Week4: Deployment on Flask

Name: Sijing Liu
Batch code: LISUM13: 30
Submission date: 27-Sep-2022
Submitted to: Data Glacier

Toy data details: demand

Total number of observations	26
Total number of files	1
Total number of features	6
Base format of the file	.csv
Size of the data	925+ bytes

Each step of deployment

Install Flask

pip install flask

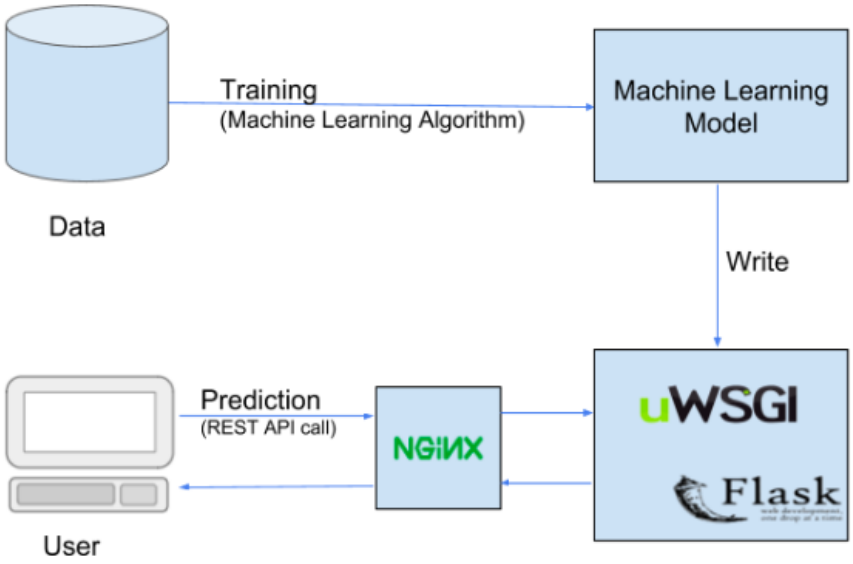
```
C:\Users\pc>flask --version
Flask 1.0.2
Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)]
```

Observe toy data

City	Number of weekly riders	Price per week	Population of city	Monthly income of riders	Average parking rates per month
1	192000	15	1800000	5800	50
2	190400	15	1790000	6200	50
3	191200	15	1780000	6400	60
4	177600	25	1778000	6500	60
5	176800	25	1750000	6550	60
6	178400	25	1740000	6580	70
7	180800	25	1725000	8200	75
8	175200	30	1725000	8600	75
9	174400	30	1720000	8800	75
10	173920	30	1705000	9200	80
11	172800	30	1710000	9630	80
12	163200	40	1700000	10570	80
13	161600	40	1695000	11330	85
14	161600	40	1695000	11600	100
15	160800	40	1690000	11800	105
16	159200	40	1630000	11830	105
17	148800	65	1640000	12650	105
18	115696	102	1635000	13000	110
19	147200	75	1630000	13224	125
20	150400	75	1620000	13766	130
21	152000	75	1615000	14010	150
22	136000	80	1605000	14468	155
23	126240	86	1590000	15000	165
24	123888	98	1595000	15200	175
25	126080	87	1590000	15600	175
26	151680	77	1600000	16000	190

- The toy data is the classic sample used to build regression model and predict transit demands.
- The 2~5 columns (`'Price per week'`, `'Population of city'`, `'Monthly income of riders'`, `'Average parking rates per month'`) are supposed to predict the 1st column (`'Number of weekly riders'`).

Set up the project workflow



Pipeline for deployment of a Machine Learning model

- ***model.py*** — This contains code for the regression model to predict transit demands (`'Number of weekly riders'`) based on the four attributions (`'Price per week'`, `'Population of city'`, `'Monthly income of riders'`, `'Average parking rates per month'`).
- ***app.py*** — This contains Flask APIs that receives sales details through GUI or API calls, computes the predicted value based on our model and returns it.
- request.py — This uses requests module to call APIs defined in app.py and displays the returned value.
- HTML/CSS — This contains the HTML template and CSS styling to allow user to enter data detail and displays the predicted 'Number of weekly riders'.

Snapshots

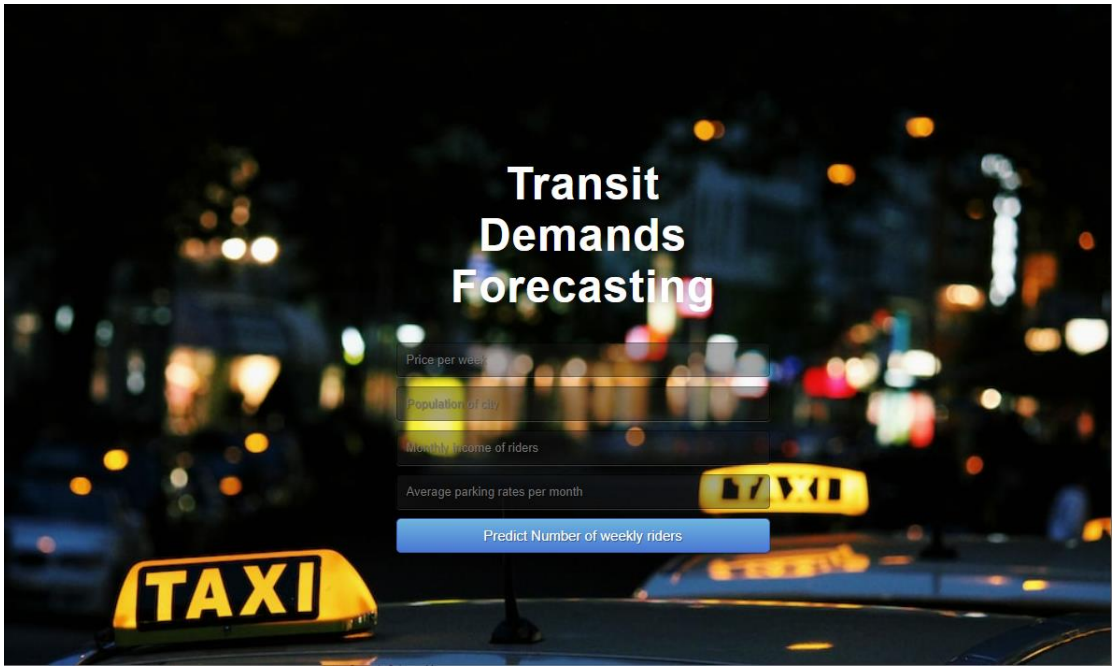
- ***model.py***

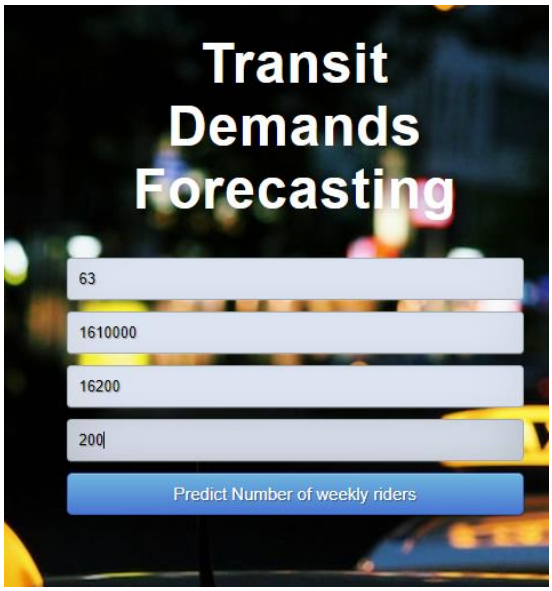
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pickle
from sklearn.linear_model import LinearRegression

dataset = pd.read_csv('demand.csv')
```

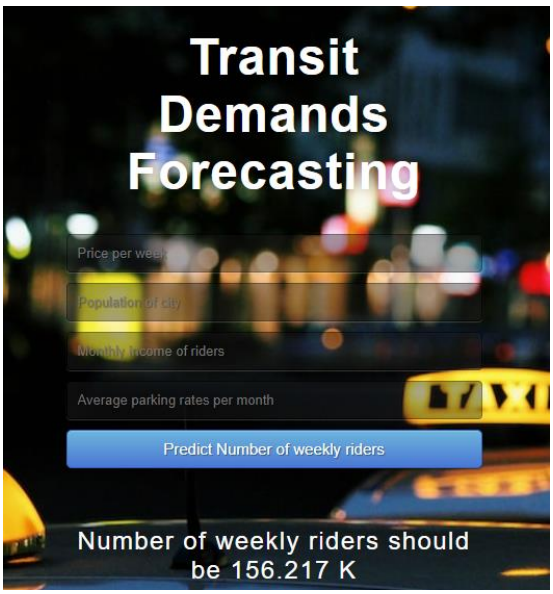
```
X = dataset.iloc[:, 2:6]
Y = dataset.iloc[:, 1]  # predicted column

# build regression model
regressor = LinearRegression()
regressor.fit(X, Y)

# save the simple model
pickle.dump(regressor, open('model.pkl','wb'))

# load the simple model
model = pickle.load(open('model.pkl','rb'))

print(model.predict([[63, 1610000, 16200, 200]]))
```

- ***app.py***

```
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 = [int(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    output = round(prediction[0])

    return render_template('index.html', prediction_text='Number of weekly riders should be {} K'.format(output/1000))

@app.route('/results', methods=['POST'])
def results():

    data = request.get_json(force=True)
    prediction = model.predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)

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







References

https://towardsdatascience.com/how-to-easily-deploy-machine-learningmodels-using-flask-b95af8fe34d4