(Shared)MLP_Week_4_SWI(01_10_22)

October 28, 2022

You are working as a data scientist in a big automobile company. Your company aspires to enter the US market by setting up their manufacturing unit there and producing cars locally to give competition to their US and European counterparts.

They got some data to understand the factors on which the pricing of cars depends in the American market, since those may vary different from the indian market. The company wants to know:

Which variables are significant in predicting the price of a car, How well those variables describe the price of a car Based on various market surveys.

1 Business Goal:

Data science team are required to model the price of cars with the available independent variables. It will be used by the management to understand how exactly the prices vary with the independent variables. They can accordingly manipulate the design of the cars, the business strategy etc. to meet certain price levels. Further, the model will be a good way for management to understand the pricing dynamics of a new market.

2 Step-1: Importing Libraries

[1]:
```
# Importing the libraries
import numpy as np
import pandas as pd
from numpy import math

from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score


import seaborn as sns
import matplotlib.pyplot as plt
import warnings
```

```
warnings.filterwarnings('ignore')
```

2.0.1 DataSet Information:

Dataset_link: https://drive.google.com/file/d/1EXZXCVl-be9M2zB7H1NkfDkSUSt8poCc/view?usp=sharing

Features:-

Car_ID: Unique id of each observation (Interger)

Symboling: Its assigned insurance risk rating, A value of +3 indicates that the auto is risky, -3 that it is probably pretty safe.(Categorical)

CarName: Name of car company (Categorical)

fueltype: Car fuel type i.e gas or diesel (Categorical)

aspiration: Aspiration used in a car (Categorical)

doornumber: Number of doors in a car (Categorical)

carbody: body of car (Categorical)

drivewheel: type of drive wheel (Categorical)

enginelocation: Location of car engine (Categorical)

wheelbase: Weelbase of car (Numeric)

carlength: Length of car (Numeric)

carwidth: Width of car (Numeric)

carheight: height of car (Numeric)

curbweight: The weight of a car without occupants or baggage. (Numeric)

enginetype: Type of engine. (Categorical)

cylindernumber: cylinder placed in the car (Categorical)

enginesize: Size of car (Numeric)

fuelsystem: Fuel system of car (Categorical)

boreratio: Boreratio of car (Numeric)

stroke: Stroke or volume inside the engine (Numeric)

compressionratio: compression ratio of car (Numeric)

horsepower: Horsepower (Numeric)

peakrpm: car peak rpm (Numeric)

citympg: Mileage in city (Numeric)

highwaympg: Mileage on highway (Numeric)

price(Dependent variable): Price of car (Numeric)

[ ]:
```

```

3 Step-2: Loading the data

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

[ ]:
```
#pd.read_csv("/content/drive/MyDrive/Dataset/Car_price in US market.csv")
```

[ ]:
```
from google.colab import files
files.upload()
```

[5]:
```
# Importing the dataset
dataset = pd.read_csv('Car_price in US market.csv')
```

4 Step-3: Data Inspection

5 Question set-1:

(i) No of data point

(ii) No of features

(iii) No of categorical features

(iv) No of numerical features

(v) No of NA values

(vi) List of all features

(vii) What about duplicate data ?

[6]:
```
dataset.shape[1]
```

[6]:
```
26
```

[7]:
```
dataset.head(5)
```

[7]:
```
   car_ID  symboling                   CarName fueltype aspiration doornumber  \
0       1          3        alfa-romero giulia      gas        std        two
1       2          3       alfa-romero stelvio      gas        std        two
2       3          1  alfa-romero Quadrifoglio      gas        std        two
3       4          2               audi 100 ls      gas        std       four
4       5          2                audi 100ls      gas        std       four
```

```
       carbody drivewheel enginelocation  wheelbase  …  enginesize  \
0  convertible        rwd          front       88.6  …         130
1  convertible        rwd          front       88.6  …         130
2    hatchback        rwd          front       94.5  …         152
3        sedan        fwd          front       99.8  …         109
4        sedan        4wd          front       99.4  …         136

   fuelsystem  boreratio  stroke compressionratio horsepower  peakrpm citympg  \
0        mpfi       3.47    2.68              9.0        111     5000      21
1        mpfi       3.47    2.68              9.0        111     5000      21
2        mpfi       2.68    3.47              9.0        154     5000      19
3        mpfi       3.19    3.40             10.0        102     5500      24
4        mpfi       3.19    3.40              8.0        115     5500      18

   highwaympg    price
0          27  13495.0
1          27  16500.0
2          26  16500.0
3          30  13950.0
4          22  17450.0

[5 rows x 26 columns]
```

[8]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
 4   aspiration        205 non-null    object
 5   doornumber        205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheel        205 non-null    object
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    object
 15  cylindernumber    205 non-null    object
 16  enginesize        205 non-null    int64
```

```
 17  fuelsystem        205 non-null    object
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64
 22  peakrpm           205 non-null    int64
 23  citympg           205 non-null    int64
 24  highwaympg        205 non-null    int64
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

[9]:
```
dataset.isnull().sum()
```

[9]:
```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

[10]:
```
dataset.describe()
```

[10]:
```
           car_ID   symboling   wheelbase   carlength    carwidth   carheight  \
count  205.000000  205.000000  205.000000  205.000000  205.000000  205.000000
mean   103.000000    0.834146   98.756585  174.049268   65.907805   53.724878
```

```
std     59.322565    1.245307    6.021776   12.337289    2.145204    2.443522
min      1.000000   -2.000000   86.600000  141.100000   60.300000   47.800000
25%     52.000000    0.000000   94.500000  166.300000   64.100000   52.000000
50%    103.000000    1.000000   97.000000  173.200000   65.500000   54.100000
75%    154.000000    2.000000  102.400000  183.100000   66.900000   55.500000
max    205.000000    3.000000  120.900000  208.100000   72.300000   59.800000

        curbweight  enginesize   boreratio      stroke  compressionratio  \
count   205.000000  205.000000  205.000000  205.000000        205.000000
mean   2555.565854  126.907317    3.329756    3.255415         10.142537
std     520.680204   41.642693    0.270844    0.313597          3.972040
min    1488.000000   61.000000    2.540000    2.070000          7.000000
25%    2145.000000   97.000000    3.150000    3.110000          8.600000
50%    2414.000000  120.000000    3.310000    3.290000          9.000000
75%    2935.000000  141.000000    3.580000    3.410000          9.400000
max    4066.000000  326.000000    3.940000    4.170000         23.000000

       horsepower      peakrpm     citympg  highwaympg         price
count  205.000000   205.000000  205.000000  205.000000    205.000000
mean   104.117073  5125.121951   25.219512   30.751220  13276.710571
std     39.544167   476.985643    6.542142    6.886443   7988.852332
min     48.000000  4150.000000   13.000000   16.000000   5118.000000
25%     70.000000  4800.000000   19.000000   25.000000   7788.000000
50%     95.000000  5200.000000   24.000000   30.000000  10295.000000
75%    116.000000  5500.000000   30.000000   34.000000  16503.000000
max    288.000000  6600.000000   49.000000   54.000000  45400.000000
```

[11]:
```
features=(dataset.columns)
```

[12]:
```
features
```

[12]:
```
Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price'],
      dtype='object')
```

[13]:
```
len(dataset[dataset.duplicated()])
```

[13]: 0

6 Step-5: Exploratory data analysis

Question Set 2:

1) Give list of all numeric features
2) List of all categorical features
3) Type of distribution your dependent variable follow.
4) Plot different graph (histogram,box-plot,scatter plot e.t.c) for all independent variable to get some insight.
5) some scaling is needed or not.

(5) Comment about different categorical feature.

(6) Create a function which converts string into numerical e.g- {"four": 4, "two": 2}

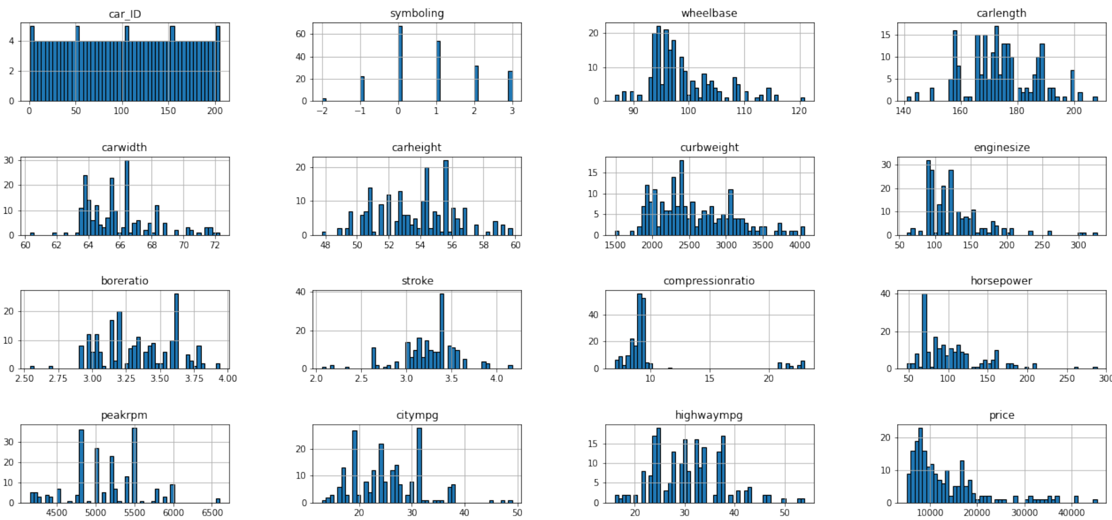
[14]:
```
num_feat=dataset.describe().columns
list(num_feat)
```

[14]:
```
['car_ID',
 'symboling',
 'wheelbase',
 'carlength',
 'carwidth',
 'carheight',
 'curbweight',
 'enginesize',
 'boreratio',
 'stroke',
 'compressionratio',
 'horsepower',
 'peakrpm',
 'citympg',
 'highwaympg',
 'price']
```

[15]:
```
all_feat=list(dataset.columns)
cat_feat= [i for i in all_feat  if i not in num_feat]
cat_feat
```

[15]:
```
['CarName',
 'fueltype',
 'aspiration',
 'doornumber',
 'carbody',
 'drivewheel',
 'enginelocation',
 'enginetype',
 'cylindernumber',
 'fuelsystem']
```

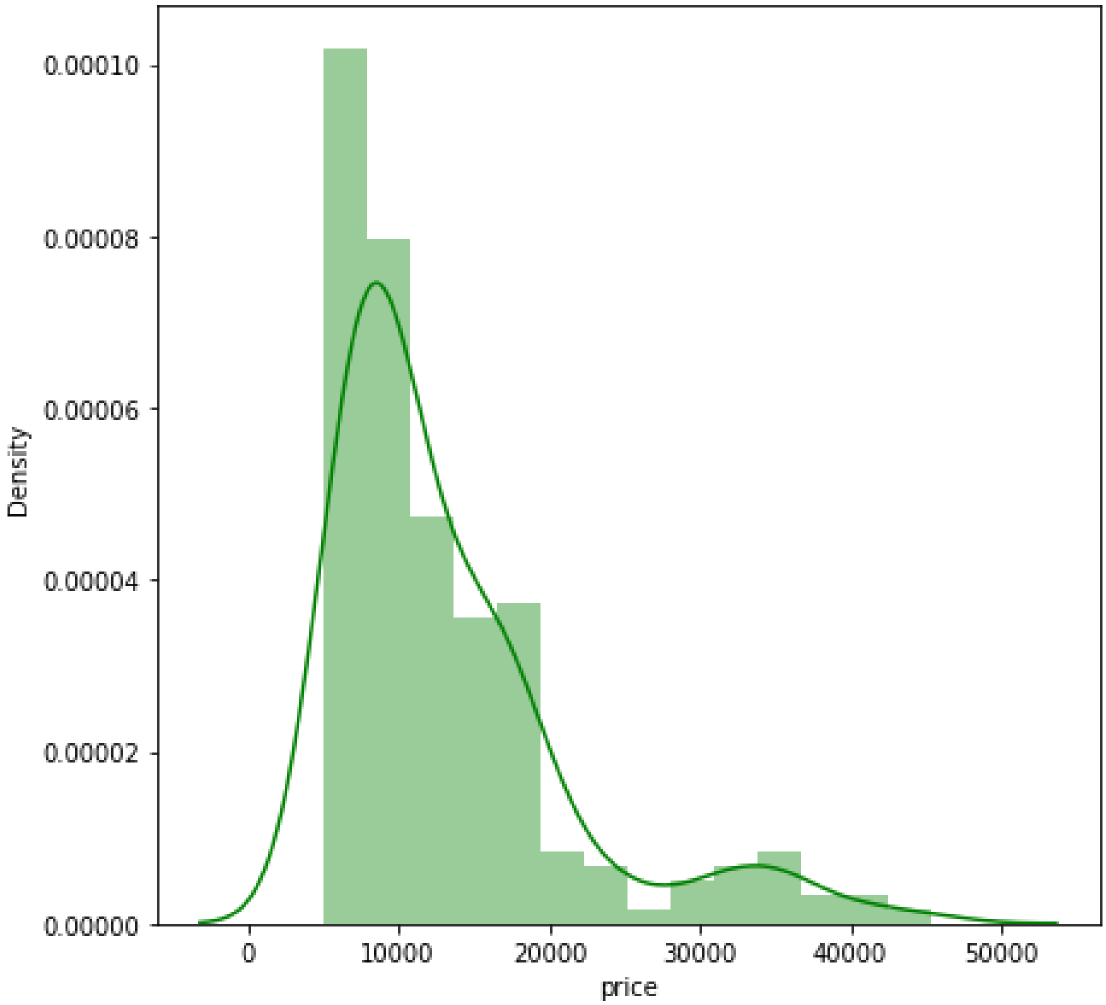
```
[ ]: dataset.hist(figsize=(22, 10), bins=50, edgecolor="black")
     plt.subplots_adjust(hspace=0.7, wspace=0.4)
```



7 Distribution of independent variable

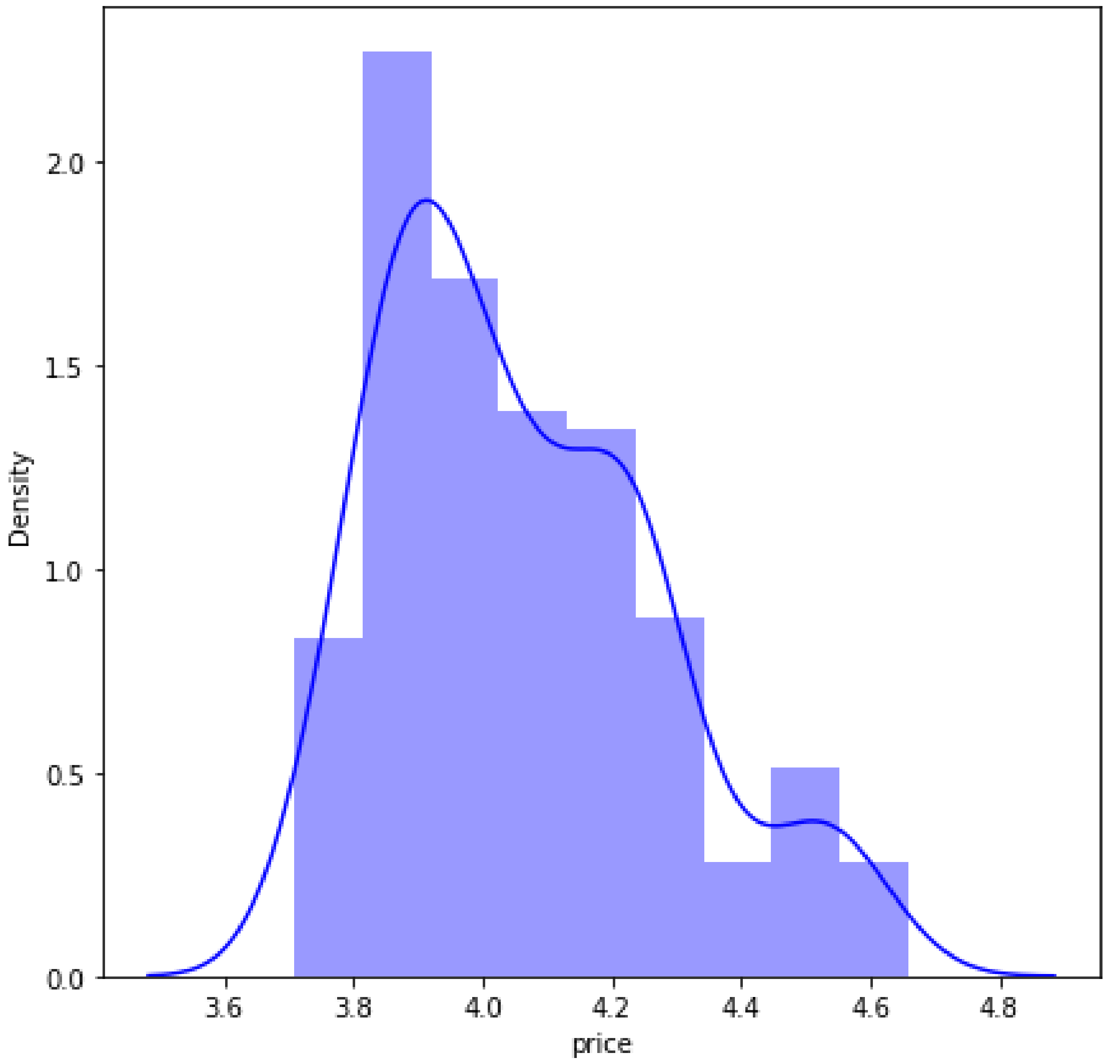
```
[ ]: plt.figure(figsize=(7,7))
     sns.distplot(dataset['price'],color="g")
```

```
[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f0cc17c6450>
```



[ ]:
```
plt.figure(figsize=(7,7))
sns.distplot(np.log10(dataset['price']),color="b")
```

[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f0cc0712ad0>



8 Correlation between different features

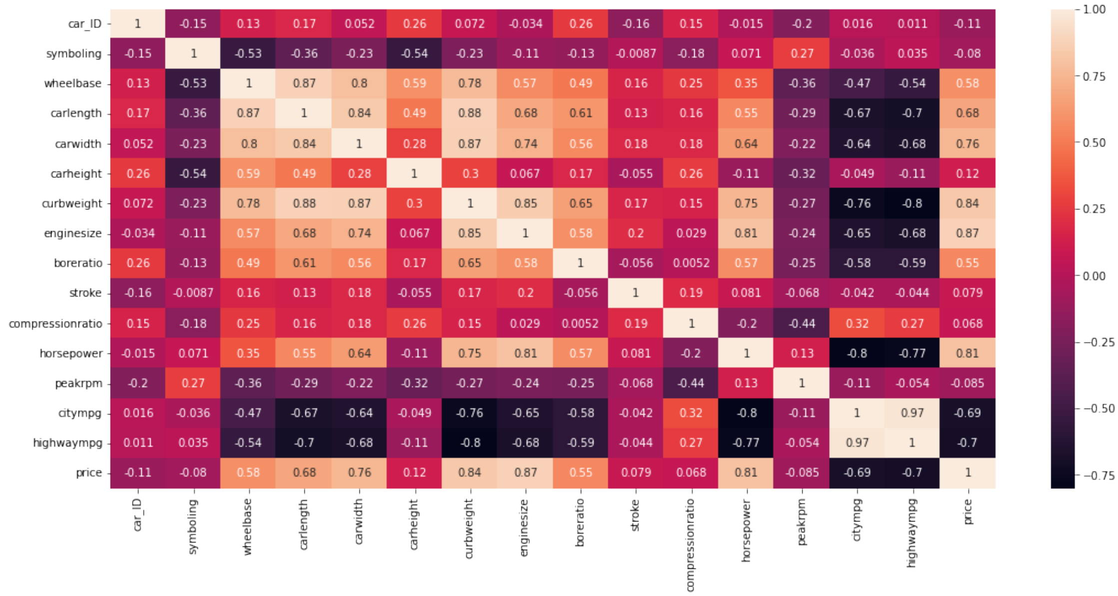
Question Set 3:

1) Create a dataframe named vif_df which has 2 columns named variables and their vif.
2) Create a function named make_vif_df which returns the above dataframe.
3) Start removing column (whose vif >5 or maybe 10), one at a time and check.
4) Apply one hot encoding on features like (”carbody”, ”enginetype”,”fuelsystem”)

[16]:
```
plt.figure(figsize=(18,8))
corre = dataset.corr()
```

```python
sns.heatmap(corre,annot=True)
```

[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe5581eff90>



9 Feature engineering

- Check How many unique car brands are working in the USA?
- What is avg price of Jaguar car.

[ ]:
```python
dataset['CarName'].unique()
```

[ ]:
```
array(['alfa-romero giulia', 'alfa-romero stelvio',
       'alfa-romero Quadrifoglio', 'audi 100 ls', 'audi 100ls',
       'audi fox', 'audi 5000', 'audi 4000', 'audi 5000s (diesel)',
       'bmw 320i', 'bmw x1', 'bmw x3', 'bmw z4', 'bmw x4', 'bmw x5',
       'chevrolet impala', 'chevrolet monte carlo', 'chevrolet vega 2300',
       'dodge rampage', 'dodge challenger se', 'dodge d200',
       'dodge monaco (sw)', 'dodge colt hardtop', 'dodge colt (sw)',
       'dodge coronet custom', 'dodge dart custom',
       'dodge coronet custom (sw)', 'honda civic', 'honda civic cvcc',
       'honda accord cvcc', 'honda accord lx', 'honda civic 1500 gl',
       'honda accord', 'honda civic 1300', 'honda prelude',
       'honda civic (auto)', 'isuzu MU-X', 'isuzu D-Max ',
       'isuzu D-Max V-Cross', 'jaguar xj', 'jaguar xf', 'jaguar xk',
       'maxda rx3', 'maxda glc deluxe', 'mazda rx2 coupe', 'mazda rx-4',
```

```
       'mazda glc deluxe', 'mazda 626', 'mazda glc', 'mazda rx-7 gs',
       'mazda glc 4', 'mazda glc custom l', 'mazda glc custom',
       'buick electra 225 custom', 'buick century luxus (sw)',
       'buick century', 'buick skyhawk', 'buick opel isuzu deluxe',
       'buick skylark', 'buick century special',
       'buick regal sport coupe (turbo)', 'mercury cougar',
       'mitsubishi mirage', 'mitsubishi lancer', 'mitsubishi outlander',
       'mitsubishi g4', 'mitsubishi mirage g4', 'mitsubishi montero',
       'mitsubishi pajero', 'Nissan versa', 'nissan gt-r', 'nissan rogue',
       'nissan latio', 'nissan titan', 'nissan leaf', 'nissan juke',
       'nissan note', 'nissan clipper', 'nissan nv200', 'nissan dayz',
       'nissan fuga', 'nissan otti', 'nissan teana', 'nissan kicks',
       'peugeot 504', 'peugeot 304', 'peugeot 504 (sw)', 'peugeot 604sl',
       'peugeot 505s turbo diesel', 'plymouth fury iii',
       'plymouth cricket', 'plymouth satellite custom (sw)',
       'plymouth fury gran sedan', 'plymouth valiant', 'plymouth duster',
       'porsche macan', 'porcshce panamera', 'porsche cayenne',
       'porsche boxter', 'renault 12tl', 'renault 5 gtl', 'saab 99e',
       'saab 99le', 'saab 99gle', 'subaru', 'subaru dl', 'subaru brz',
       'subaru baja', 'subaru r1', 'subaru r2', 'subaru trezia',
       'subaru tribeca', 'toyota corona mark ii', 'toyota corona',
       'toyota corolla 1200', 'toyota corona hardtop',
       'toyota corolla 1600 (sw)', 'toyota carina', 'toyota mark ii',
       'toyota corolla', 'toyota corolla liftback',
       'toyota celica gt liftback', 'toyota corolla tercel',
       'toyota corona liftback', 'toyota starlet', 'toyota tercel',
       'toyota cressida', 'toyota celica gt', 'toyouta tercel',
       'vokswagen rabbit', 'volkswagen 1131 deluxe sedan',
       'volkswagen model 111', 'volkswagen type 3', 'volkswagen 411 (sw)',
       'volkswagen super beetle', 'volkswagen dasher', 'vw dasher',
       'vw rabbit', 'volkswagen rabbit', 'volkswagen rabbit custom',
       'volvo 145e (sw)', 'volvo 144ea', 'volvo 244dl', 'volvo 245',
       'volvo 264gl', 'volvo diesel', 'volvo 246'], dtype=object)
```

[18]:

```
df2=dataset.copy()
```

[19]:

```
df2['company'] = df2['CarName'].str.split(" ", expand=True)[0]
df2['company'] = df2['company'].replace({'toyouta': 'Toyota','vw':
 ↪'Volkswagen','vokswagen':'Volkswagen',
                                                             'maxda':
 ↪'Mazda','porcshce':'Porsche'})
df2['company'] = df2['company'].str.title()
df2['company'].value_counts()
```

[19]:

```
Toyota          32
Nissan          18
Mazda           17
```

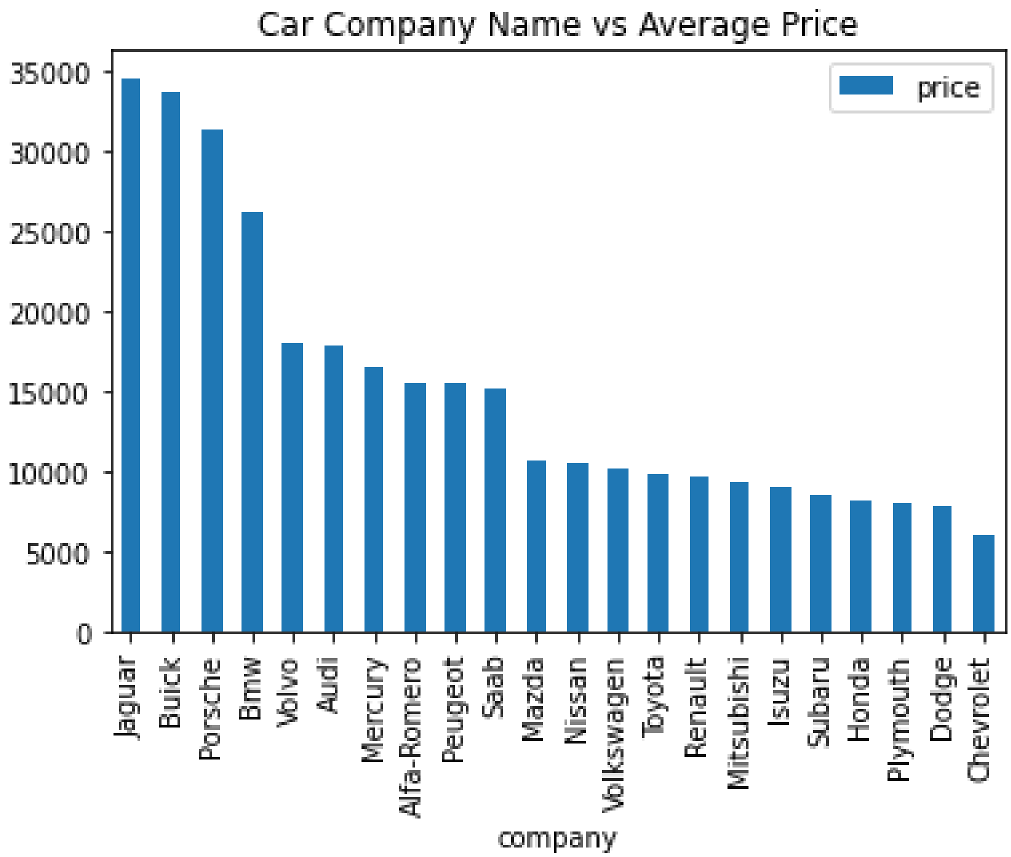
```
Mitsubishi     13
Honda          13
Volkswagen     12
Subaru         12
Peugeot        11
Volvo          11
Dodge           9
Buick           8
Bmw             8
Audi            7
Plymouth        7
Saab            6
Porsche         5
Isuzu           4
Jaguar          3
Chevrolet       3
Alfa-Romero     3
Renault         2
Mercury         1
Name: company, dtype: int64
```

[20]:

```
plt.figure(figsize=(20, 6))

df_autox = pd.DataFrame(df2.groupby(['company'])['price'].mean().
 ↪sort_values(ascending = False))
df_autox.plot.bar()
plt.title('Car Company Name vs Average Price')
plt.show()
```

```
<Figure size 1440x432 with 0 Axes>
```



[21]:
```
df_autox.rename(columns={'price':'mean_price'},inplace=True)
```

[22]:
```
df_autox.head(10)
```

[22]:

	mean_price
company	
Jaguar	34600.000000
Buick	33647.000000
Porsche	31400.500000
Bmw	26118.750000
Volvo	18063.181818
Audi	17859.166714
Mercury	16503.000000
Alfa-Romero	15498.333333
Peugeot	15489.090909
Saab	15223.333333

[23]:
```
dataset["enginelocation"].value_counts()
```

[23]:
```
front    202
rear       3
Name: enginelocation, dtype: int64
```

[24]:
```
dataset.fuelsystem.value_counts()
```

[24]:
```
mpfi    94
2bbl    66
idi     20
1bbl    11
spdi     9
4bbl     3
mfi      1
spfi     1
Name: fuelsystem, dtype: int64
```

10 Encoding categorical variable

[52]:
```
dataset_new = dataset.copy()
```

[53]:
```
encoders_nums = {"fueltype":{"diesel":1,"gas":0},
                 "aspiration":{"turbo":1,"std":0},
                 "doornumber":     {"four": 4, "two": 2},
                 "drivewheel":{"fwd":0,"4wd":0,"rwd":1},
                 "cylindernumber":{"four": 4, "six": 6, "five": 5, "eight": 8,
                                  "two": 2, "twelve": 12, "three":3 },
                 "enginelocation": {"front":0,"back":1}
                 }

dataset_new = dataset_new.replace(encoders_nums)
```

[54]:
```
dataset_new = pd.get_dummies(dataset_new, columns=["carbody",␣
↪"enginetype","fuelsystem","enginelocation"], prefix=["body",␣
↪"etype","fsystem","e_location"])
```

[65]:
```
dataset_new =dataset_new.drop(["CarName"],axis=1)
```

[66]:
```
dataset_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 43 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
```

```
 1   symboling         205 non-null    int64
 2   fueltype          205 non-null    int64
 3   aspiration        205 non-null    int64
 4   doornumber        205 non-null    int64
 5   drivewheel        205 non-null    int64
 6   wheelbase         205 non-null    float64
 7   carlength         205 non-null    float64
 8   carwidth          205 non-null    float64
 9   carheight         205 non-null    float64
 10  curbweight        205 non-null    int64
 11  cylindernumber    205 non-null    int64
 12  enginesize        205 non-null    int64
 13  boreratio         205 non-null    float64
 14  stroke            205 non-null    float64
 15  compressionratio  205 non-null    float64
 16  horsepower        205 non-null    int64
 17  peakrpm           205 non-null    int64
 18  citympg           205 non-null    int64
 19  highwaympg        205 non-null    int64
 20  price             205 non-null    float64
 21  body_convertible  205 non-null    uint8
 22  body_hardtop      205 non-null    uint8
 23  body_hatchback    205 non-null    uint8
 24  body_sedan        205 non-null    uint8
 25  body_wagon        205 non-null    uint8
 26  etype_dohc        205 non-null    uint8
 27  etype_dohcv       205 non-null    uint8
 28  etype_l           205 non-null    uint8
 29  etype_ohc         205 non-null    uint8
 30  etype_ohcf        205 non-null    uint8
 31  etype_ohcv        205 non-null    uint8
 32  etype_rotor       205 non-null    uint8
 33  fsystem_1bbl      205 non-null    uint8
 34  fsystem_2bbl      205 non-null    uint8
 35  fsystem_4bbl      205 non-null    uint8
 36  fsystem_idi       205 non-null    uint8
 37  fsystem_mfi       205 non-null    uint8
 38  fsystem_mpfi      205 non-null    uint8
 39  fsystem_spdi      205 non-null    uint8
 40  fsystem_spfi      205 non-null    uint8
 41  e_location_0      205 non-null    uint8
 42  e_location_rear   205 non-null    uint8
dtypes: float64(8), int64(13), uint8(22)
memory usage: 38.2 KB
```

[67]:
```
X=dataset_new.drop(["price"],axis=1)
y=dataset_new["price"]
```

[68]:
```python
len(dataset_new.columns)
```

[68]: 43

11 Step-6: Dataset Splitting

[69]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X,y , test_size = 0.2,␣
↪random_state = 0)
print(X_train.shape)
print(X_test.shape)
```

```
(164, 42)
(41, 42)
```

12 Step-7: Model Implementation

(1) Apply Linear regression,Lasso,Ridge and elastic net regression on the model.

(2) Evaluate performance of different model.

(3) Comment your observation

[70]:
```python
from sklearn.linear_model import LinearRegression

reg = LinearRegression().fit(X_train, y_train)
```

[72]:
```python
reg.score(X_train, y_train)
```

[72]: 0.956070774146671

[73]:
```python
reg.coef_
```

[73]:
```
array([-1.39834906e+01,  5.65990898e+02,  6.15090102e+03,  1.89541668e+03,
        5.55971157e+02,  2.53161175e+03,  2.36765487e+02, -4.09007253e+01,
        5.63646392e+02,  1.84383691e+02,  5.07308735e+00,  1.33461886e+03,
        7.26129191e+01, -1.82848445e+03, -2.44930615e+03, -9.56663239e+02,
       -1.86609539e+01,  1.61651527e+00,  5.25998459e+01,  8.13788134e+01,
        3.41415840e+03, -7.23092654e+02, -4.56745994e+02, -3.04057644e+02,
       -1.93026211e+03, -7.37875486e+02,  0.00000000e+00, -6.18974485e+03,
        5.20783434e+02,  1.00372714e+03, -3.55651366e+03,  8.95962342e+03,
       -3.56444407e+01, -8.12430759e+00, -1.53848439e+03,  6.15090102e+03,
       -2.10733609e+03,  7.47346167e+02, -1.23721227e+03, -1.97144569e+03,
       -7.18110754e+03,  7.18110754e+03])
```

[74]:
```
len(reg.coef_)
```

[74]: 42

[75]:
```
reg.intercept_
```

[75]: -60769.327782530134

[76]:
```
y_pred = reg.predict(X_test)
```

[78]:
```
from sklearn.metrics import mean_squared_error

MSE  = mean_squared_error(y_test,y_pred)
print("MSE :" , MSE)

RMSE = np.sqrt(MSE)
print("RMSE :" ,RMSE)
```

```
MSE : 14490755.960495027
RMSE : 3806.6725575619225
```

[79]:
```
from sklearn.metrics import r2_score
r2 = r2_score(y_test,y_pred)
print("R2 :" ,r2)
```

```
R2 : 0.8128204904846368
```

[80]:
```
from sklearn.linear_model import Lasso
lasso  = Lasso(alpha=0.0001 , max_iter= 3000)

lasso.fit(X_train, y_train)
```

[80]: Lasso(alpha=0.0001, max_iter=3000)

[81]:
```
lasso.score(X_train, y_train)
```

[81]: 0.9560707741460119

[82]:
```
lasso.coef_
```

[82]:
```
array([-1.39835200e+01,  5.65989390e+02,  3.32534801e+03,  1.89541830e+03,
        5.55970164e+02,  2.53160999e+03,  2.36765298e+02, -4.09011240e+01,
        5.63647213e+02,  1.84384363e+02,  5.07308389e+00,  1.33461104e+03,
        7.26130345e+01, -1.82848763e+03, -2.44929990e+03, -9.56649538e+02,
       -1.86608460e+01,  1.61651101e+00,  5.25992111e+01,  8.13787277e+01,
        4.25450742e+03,  1.17255303e+02,  3.83606606e+02,  5.36295892e+02,
       -1.08990754e+03, -8.94070556e+02,  0.00000000e+00, -6.34592760e+03,
```

```
 3.64586780e+02,  8.47534733e+02, -3.71270453e+03,  8.80337191e+03,
-1.44916536e+03, -1.42164071e+03, -2.95195558e+03,  7.56275630e+03,
-3.52082787e+03, -6.66167590e+02, -2.65071430e+03, -3.38494725e+03,
-1.43621992e+04,  9.60023061e-09])
```

[83]:
```
from sklearn.model_selection import GridSearchCV
lasso = Lasso()
parameters = {'alpha':␣
↪[1e-15,1e-13,1e-10,1e-8,1e-5,1e-4,1e-3,1e-2,1e-1,1,5,10,20,30,40,45,50,55,60,100,0.
↪0014]}
lasso_regressor = GridSearchCV(lasso, parameters,␣
↪scoring='neg_mean_squared_error', cv=5)
lasso_regressor.fit(X_train, y_train)
```

[83]:
```
GridSearchCV(cv=5, estimator=Lasso(),
             param_grid={'alpha': [1e-15, 1e-13, 1e-10, 1e-08, 1e-05, 0.0001,
                                   0.001, 0.01, 0.1, 1, 5, 10, 20, 30, 40, 45,
                                   50, 55, 60, 100, 0.0014]},
             scoring='neg_mean_squared_error')
```

[84]:
```
print("The best fit alpha value is found out to be :" ,lasso_regressor.
↪best_params_)
print("\nUsing ",lasso_regressor.best_params_, " the negative mean squared␣
↪error is: ", lasso_regressor.best_score_)
```

```
The best fit alpha value is found out to be : {'alpha': 1}

Using  {'alpha': 1}  the negative mean squared error is:  -5715234.081444084
```

[86]:
```
y_pred_lasso = lasso_regressor.predict(X_test)
```

[87]:
```
MSE  = mean_squared_error(y_test,y_pred_lasso)
print("MSE :" , MSE)

RMSE = np.sqrt(MSE)
print("RMSE :" ,RMSE)

r2 = r2_score(y_test,y_pred_lasso)
print("R2 :" ,r2)
```

```
MSE : 14370473.87325172
RMSE : 3790.8407871146105
R2 : 0.8143741942496484
```

[88]:
```
from sklearn.linear_model import Ridge
ridge = Ridge()
```

```
parameters = {'alpha':␣
↪[1e-15,1e-10,1e-8,1e-5,1e-4,1e-3,1e-2,1,5,10,20,30,40,45,50,55,60,100]}
ridge_regressor = GridSearchCV(ridge, parameters,␣
↪scoring='neg_mean_squared_error', cv=3)
ridge_regressor.fit(X_train,y_train)
```

[88]:
```
GridSearchCV(cv=3, estimator=Ridge(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 1e-05, 0.0001, 0.001,
                                   0.01, 1, 5, 10, 20, 30, 40, 45, 50, 55, 60,
                                   100]},
             scoring='neg_mean_squared_error')
```

[89]:
```
print("The best fit alpha value is found out to be :" ,ridge_regressor.
↪best_params_)
print("\nUsing ",ridge_regressor.best_params_, " the negative mean squared␣
↪error is: ", ridge_regressor.best_score_)
```

```
The best fit alpha value is found out to be : {'alpha': 1e-08}

Using  {'alpha': 1e-08}  the negative mean squared error is:  -6297757.865667018
```

[90]:
```
y_pred_ridge = ridge_regressor.predict(X_test)
```

[92]:
```
MSE  = mean_squared_error(y_test,y_pred_ridge)
print("MSE :" , MSE)

RMSE = np.sqrt(MSE)
print("RMSE :" ,RMSE)

r2 = r2_score(y_test,y_pred_ridge)
print("R2 :" ,r2)
```

```
MSE : 14490753.857985102
RMSE : 3806.672281400791
R2 : 0.8128205176431078
```

[93]:
```
from sklearn.linear_model import ElasticNet
#a * L1 + b * L2
#alpha = a + b and l1_ratio = a / (a + b)
elasticnet = ElasticNet(alpha=0.1, l1_ratio=0.5)
```

[94]:
```
elasticnet.fit(X_train,y_train)
```

[94]:
```
ElasticNet(alpha=0.1)
```

[95]:
```
elasticnet.score(X_train, y_train)
```

[95]: 0.9215222910941743

[96]:
```
y_pred_en = elasticnet.predict(X_test)
```

[98]:
```
MSE  = mean_squared_error(y_test,y_pred_en)
print("MSE :" , MSE)
RMSE = np.sqrt(MSE)
print("RMSE :" ,RMSE)
r2 = r2_score(y_test,y_pred_en)
print("R2 :" ,r2)
```

```
MSE : 12008339.137873283
RMSE : 3465.305056971649
R2 : 0.8448862822582186
```