Project2

January 22, 2021

Machine Learning

Mohsen Iranmanesh

98155002

---

1 Part 1

First of all, we import data to get an insight

[75]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_excel('ToyotaCorolla.xls', sheet_name='data')
df.head()
```

[75]:

	Id	Model	Price	Age_08_04	\
0	1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	
1	2	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	
2	3	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	
3	4	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	14950	26	
4	5	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	

	Mfg_Month	Mfg_Year	KM	Fuel_Type	HP	Met_Color	…	Powered_Windows	\
0	10	2002	46986	Diesel	90	1	…	1	
1	10	2002	72937	Diesel	90	1	…	0	
2	9	2002	41711	Diesel	90	1	…	0	
3	7	2002	48000	Diesel	90	0	…	0	
4	3	2002	38500	Diesel	90	0	…	1	

	Power_Steering	Radio	Mistlamps	Sport_Model	Backseat_Divider	\
0	1	0	0	0	1	
1	1	0	0	0	1	
2	1	0	0	0	1	
3	1	0	0	0	1	

```
4               1      0          1             0                  1

   Metallic_Rim  Radio_cassette  Parking_Assistant  Tow_Bar
0             0               0                  0        0
1             0               0                  0        0
2             0               0                  0        0
3             0               0                  0        0
4             0               0                  0        0

[5 rows x 39 columns]
```

Now we check the features and type of each one

---

[76]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1436 entries, 0 to 1435
Data columns (total 39 columns):
Id                  1436 non-null int64
Model               1436 non-null object
Price               1436 non-null int64
Age_08_04           1436 non-null int64
Mfg_Month           1436 non-null int64
Mfg_Year            1436 non-null int64
KM                  1436 non-null int64
Fuel_Type           1436 non-null object
HP                  1436 non-null int64
Met_Color           1436 non-null int64
Color               1436 non-null object
Automatic           1436 non-null int64
CC                  1436 non-null int64
Doors               1436 non-null int64
Cylinders           1436 non-null int64
Gears               1436 non-null int64
Quarterly_Tax       1436 non-null int64
Weight              1436 non-null int64
Mfr_Guarantee       1436 non-null int64
BOVAG_Guarantee     1436 non-null int64
Guarantee_Period    1436 non-null int64
ABS                 1436 non-null int64
Airbag_1            1436 non-null int64
Airbag_2            1436 non-null int64
Airco               1436 non-null int64
Automatic_airco     1436 non-null int64
Boardcomputer       1436 non-null int64
```

```
CD_Player            1436 non-null int64
Central_Lock         1436 non-null int64
Powered_Windows      1436 non-null int64
Power_Steering       1436 non-null int64
Radio                1436 non-null int64
Mistlamps            1436 non-null int64
Sport_Model          1436 non-null int64
Backseat_Divider     1436 non-null int64
Metallic_Rim         1436 non-null int64
Radio_cassette       1436 non-null int64
Parking_Assistant    1436 non-null int64
Tow_Bar              1436 non-null int64
dtypes: int64(36), object(3)
memory usage: 437.7+ KB
```

As we can see, "Model", "Fuel_Type" and "Color" are categorical features. So we will change their types in the DataFrame

---

[77]:
```
df['Fuel_Type'] = df['Fuel_Type'].astype('category')
df['Color'] = df['Color'].astype('category')

display(df['Fuel_Type'])
print("-----------------")
display(df['Color'])
print("-----------------")

df.info()
```

```
0       Diesel
1       Diesel
2       Diesel
3       Diesel
4       Diesel
         ...
1431    Petrol
1432    Petrol
1433    Petrol
1434    Petrol
1435    Petrol
Name: Fuel_Type, Length: 1436, dtype: category
Categories (3, object): [CNG, Diesel, Petrol]


-----------------

0         Blue
```

```
1       Silver
2         Blue
3        Black
4        Black
         ...
1431      Blue
1432      Grey
1433      Blue
1434      Grey
1435     Green
Name: Color, Length: 1436, dtype: category
Categories (10, object): [Beige, Black, Blue, Green, ..., Silver, Violet, White, Yellow]


-----------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1436 entries, 0 to 1435
Data columns (total 39 columns):
Id                  1436 non-null int64
Model               1436 non-null object
Price               1436 non-null int64
Age_08_04           1436 non-null int64
Mfg_Month           1436 non-null int64
Mfg_Year            1436 non-null int64
KM                  1436 non-null int64
Fuel_Type           1436 non-null category
HP                  1436 non-null int64
Met_Color           1436 non-null int64
Color               1436 non-null category
Automatic           1436 non-null int64
CC                  1436 non-null int64
Doors               1436 non-null int64
Cylinders           1436 non-null int64
Gears               1436 non-null int64
Quarterly_Tax       1436 non-null int64
Weight              1436 non-null int64
Mfr_Guarantee       1436 non-null int64
BOVAG_Guarantee     1436 non-null int64
Guarantee_Period    1436 non-null int64
ABS                 1436 non-null int64
Airbag_1            1436 non-null int64
Airbag_2            1436 non-null int64
Airco               1436 non-null int64
Automatic_airco     1436 non-null int64
Boardcomputer       1436 non-null int64
CD_Player           1436 non-null int64
Central_Lock        1436 non-null int64
Powered_Windows     1436 non-null int64
```

```
Power_Steering       1436 non-null int64
Radio                1436 non-null int64
Mistlamps            1436 non-null int64
Sport_Model          1436 non-null int64
Backseat_Divider     1436 non-null int64
Metallic_Rim         1436 non-null int64
Radio_cassette       1436 non-null int64
Parking_Assistant    1436 non-null int64
Tow_Bar              1436 non-null int64
dtypes: category(2), int64(36), object(1)
memory usage: 418.5+ KB
```

Now we will drop "Id" and "Model" features, and perform a One Hot Encoding on "Color" and "Fuel_Type" features.

---

[78]:
```
from sklearn.preprocessing import OneHotEncoder

y = df['Price']
X = df.drop(columns=['Price', 'Id', 'Model'])

encoder = OneHotEncoder(handle_unknown='ignore')
X = pd.get_dummies(X, columns=['Color', 'Fuel_Type'], prefix=['Color_',␣
↪'Fuel_'] )
X_columns = X.columns
X.head()
```

[78]:

	Age_08_04	Mfg_Month	Mfg_Year	KM	HP	Met_Color	Automatic	CC	\
0	23	10	2002	46986	90	1	0	2000	
1	23	10	2002	72937	90	1	0	2000	
2	24	9	2002	41711	90	1	0	2000	
3	26	7	2002	48000	90	0	0	2000	
4	30	3	2002	38500	90	0	0	2000	

	Doors	Cylinders	…	Color__Green	Color__Grey	Color__Red	\
0	3	4	…	0	0	0	
1	3	4	…	0	0	0	
2	3	4	…	0	0	0	
3	3	4	…	0	0	0	
4	3	4	…	0	0	0	

	Color__Silver	Color__Violet	Color__White	Color__Yellow	Fuel__CNG	\
0	0	0	0	0	0	
1	1	0	0	0	0	
2	0	0	0	0	0	
3	0	0	0	0	0	

```
4              0              0              0              0          0

   Fuel__Diesel  Fuel__Petrol
0             1             0
1             1             0
2             1             0
3             1             0
4             1             0

[5 rows x 47 columns]
```

And in the final data manipulation step, we will normalize features and classes with a MinMax Scalar.

---

[79]:
```
from sklearn.preprocessing import MinMaxScaler

scaler_x = MinMaxScaler()
scaler_x.fit(X)
X = scaler_x.transform(X)

y = y.values.reshape(-1, 1)
scaler_y = MinMaxScaler()
scaler_y.fit_transform(y)
y = scaler_y.transform(y)

X = pd.DataFrame(X, columns=X_columns)
```

Now let's split the data into 3 parts, 70% for training, 15% for validation and 15% for test.

---

[80]:
```
from sklearn.model_selection import train_test_split

X_train, X_validation, y_train, y_validation = train_test_split(X, y,␣
 ↪test_size=0.3)
X_validation, X_test, y_validation, y_test = train_test_split(X_validation,␣
 ↪y_validation, test_size=0.5)

display(X_train)
display(y_train)
```

```
      Age_08_04  Mfg_Month  Mfg_Year        KM        HP  Met_Color  \
996    0.835443   0.090909  0.166667  0.173256  0.333333        1.0
```

```
1114   0.949367   0.363636  0.000000  0.474669  0.333333        1.0
765    0.746835   0.727273  0.166667  0.325717  0.333333        1.0
648    0.797468   0.363636  0.166667  0.477364  0.138211        0.0
455    0.658228   0.272727  0.333333  0.296293  0.227642        1.0
...         ...        ...       ...       ...       ...        ...
788    0.810127   0.272727  0.166667  0.308639  0.138211        0.0
1286   0.987342   0.090909  0.000000  0.297643  0.333333        1.0
965    0.835443   0.090909  0.166667  0.204050  0.333333        0.0
264    0.481013   0.454545  0.500000  0.205758  0.227642        1.0
1226   1.000000   0.000000  0.000000  0.345676  0.333333        0.0

      Automatic        CC     Doors  Cylinders  ...  Color__Green  \
996         0.0  0.020408  1.000000        0.0  ...           0.0
1114        0.0  0.020408  1.000000        0.0  ...           0.0
765         0.0  0.020408  1.000000        0.0  ...           1.0
648         0.0  0.000000  1.000000        0.0  ...           0.0
455         0.0  0.006803  0.333333        0.0  ...           1.0
...         ...       ...       ...        ...  ...           ...
788         0.0  0.000000  0.333333        0.0  ...           0.0
1286        0.0  0.020408  1.000000        0.0  ...           0.0
965         0.0  0.020408  1.000000        0.0  ...           0.0
264         0.0  0.006803  1.000000        0.0  ...           0.0
1226        0.0  0.020408  1.000000        0.0  ...           0.0

      Color__Grey  Color__Red  Color__Silver  Color__Violet  Color__White  \
996           0.0         0.0            1.0            0.0           0.0
1114          0.0         1.0            0.0            0.0           0.0
765           0.0         0.0            0.0            0.0           0.0
648           0.0         0.0            0.0            0.0           0.0
455           0.0         0.0            0.0            0.0           0.0
...           ...         ...            ...            ...           ...
788           0.0         0.0            0.0            0.0           0.0
1286          0.0         0.0            1.0            0.0           0.0
965           0.0         1.0            0.0            0.0           0.0
264           0.0         0.0            0.0            0.0           0.0
1226          0.0         0.0            0.0            0.0           0.0

      Color__Yellow  Fuel__CNG  Fuel__Diesel  Fuel__Petrol
996             0.0        0.0           0.0           1.0
1114            0.0        0.0           0.0           1.0
765             0.0        0.0           0.0           1.0
648             0.0        0.0           0.0           1.0
455             0.0        0.0           0.0           1.0
...             ...        ...           ...           ...
788             0.0        0.0           0.0           1.0
1286            0.0        0.0           0.0           1.0
965             0.0        0.0           0.0           1.0
264             0.0        0.0           0.0           1.0
```

```
1226            0.0         0.0            0.0            1.0

[1005 rows x 47 columns]


array([[0.19893428],
       [0.09058615],
       [0.21847247],
       ...,
       [0.19715808],
       [0.27175844],
       [0.09236234]])
```

Here we will generate our linear regression model with the help of sci-kit learn. And we will predict the test data with the model.

---

[81]:
```
from sklearn.linear_model import LinearRegression

regression = LinearRegression().fit(X_train, y_train)
predicted = regression.predict(X_test)
```

Finally, we will calculate the performance metrics for the model on the data.

---

[82]:
```
mean_squared_error = np.mean(np.power((y_test - predicted), 2))
root_mean_squared_error = np.sqrt(mean_squared_error)
mean_absolute_error = np.mean(np.abs(y_test - predicted))

print("MSE (Mean Squared Error) = ", mean_squared_error)
print("RMSE (Root Mean Squared Error) = ",root_mean_squared_error)
print("MAE (Mean Absolute Error) = ", mean_absolute_error)
```

```
MSE (Mean Squared Error) =  0.0023755472211407884
RMSE (Root Mean Squared Error) =  0.04873958577112436
MAE (Mean Absolute Error) =  0.03263831963296643
```

1.1 Accuracy (Toyota Dataset)

1.1.1 MSE (Mean Squared Error) = 0.0015140205183283933

1.1.2 RMSE (Root Mean Squared Error) = 0.03891041657870541

1.1.3 MAE (Mean Absolute Error) = 0.02941729708271868

---

2 Part 2

We do all the steps again (Except one hot encoding, because there is no categorical data in this dataset)

[83]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

df = pd.read_csv('Regression.csv')
df.head()

y = df['Chance of Admit ']
X = df.drop(columns=['Chance of Admit ', 'Serial No.'])

scaler_x = MinMaxScaler()
scaler_x.fit(X)
X = scaler_x.transform(X)

y = y.values.reshape(-1, 1)
scaler_y = MinMaxScaler()
scaler_y.fit_transform(y)
y = scaler_y.transform(y)

X_train, X_validation, y_train, y_validation = train_test_split(X, y,␣
 ↪test_size=0.3)
X_validation, X_test, y_validation, y_test = train_test_split(X_validation,␣
 ↪y_validation, test_size=0.5)

regression = LinearRegression().fit(X_train, y_train)
predicted = regression.predict(X_test)

mean_squared_error = np.mean(np.power((y_test - predicted), 2))
root_mean_squared_error = np.sqrt(mean_squared_error)
mean_absolute_error = np.mean(np.abs(y_test - predicted))

print("MSE (Mean Squared Error) = ", mean_squared_error)
print("RMSE (Root Mean Squared Error) = ",root_mean_squared_error)
```

```
print("MAE (Mean Absolute Error) = ", mean_absolute_error)
```

```
MSE (Mean Squared Error) =  0.00904226378228622
RMSE (Root Mean Squared Error) =  0.09509081860140978
MAE (Mean Absolute Error) =  0.06550324624443074
```

2.1 Accuracy (Student Chance Dataset)

2.1.1 MSE (Mean Squared Error) = 0.008418063292323244

2.1.2 RMSE (Root Mean Squared Error) = 0.09175000431783774

2.1.3 MAE (Mean Absolute Error) = 0.06895701897384861
