Deployment on Flask

Build and Deploy a car price prediction Machine Learning Model on Flask

Import Data

The first move is to import a data to train a model

Importin data

[10]:
```
import numpy as np
import pandas as pd
import sklearn
data = pd.read_csv(r'C:\Users\m.m pc\PycharmProjects\flask\car.csv')
data.head()
```

[10]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

Sorting the data

sorting the Data

[6]:
```
data = data.drop(['Car_Name'], axis=1)
data['current_year'] = 2020
data['no_year'] = data['current_year'] - data['Year']
data = data.drop(['Year','current_year'],axis = 1)
data = pd.get_dummies(data,drop_first=True)
data = data[['Selling_Price','Present_Price','Kms_Driven','no_year','Owner','Fuel_Type_Diesel','Fuel_Type_Petrol',
'Seller_Type_Individual','Transmission_Manual']]
data
```

[6]:

	Selling_Price	Present_Price	Kms_Driven	no_year	Owner	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	3.35	5.59	27000	6	0	0	1	0	1
1	4.75	9.54	43000	7	0	1	0	0	1
2	7.25	9.85	6900	3	0	0	1	0	1
3	2.85	4.15	5200	9	0	0	1	0	1
4	4.60	6.87	42450	6	0	1	0	0	1
...	...	...	...	...	...	...	...	...	...
296	9.50	11.60	33988	4	0	1	0	0	1
297	4.00	5.90	60000	5	0	0	1	0	1
298	3.35	11.00	87934	11	0	0	1	0	1
299	11.50	12.50	9000	3	0	1	0	0	1
300	5.30	5.90	5464	4	0	0	1	0	1

301 rows × 9 columns

Remove the correlated features

The data. corr() will give you an intuition on the correlation between all attributes in the dataset. More correlated features can be removed since they can lead to overfitting of the model.

remove the correlated features

```
[7]: data.corr()
```

[7]:

	Selling_Price	Present_Price	Kms_Driven	no_year	Owner	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
Selling_Price	1.000000	0.878983	0.029187	-0.236141	-0.088344	0.552339	-0.540571	-0.550724	-0.367128
Present_Price	0.878983	1.000000	0.203647	0.047584	0.008057	0.473306	-0.465244	-0.512030	-0.348715
Kms_Driven	0.029187	0.203647	1.000000	0.524342	0.089216	0.172515	-0.172874	-0.101419	-0.162510
no_year	-0.236141	0.047584	0.524342	1.000000	0.182104	-0.064315	0.059959	0.039896	-0.000394
Owner	-0.088344	0.008057	0.089216	0.182104	1.000000	-0.053469	0.055687	0.124269	-0.050316
Fuel_Type_Diesel	0.552339	0.473306	0.172515	-0.064315	-0.053469	1.000000	-0.979648	-0.350467	-0.098643
Fuel_Type_Petrol	-0.540571	-0.465244	-0.172874	0.059959	0.055687	-0.979648	1.000000	0.358321	0.091013
Seller_Type_Individual	-0.550724	-0.512030	-0.101419	0.039896	0.124269	-0.350467	0.358321	1.000000	0.063240
Transmission_Manual	-0.367128	-0.348715	-0.162510	-0.000394	-0.050316	-0.098643	0.091013	0.063240	1.000000

Slicing the data into training and test set and remove the less important features from the data.

The extratressregressor library allows you to view feature importances and thereby remove the less important features from the data.

slicing the data into training and test set

```
[8]: x = data.iloc[:,1:]
     y = data.iloc[:,0]
```

remove the less important features from the data.

```
[12]: from sklearn.ensemble import ExtraTreesRegressor
      model = ExtraTreesRegressor()
      model.fit(x,y)
```

```
[12]: ExtraTreesRegressor()
```

```
[13]: model.feature_importances_
```

```
[13]: array([0.40893287, 0.03984788, 0.07595112, 0.00047486, 0.21118402,
             0.01658986, 0.11840909, 0.1286103 ])
```

Train Test Split and Training the model

Train Test Split

[17]:
```
from sklearn.model_selection import train_test_split    #importing train test split module
x_train, x_test,y_train,y_test = train_test_split(x,y,random_state=0,test_size=0.2)
```

Training the Model

[23]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV
model = RandomForestRegressor()
hyp = RandomizedSearchCV(estimator = model,
                         param_distributions=grid,
                         n_iter=10,
                        scoring= 'neg_mean_squared_error'
                         ,verbose = 2,
                         random_state = 42,n_jobs = 1)
hyp.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=   0.9s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.8s remaining:    0.0s
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=   0.8s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=   0.8s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=   0.8s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=   0.8s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total=   1.0s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total=   1.0s
```

finally use the model to predict the test dataset and Pack the model into the pickle file

finally use the model to predict the test dataset.

[24]:
```
y_pred = hyp.predict(x_test)
y_pred
```

[24]:
```
array([ 7.00332213,  0.51701732,  4.93121616,  8.34207497, 12.48695388,
        5.25748867,  3.3319152 ,  0.42833513,  3.90721038,  4.99797958,
        2.82710795,  0.65797005,  5.11006052,  7.25768771,  7.42243145,
       12.59097515,  7.03719792,  4.17415471,  0.48067769,  1.31344204,
        3.28061344,  5.19783858,  5.40880026, 10.43579343,  0.2327306 ,
        0.68891315,  0.32229012,  0.68241682,  0.50731348,  4.86556887,
        2.86720738,  5.81520185,  0.5167291 ,  7.1315714 ,  3.26482736,
        1.15145237,  5.75214611,  5.48952856,  0.24765779,  7.63030308,
        7.62052953, 22.05866468,  5.06851892,  4.55350907,  5.59604493,
       10.31149403,  0.25138744,  0.76067367,  5.39916615,  6.83840838,
        6.71226402,  2.98254914,  5.32051079, 22.05866468,  1.15145237,
        1.15145237,  0.3948337 ,  2.75052023,  3.65304387,  2.53973585,
        4.59412931])
```

pack the model into the pickle file

[26]:
```
import pickle
file = open("file.pkl", "wb")
pickle.dump(hyp, file)
```

Deploy the model on flask

Seting up a Flask project and loading the trained model

```
from flask import Flask,render_template,request
import pickle

from sklearn.preprocessing import StandardScaler
app = Flask(__name__)
model = pickle.load(open('file.pkl','rb'))

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

standard_to = StandardScaler()
```

Deploy the model on flask

Finishing up the predict method to predict the car price

```
@app.route('/predict',methods = ['POST'])
def predict():
    Fuel_Type_Diesel =0
    if request.method == 'POST':
        Year = int(request.form['Year'])
        Present_Price = float(request.form['Present_Price'])
        Kms_Driven = int(request.form['Kms_Driven'])
        Owner = int(request.form['Owner'])
        Fuel_Type_Petrol = request.form['Fuel_Type_Petrol']
        if(Fuel_Type_Petrol == 'Petrol'):
            Fuel_Type_Diesel = 0
            Fuel_Type_Petrol = 1

        elif(Fuel_Type_Diesel=='Diesel'):
            Fuel_Type_Petrol = 0
            Fuel_Type_Diesel = 1
        else:
            Fuel_Type_Petrol = 0
            Fuel_Type_Diesel = 0

        Year = 2020 - Year
        Seller_Type_Individual = request.form['Seller_Type_Individual
        if(Seller_Type_Individual=='Individual'):
            Seller_Type_Individual =1
        else:
            Seller_Type_Individual = 0

        Transmission_Manual = request.form['Transmission_Manual']
        if(Transmission_Manual == 'Manual'):
            Transmission_Manual = 1
```

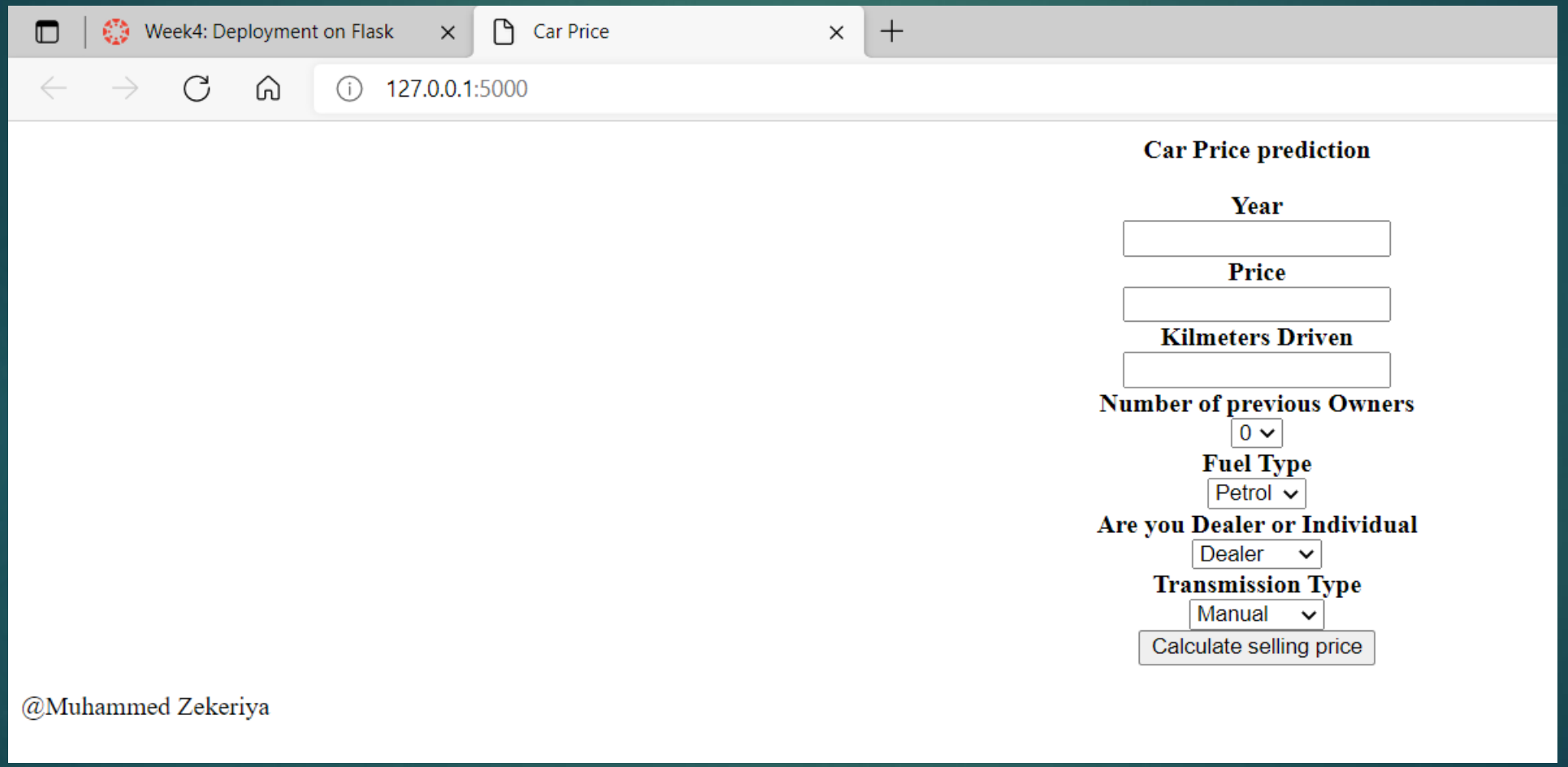
```
    Transmission_Manual = request.form['Transmission_Manual']
    if(Transmission_Manual == 'Manual'):
        Transmission_Manual = 1
    else:
        Transmission_Manual = 0

    prediction = model.predict([[Present_Price,Kms_Driven,Owner,Year,Fuel_Type_Diesel,Fuel_Type_Petrol,Seller_Type_Individual
    output = round(prediction[0],2)

    if output<0:
        return render_template('index.html',prediction_text='Sorry! You cannot sell this car')
    else:
        return render_template('index.html', prediction_text='You can sell this car at Rs.{} '.format(output))

else:
    return render_template('index.html')
```

Deploy the model on flask

Finishing up the HTML code

```
<p align="center" class="heading"><b>Car Price prediction</b></p>
<div align="center">
    <form action="{{ url_for('predict')}}" method="POST">
        <div class="b"><b>Year</b><br>
            <input class="a" type="Number" name ="Year" required="required"></div>

        <div class="b"><b>Price</b><br>
            <input type="Number" name="Present_Price" required="required" class="a"

        <div class="b"><b>Kilmeters Driven</b><br>
            <input type="Number" name="Kms_Driven" required="required" class="a"></

        <div class="b"><b>Number of previous Owners</b><br>
            <select name="Owner" class="a">
            <option value="0">0</option>
            <option value="1">1</option>
            <option value="2">2</option>
            <option value="3">3</option></select></div>

        <div class="b"><b>Fuel Type</b><br>
            <select name="Fuel_Type_Petrol" class="a">
            <option value="Petrol">Petrol</option>
            <option value="Diesel">Diesel</option>
            <option value="CNG">CNG</option></select></div>

        <div class="b"><b>Are you Dealer or Individual</b><br>
            <select name="Seller_Type_Individual" class="a">
            <option value="Dealer">Dealer</option>
            <option value="Individual">Individual</option></select></div>
```

The result


Week4: Deployment on Flask
Car Price
127.0.0.1:5000
Car Price prediction
Year
Price
Kilmeters Driven
Number of previous Owners
0
Fuel Type
Petrol
Are you Dealer or Individual
Dealer
Transmission Type
Manual
Calculate selling price
@Muhammed Zekeriya

Prediction example