Lab report 10

Fall 2021

CSE422L Data Analytics Lab

Submitted by: **Ayaz Mehmood**

Registration No.:**18PWCSE1652**

Section: **A**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: _______________

Submitted to:

Engr. Mian Ibad Ali Shah

Last date of Submission:

24 February 2022

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

TASKS

Try and understand the provided code and dataset. Increase the accuracy using same dataset and algorithm. Make changes in this code. Make a

proper lab report with screenshots and details about how have you achieved the increase in accuracy.

Import Relevant Libraries:

Importing the relevant libraries

In [1]:

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import seaborn as sns
sns.set()
```

Reading a csv file

Loading the raw data

In [2]:

```
raw_data = pd.read_csv('CarSelling Portal Data.csv')
raw_data.head()
```

Out[2]:

	Brand	Price	Body	Mileage	EngineV	Engine Type	Registration	Year	Model
0	BMW	4200.0	sedan	277	2.0	Petrol	yes	1991	320
1	Mercedes-Benz	7900.0	van	427	2.9	Diesel	yes	1999	Sprinter 212
2	Mercedes-Benz	13300.0	sedan	358	5.0	Gas	yes	2003	S 500
3	Audi	23000.0	crossover	240	4.2	Petrol	yes	2007	Q7
4	Toyota	18300.0	crossover	120	2.0	Petrol	yes	2011	Rav 4

descriptive statistics of the variables

In [3]:

```
raw_data.describe(include='all')
```

Out[3]:

	Brand	Price	Body	Mileage	EngineV	Engine Type	Registration	Year	Model
count	4345	4173.000000	4345	4345.000000	4195.000000	4345	4345	4345.000000	4345
unique	7	NaN	6	NaN	NaN	4	2	NaN	312
top	Volkswagen	NaN	sedan	NaN	NaN	Diesel	yes	NaN	E-Class
freq	936	NaN	1649	NaN	NaN	2019	3947	NaN	199
mean	NaN	19418.746935	NaN	161.237284	2.790734	NaN	NaN	2006.550058	NaN
std	NaN	25584.242620	NaN	105.705797	5.066437	NaN	NaN	6.719097	NaN
min	NaN	600.000000	NaN	0.000000	0.600000	NaN	NaN	1969.000000	NaN
25%	NaN	6999.000000	NaN	86.000000	1.800000	NaN	NaN	2003.000000	NaN
50%	NaN	11500.000000	NaN	155.000000	2.200000	NaN	NaN	2008.000000	NaN
75%	NaN	21700.000000	NaN	230.000000	3.000000	NaN	NaN	2012.000000	NaN
max	NaN	300000.000000	NaN	980.000000	99.990000	NaN	NaN	2016.000000	NaN

Dropping non important variables:

Determining the variables of interest

In [4]:
```
data = raw_data.drop(['Model'],axis=1)
data.describe(include='all')
```

Checking Null values of variables:

Dealing with missing values

In [5]:
```
data.isnull().sum()
```

Out[5]:
```
Brand             0
Price           172
Body              0
Mileage           0
EngineV         150
Engine Type       0
Registration      0
Year              0
dtype: int64
```

Dropping Null values:

In [9]:
```
data_no_mv = data.dropna(axis=0)
```

In [10]:
```
data_no_mv.isnull().sum()
```

Out[10]:
```
Brand           0
Price           0
Body            0
Mileage         0
EngineV         0
Engine Type     0
Registration    0
Year            0
dtype: int64
```

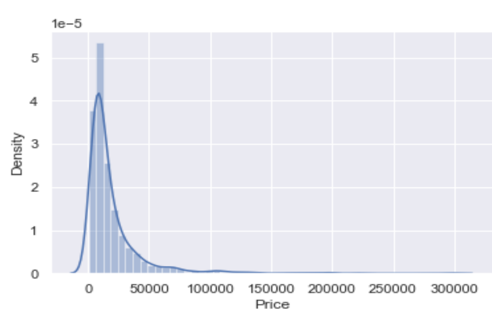
Checking outlier using PDF:

Exploring the PDFs

In [12]:
```
sns.distplot(data_no_mv['Price'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a d
d will be removed in a future version. Please adapt your code to use either `displot` (a figure-level func
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[12]: <AxesSubplot:xlabel='Price', ylabel='Density'>



Dropping outlier:

Dealing with outliers

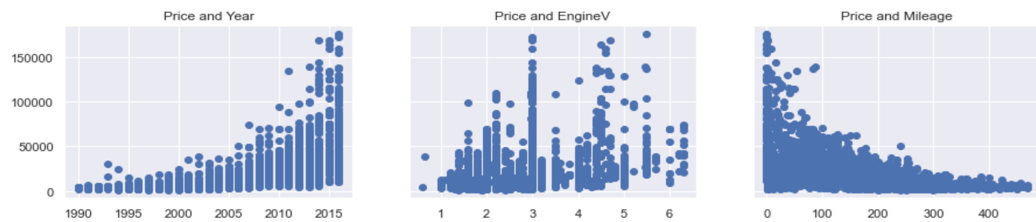
In [13]:
```
q = data_no_mv['Price'].quantile(0.995)
data_1 = data_no_mv[data_no_mv['Price']<q]
data_1.describe(include='all')
```

Transforming the data to suit for LR:

Transforming the data to suit Linear Regression

In [26]:
```
f, (ax1, ax2, ax3) = plt.subplots(1, 3, sharey=True, figsize =(15,3))
ax1.scatter(data_cleaned['Year'],data_cleaned['Price'])
ax1.set_title('Price and Year')
ax2.scatter(data_cleaned['EngineV'],data_cleaned['Price'])
ax2.set_title('Price and EngineV')
ax3.scatter(data_cleaned['Mileage'],data_cleaned['Price'])
ax3.set_title('Price and Mileage')


plt.show()
```



Taking log of price variable:

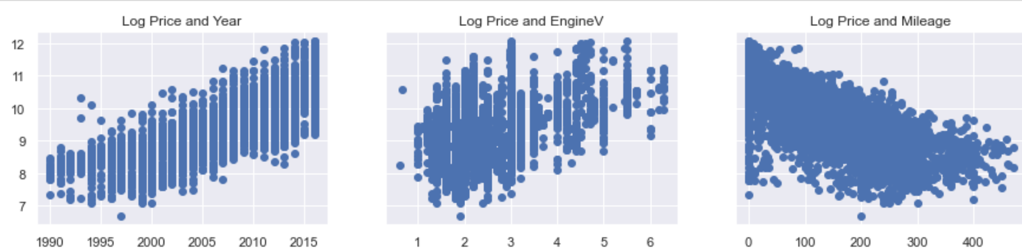
Relaxing the assumptions

In [27]:
```
log_price = np.log(data_cleaned['Price'])
data_cleaned['log_price'] = log_price
data_cleaned
```

Checking the data to suit for LR:

In [28]:
```
f, (ax1, ax2, ax3) = plt.subplots(1, 3, sharey=True, figsize =(15,3))
ax1.scatter(data_cleaned['Year'],data_cleaned['log_price'])
ax1.set_title('Log Price and Year')
ax2.scatter(data_cleaned['EngineV'],data_cleaned['log_price'])
ax2.set_title('Log Price and EngineV')
ax3.scatter(data_cleaned['Mileage'],data_cleaned['log_price'])
ax3.set_title('Log Price and Mileage')


plt.show()
```



Checking Multi collinearity and dropping variable having higher VIF value:

In [31]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
variables = data_cleaned[['Mileage','Year','EngineV']]
vif = pd.DataFrame()
vif["VIF"] = [variance_inflation_factor(variables.values, i)
for i in range(variables.shape[1])]
vif["features"] = variables.columns
```

In [32]:
```
vif
```

Out[32]:

	VIF	features
0	3.721769	Mileage
1	10.319178	Year
2	7.595456	EngineV

In [33]:
```
data_no_multicollinearity = data_cleaned.drop(['Year'],axis=1)
```

Creating dummy variables for multiclass values:

Create dummy variables

In [34]:
```
data_with_dummies = pd.get_dummies(data_no_multicollinearity, drop_first=True)
```

In [35]:
```
data_with_dummies.head()
```

Out[35]:

Declaring input features and output label:

Declare the inputs and the targets

In [39]:
```
targets = data_preprocessed['log_price']
inputs = data_preprocessed.drop(['log_price'],axis=1)
```

Scaling the data using standard scaler:

Scale the data

In [40]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(inputs)
```

Out[40]: StandardScaler()

In [41]:
```
inputs_scaled = scaler.transform(inputs)
```

Train Test Split:

Train Test Split

In [42]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(inputs_scaled, targets, test_size=0.2, random_state=365)
```

Linear Regression Model and training the Model:

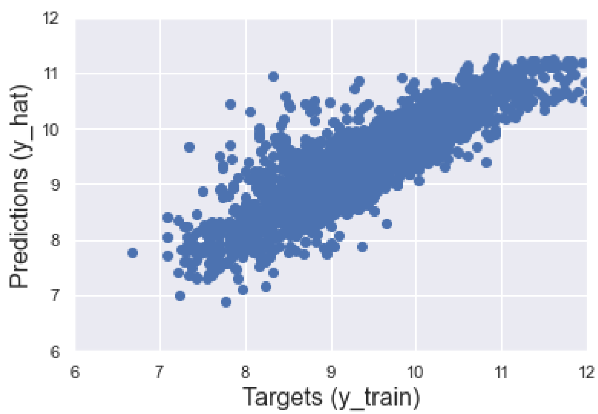
In [43]:
```
reg = LinearRegression()
reg.fit(x_train,y_train)
```

Out[43]: LinearRegression()

Checking the model (actual Result and Predicted result):

In [44]:
```
y_hat = reg.predict(x_train)
```

In [45]:
```
plt.scatter(y_train, y_hat)
plt.xlabel('Targets (y_train)',size=16)
plt.ylabel('Predictions (y_hat)',size=16)
plt.xlim(6,12)
plt.ylim(6,12)
plt.show()
```

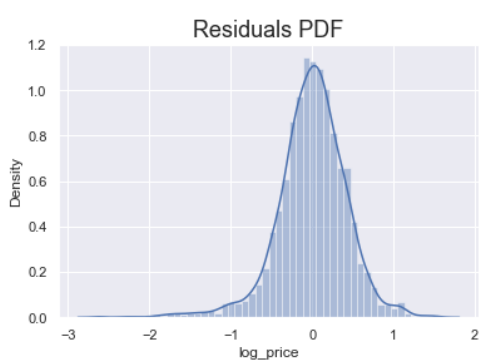


Residuals PDF:

In [46]:
```
sns.distplot(y_train - y_hat)
plt.title("Residuals PDF", size=18)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplo
d will be removed in a future version. Please adapt your code to use either `displot` (a figure-l
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[46]: Text(0.5, 1.0, 'Residuals PDF')



Checking accuracy of the model:

```
In [47]: reg.score(x_train,y_train)
Out[47]: 0.76295052810071
```

Finding weights and bias:

Finding the weights and bias

```
In [48]: reg.intercept_
Out[48]: 9.43959623405551
```

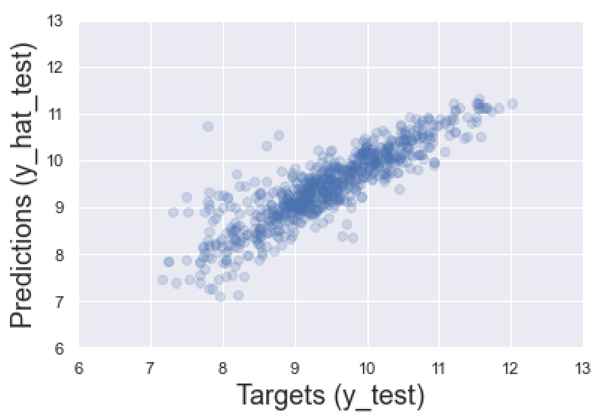
```
In [49]: reg.coef_
Out[49]: array([-0.46727728,  0.21548062,  0.01863881,  0.01850695, -0.1377817 ,
                -0.1765456 , -0.06408459, -0.08497277, -0.15030964, -0.09492324,
                -0.18905509, -0.11897914, -0.16140196, -0.11619223, -0.03048399,
                -0.13982773,  0.31916797])
```

Testing the model on test dataset:

Testing

```
In [52]: y_hat_test = reg.predict(x_test)
```

```
In [53]: plt.scatter(y_test, y_hat_test, alpha=0.2)
         plt.xlabel('Targets (y_test)',size=18)
         plt.ylabel('Predictions (y_hat_test)',size=18)
         plt.xlim(6,13)
         plt.ylim(6,13)
         plt.show()
```



What I did to increase the accuracy:

In [13]:
```
q = data_no_mv['Price'].quantile(0.995)
data_1 = data_no_mv[data_no_mv['Price']<q]
data_1.describe(include='all')
```

Out[13]:

Instead of dropping 1% quartile I dropped 0.5% quartile in the price variable.

In [16]:
```
q = data_1['Mileage'].quantile(0.995)
data_2 = data_1[data_1['Mileage']<q]
data_2.describe(include='all')
```

Did the same in Mileage variable and got an increase accuracy of 2.1%
Previously the model accuracy was 74% and now it is 76.2%.

I tried some other method to increase the accuracy but that did not work according to my requirement. Instead of dropping the null variables I tried to fill it using the mean of that variables but that did not increase my accuracy.