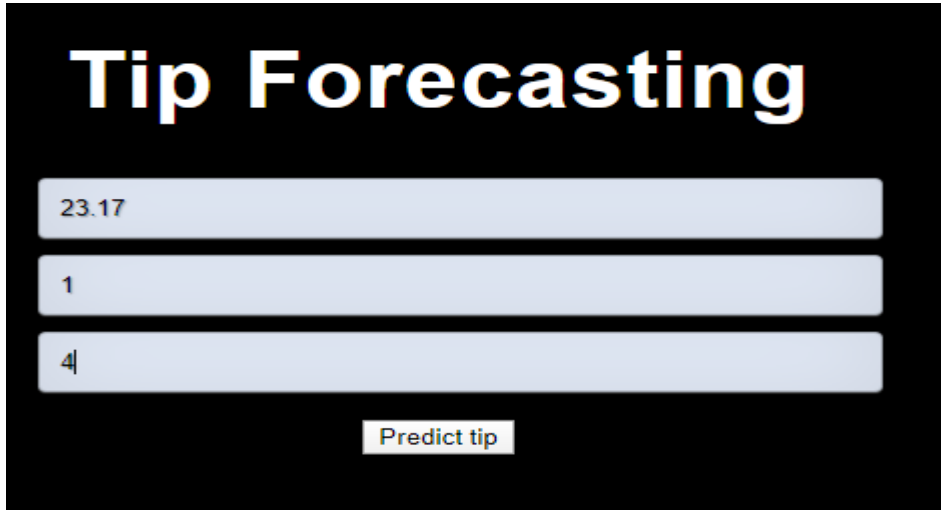
Step 4-5: Deployment

```
 * 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: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 268-131-083
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [16/Mar/2021 14:13:38] "←[37mGET / HTTP/1.1←[0m" 200 -
127.0.0.1 - - [16/Mar/2021 14:14:58] "←[37mPOST /predict HTTP/1.1←[0m" 200 -
127.0.0.1 - - [16/Mar/2021 14:15:14] "←[37mPOST /predict HTTP/1.1←[0m" 200 -
127.0.0.1 - - [16/Mar/2021 14:15:21] "←[37mPOST /predict HTTP/1.1←[0m" 200 -
```

Web app



Try examples



Predict

The web app is created and deployed.