

Week 4 assignment: Deployment on Flask

Name: Roger Burek-Bors

Batch code: LICAN01

Submission date: March 21, 2021

Submitted to: Data Glacier

1. Introduction

I am fan of cars therefore I decided to use for this week assignment “FuelConsumptionCo2.csv” which is a 1000-point data set about modern cars. This data set contains such features like:

- model of car,
- year of introduction,
- engine size,
- cylinder,
- fuel consumption,
- CO2 emissions.

My aim was to create a prediction model that would help to determine CO2 emissions when it comes to engine size.

2. Building the model

https://github.com/rodbergerrone/VC/blob/rbb/Model_deploy/FuelConsumptionCo2_Regr.ipynb

- a) I used Jupyter Notebook for “FuelConsumptionCo2.csv” analysis. I determined 4 most promising features for CO2 emissions predictions:

Choosing data for modelling

```
4 cdf = df[['ENGINE SIZE', 'CYLINDERS', 'FUEL CONSUMPTION_COMB', 'CO2 EMISSIONS']]
cdf.head(5)
```

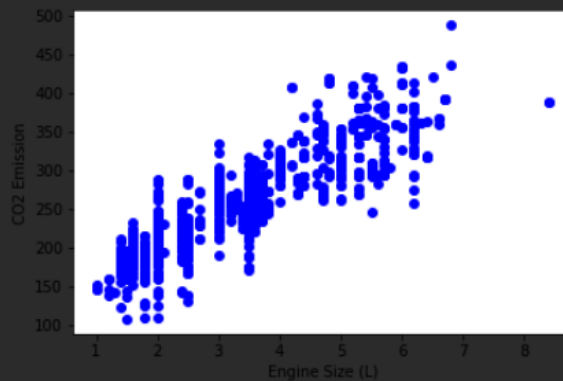
	ENGINE SIZE	CYLINDERS	FUEL CONSUMPTION_COMB	CO2 EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244

- b) Analysis led to conclusions that features like engine size, cylinders and fuel consumption have almost linear correlation with CO2 emissions feature therefore a linear regression model could be built. To keep this assignment simple I decided that model will be predicting CO2 emission based on only one feature – engine size.

```

7 plt.scatter(cdf.ENGINESIZE, cdf.CO2EMISSIONS, color='blue')
  plt.xlabel("Engine Size (L)")
  plt.ylabel("CO2 Emission")
  plt.show()

```



c) Building the model was achieved as on following pictures with help of Scikit-learn library:

```

9 msk = np.random.rand(len(df)) < 0.8
  train = cdf[msk]
  test = cdf[~msk]

```

```

10 plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS, color='blue')
   plt.xlabel("Engine Size (L)")
   plt.ylabel("CO2 Emission")
   plt.show()

```

```

11 from sklearn import linear_model
   regr = linear_model.LinearRegression()
   train_x = np.asanyarray(train[['ENGINESIZE']])
   train_y = np.asanyarray(train[['CO2EMISSIONS']])
   regr.fit (train_x, train_y)
   print('Coefficients: ', regr.coef_)
   print('Intercept: ',regr.intercept_)

```

```

Coefficients: [[38.70010101]]
Intercept: [126.4800893]

```

```

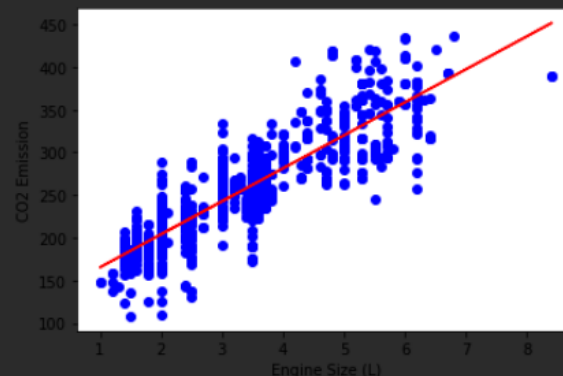
12 plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS, color='blue')
   plt.plot(train_x, regr.coef_[0][0]*train_x + regr.intercept_[0], '-r')
   plt.xlabel("Engine Size (L)")
   plt.ylabel("CO2 Emission")

```

```

12 Text(0, 0.5, 'CO2 Emission')

```



d) I performed model evaluation to see how prediction behave in correlation to data in "FuelConsumptionCo2.csv" and on leaflets of car dealerships. It worked really fine. I also checked metrics available in Scikit-learn library:

```

13 print("For 2 L engine:", regr.predict(np.array([[2]])))
    print("For 5 L engine:", regr.predict(np.array([[5]])))

    For 2 L engine: [[203.88029131]]
    For 5 L engine: [[319.98059433]]

14 from sklearn.metrics import r2_score

    test_x = np.asanyarray(test[['ENGINE_SIZE']])
    test_y = np.asanyarray(test[['CO2EMISSIONS']])
    test_y_ = regr.predict(test_x)

    print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_ - test_y)))
    print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
    print("R2-score: %.2f" % r2_score(test_y, test_y_))

    Mean absolute error: 22.11
    Residual sum of squares (MSE): 851.73
    R2-score: 0.80

```

Coefficient of determination is 0.8 which in my opinion is quite good.

- e) Then I serialized the model with Pickle library as "pickle_model.pkl":

```

15 with open('pickle_model.pkl', 'wb') as f:
    pickle.dump(regr, f)

```

3. Building Flask backend

https://github.com/rodbergerrone/VC/blob/rbb/Model_deploy/deploy_flask.py

I used Pycharm to develop Python code for backend:

```

deploy_flask.py
1  import numpy as np
2  from flask import Flask, request, render_template
3  import pickle
4
5
6  app = Flask(__name__)
7  with open('pickle_model.pkl', 'rb') as f:
8      model = pickle.load(f)
9
10
11 @app.route('/')
12 def home():
13     return render_template('index.html')
14
15
16 @app.route('/predict', methods=['POST'])
17 def predict():
18     int_features = [float(x) for x in request.form.values()]
19     final_features = [np.array(int_features)]
20     prediction = model.predict(final_features)
21
22     prediction = round(prediction[0][0], 2)
23
24     return render_template('index.html', prediction_text='CO2 emission for this size of engine should be {} G/KM'
25                           .format(prediction))
26
27
28 if __name__ == "__main__":
29     app.run(port=5000, debug=True)
30

```

4. Building Flask frontend

https://github.com/rodbergerrone/VC/blob/rbb/Model_deploy/templates/index.html

I used Pycharm to develop Python code for frontend:

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>C02 Emission Prediction</title>
6      <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
7      <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
8      <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
9      <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
10     <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
11
12 </head>
13
14 <body style="background: #050;">
15     <div class="login">
16         <h1>C02 Emission Prediction Based On Car Engine Size</h1>
17
18         <!-- Main Input For Receiving Query to our ML -->
19         <form action="{{ url_for('predict') }}" method="post">
20             <input type="text" name="car engine size" placeholder="car engine size" required="required" />
21             <button type="submit" class="btn btn-primary btn-block btn-large">Predict C02 Emission</button>
22         </form>
23
24         <br>
25         <br>
26         {{ prediction_text }}
27
28     </div>
29 </body>
30 </html>

```

5. Performing predictions on local network with Flask

```

Run: deploy_flask
C:\Users\RBB\miniconda3\envs\Glacier\python.exe C:/Users/RBB/OneDrive/Python/Github/VC/Model_deploy/deploy_flask.py
* Serving Flask app "deploy_flask" (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: 300-683-121
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

```

I am presenting on following two screenshots:

- frontend of my predictor – the result: 188.4 G/KM is for 1.6 L engine
- specification of Toyota Yaris GR – the result: 186 G/KM is for 1.6 L engine

CO2 Emission Prediction

Nowa Toyota GR Yaris | Wyposażenie

127.0.0.1:5000/predict

CO2 Emission Prediction Based On Car Engine Size

1.6

Predict CO2 Emission

CO2 emission for this size of engine should be 188.4 G/KM

Nowa Toyota GR Yaris | Wyposażenie

toyota.pl/world-of-toyota/toyotagazooracing/gr-yaris/specyfikacja

Samochody Oferty specjalne Oferta dla firm Serwis i akcesoria **Świat Toyoty** Internetowy Salon Toyoty

Nowa Toyota GR Yaris **Specyfikacja** Wycigowe DNA Dziedzictwo

SPECYFIKACJE

SILNIK	1,6 LITRA, 3-CYLINDROWY TURBODOŁADOWANY
PRZEKŁADNIA	6-BIEGOWA, MANUALNA
RODZAJ NAPĘDU	NAPĘD NA 4 KOŁA
MASA (KG)	1280
ŚREDNIE ZUŻYCIE PALIWA (L/100 KM)	8,2*
MOC SILNIKA (KM)	261
MOMENT OBROTOWY (NM)	360
PRZYSPIESZENIE (0-100 KM/H)	5,5 s
PRĘDKOŚĆ MAKSYMALNA (KM/H)	230
EMISJA CO₂ (G/KM)	186*

