Week 4: Model Deployment on Flask

Internship Batch LISUM37

Submitted by: Mark Maina

Data Information
Tabular data details: Medical_insurance

Total number of observations	2772
Total number of files	1
Total number of features	7
Base format of the file	.csv
Size of the data	0.112 MB

Introduction
The aim of the project was to deploy a Linear Regression model that would predict the charges insurance client would pay given certain variables.

Building the Model

```
In [1]: # import necessary libraries
        import pandas as pd
        import pickle
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression

In [2]: # import data
        data = pd.read_csv('Medical_insurance.csv')

In [3]: data.head()
```

Out[3]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

```
In [4]: data.describe()
```

Out[4]:

	age	bmi	children	charges
count	2772.000000	2772.000000	2772.000000	2772.000000
mean	39.109668	30.701349	1.101732	13261.369959
std	14.081459	6.129449	1.214806	12151.768945
min	18.000000	15.960000	0.000000	1121.873900
25%	26.000000	26.220000	0.000000	4687.797000
50%	39.000000	30.447500	1.000000	9333.014350
75%	51.000000	34.770000	2.000000	16577.779500
max	64.000000	53.130000	5.000000	63770.428010

In [6]:
```
# Data Preprocessing
from sklearn.preprocessing import LabelEncoder
```

In [7]:
```
encode = LabelEncoder()
```

In [8]:
```
data['sex'] = encode.fit_transform(data['sex'])
data['smoker'] = encode.fit_transform(data['smoker'])
data['region'] = encode.fit_transform(data['region'])
```

In [9]:
```
data
```

Out[9]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	3	16884.92400
1	18	1	33.770	1	0	2	1725.55230
2	28	1	33.000	3	0	2	4449.46200
3	33	1	22.705	0	0	1	21984.47061
4	32	1	28.880	0	0	1	3866.85520
...	...	...	...	...	...	...	...
2767	47	0	45.320	1	0	2	8569.86180
2768	21	0	34.600	0	0	3	2020.17700
2769	19	1	26.030	1	1	1	16450.89470
2770	23	1	18.715	0	0	1	21595.38229
2771	54	1	31.600	0	0	3	9850.43200

2772 rows × 7 columns

Create the x and y data necessary for the regression model

In [15]:
```
x = data.drop('charges', axis=1)
y = data['charges']
```

In [16]:
```
x
```

Out[16]:

	age	sex	bmi	children	smoker	region
0	19	0	27.900	0	1	3
1	18	1	33.770	1	0	2
2	28	1	33.000	3	0	2
3	33	1	22.705	0	0	1
4	32	1	28.880	0	0	1
...	...	...	...	...	...	...
2767	47	0	45.320	1	0	2
2768	21	0	34.600	0	0	3
2769	19	1	26.030	1	1	1
2770	23	1	18.715	0	0	1
2771	54	1	31.600	0	0	3

2772 rows × 6 columns

In [17]:
```
y
```

Out[17]:
```
0       16884.92400
1        1725.55230
2        4449.46200
3       21984.47061
4        3866.85520
           ...
2767     8569.86180
2768     2020.17700
2769    16450.89470
2770    21595.38229
2771     9850.43200
Name: charges, Length: 2772, dtype: float64
```

Save model and pickle it

```
In [15]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=101)

In [16]: print(x_train.shape, x_test.shape)
         print(y_train.shape, y_test.shape)

         (1940, 6) (832, 6)
         (1940,) (832,)

In [17]: regression = LinearRegression()
         regression.fit(x_train, y_train)

Out[17]: LinearRegression()

In [18]: pickle.dump(regression, open('model.pickle', 'wb'))
```

Create the Index HTML file

```
<!DOCTYPE html>


<head>
    <meta charset="UTF-8">
    <title>ML API</title>
    <link href='https://fonts.googleapis.com/css?family=Pacifico' rel="stylesheet" type="text/css">
<link href='https://fonts.googleapis.com/css?family=Arimio' rel="stylesheet" type="text/css">
<link href='https://fonts.googleapis.com/css?family=Mind: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 Insurance Charges</h1>
<!--    Main input for receiving query to our ML-->
    <form action="/predict" method="post">
        <input type='text' name='age' placeholder='income' required='required'/>
        <input type='text' name='sex' placeholder='sex' required='required'/>
        <input type='text' name='bmi' placeholder='bmi' required='required'/>
        <input type='text' name='children' placeholder='children' required='required'/>
        <input type='text' name='smoker' placeholder='smoker' required='required'/>
        <input type='text' name='region' placeholder='region' required='required'/>


        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>
    <br>
    <br>
    {{ prediction_text}}
    </div>

</body>
```

Create the Flask app

```python
# Using flask app to make an api
# import necessary libraries
from flask import Flask, jsonify, request, render_template
import pickle
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# create a Flask app
app = Flask(__name__)


@app.route('/')  # , methods=['GET', 'POST'])
def home():
    return render_template('index.html')


@app.route('/predict', methods=['POST'])
def price_predict():
    model = pickle.load(open('model.pickle', 'rb'))
    age = request.form.get('age')
    sex = request.form.get('sex')
    bmi = request.form.get('bmi')
    children = request.form.get('children')
    smoker = request.form.get('smoker')
    region = request.form.get('region')
    encoder = LabelEncoder()

    test_df = pd.DataFrame({'age': [age], 'sex': [sex], 'bmi': [bmi], 'children': [children], 'smoker': [smoker],
                            'region': [region]})
    test_df['sex'] = encoder.fit_transform(test_df['sex'])
    test_df['smoker'] = encoder.fit_transform(test_df['smoker'])
    test_df['region'] = encoder.fit_transform(test_df['region'])
    pred_price = model.predict(test_df)
    output = round(pred_price[0], 2)
    return render_template('index.html', prediction_text='Charges should be ${}'.format(output))


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

Prediction Model