In [1]:
```
import pandas as pd
import numpy as np
```

In [4]:
```
df = pd.read_csv("C:/Users/pravi/Desktop/Python_April/Automobile_data.csv" )
```

In [8]:
```
df.head(5)
```

Out[8]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4

5 rows × 26 columns

In [11]:
```
df.tail()
```

Out[11]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1

5 rows × 26 columns

In [14]:
```
orig_col_names = df.columns
print(orig_col_names)
```

```
Index(['symboling', 'normalized-losses', 'make', 'fuel-type', 'aspiration',
       'num-of-doors', 'body-style', 'drive-wheels', 'engine-location',
       'wheel-base', 'length', 'width', 'height', 'curb-weight', 'engine-typ
e',
       'num-of-cylinders', 'engine-size', 'fuel-system', 'bore', 'stroke',
       'compression-ratio', 'horsepower', 'peak-rpm', 'city-mpg',
       'highway-mpg', 'price'],
      dtype='object')
```

In [17]:
```
# Column headers and mutability
df_new = df.copy()
df_new.columns = ['Symboling', 'normalized-losses', 'make', 'fuel-type', 'aspi
ration',
       'num-of-doors', 'body-style', 'drive-wheels', 'engine-location',
       'wheel-base', 'length', 'width', 'height', 'curb-weight', 'engine-type'
,
       'num-of-cylinders', 'engine-size', 'fuel-system', 'bore', 'stroke',
       'compression-ratio', 'horsepower', 'peak-rpm', 'city-mpg',
       'highway-mpg', 'price']
df_new.head()
```

Out[17]:

	Symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4

5 rows × 26 columns

In [ ]:
```
df = df_new.copy()
```

In [18]:
```
df.head()
```

Out[18]:

	Symbols	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4

5 rows × 26 columns

In [20]:
```
df_test = df[["make","Symbols"]]
df_test["make"]=0
df_test.head()
```

```
C:\Users\pravi\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[20]:

	make	Symbols
0	0	3
1	0	3
2	0	1
3	0	2
4	0	2

In [21]:
```
df.head()
```

Out[21]:

	Symbols	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4

5 rows × 26 columns

In [22]:
```
# export dataframe
df.to_csv("auto.csv", sep = ',')
```

In [ ]:
```
#df["Symbol"].to_list()
```

In [23]:
```
df.iloc[1:4, 4:7]
```

Out[23]:

	aspiration	num-of-doors	body-style
1	std	two	convertible
2	std	two	hatchback
3	std	four	sedan

In [26]:
```
df['Symbols']
```

Out[26]:
```
0      3
1      3
2      1
3      2
4      2
      ..
200   -1
201   -1
202   -1
203   -1
204   -1
Name: Symbols, Length: 205, dtype: int64
```

In [27]:
```
# Check/Analyze the data
df.dtypes
```

Out[27]:
```
Symbols                int64
normalized-losses     object
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                  object
stroke                object
compression-ratio    float64
horsepower            object
peak-rpm              object
city-mpg               int64
highway-mpg            int64
price                 object
dtype: object
```

In [28]:
```
df.describe()
```

Out[28]:

	Symbols	wheel-base	length	width	height	curb-weight	engine-size	co
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	

```
In [29]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Symbols            205 non-null    int64  
 1   normalized-losses  205 non-null    object 
 2   make               205 non-null    object 
 3   fuel-type          205 non-null    object 
 4   aspiration         205 non-null    object 
 5   num-of-doors       205 non-null    object 
 6   body-style         205 non-null    object 
 7   drive-wheels       205 non-null    object 
 8   engine-location    205 non-null    object 
 9   wheel-base         205 non-null    float64
 10  length             205 non-null    float64
 11  width              205 non-null    float64
 12  height             205 non-null    float64
 13  curb-weight        205 non-null    int64  
 14  engine-type        205 non-null    object 
 15  num-of-cylinders   205 non-null    object 
 16  engine-size        205 non-null    int64  
 17  fuel-system        205 non-null    object 
 18  bore               205 non-null    object 
 19  stroke             205 non-null    object 
 20  compression-ratio  205 non-null    float64
 21  horsepower         205 non-null    object 
 22  peak-rpm           205 non-null    object 
 23  city-mpg           205 non-null    int64  
 24  highway-mpg        205 non-null    int64  
 25  price              205 non-null    object 
dtypes: float64(5), int64(5), object(16)
memory usage: 41.8+ KB
```

```
In [ ]: x = list[1,2]
```

```
In [33]: # Accessing columns
         x = np.array(df["body-style"])
```

In [35]:

```
# Missing value

df.replace('?', np.nan, inplace=True)
df.head()
```

Out[35]:

	Symbols	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4

5 rows × 26 columns

In [36]:

```
df.shape
```

Out[36]:

```
(205, 26)
```

In [37]:

```
print(df.shape)
df.dropna(subset = ["price"], axis = 0)
print(df.shape)
df.dropna(subset = ["price"], axis = 0, inplace = True)
print(df.shape)
```

```
(205, 26)
(205, 26)
(201, 26)
```

In [49]:

```
df_null = df.isnull()
df_null.head(5)
```

Out[49]:

	Symbols	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	e
0	False	False	False	False	False	False	False	False	False	False	...	
1	False	False	False	False	False	False	False	False	False	False	...	
2	False	False	False	False	False	False	False	False	False	False	...	
3	False	False	False	False	False	False	False	False	False	False	...	
4	False	False	False	False	False	False	False	False	False	False	...	

5 rows × 26 columns

In [51]:

```
# Find columns with missing data and number of missing values
for col in df_null.columns:
#    print(col)
    if True in df_null[col].unique():
        print("in loop",col)
        print(df_null[col].value_counts())
```

```
in loop num-of-doors
False    199
True       2
Name: num-of-doors, dtype: int64
in loop bore
False    197
True       4
Name: bore, dtype: int64
in loop stroke
False    197
True       4
Name: stroke, dtype: int64
in loop horsepower
False    199
True       2
Name: horsepower, dtype: int64
in loop peak-rpm
False    199
True       2
Name: peak-rpm, dtype: int64
```

In [44]:

```
# Unique values in a column
print(df["normalized-losses"].unique())
```

```
[nan '164' '158' '192' '188' '121' '98' '81' '118' '148' '110' '145' '137'
 '101' '78' '106' '85' '107' '104' '113' '150' '129' '115' '93' '142'
 '161' '153' '125' '128' '122' '103' '168' '108' '194' '231' '119' '154'
 '74' '186' '83' '102' '89' '87' '77' '91' '134' '65' '197' '90' '94'
 '256' '95']
```

In [45]:

```
# Change datatype of column
df["normalized-losses"] = df["normalized-losses"].astype("float")
```

In [46]:

```
# replace missing values of column normalized-losses bu mean value
df["normalized-losses"].replace(np.nan, df["normalized-losses"].mean(),inplace
=True)
```

In [66]:
```
df.tail(20)
```

Out[66]:

	Symbols	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
185	2	94	volkswagen	gas	std	four	sedan	fwd	front
186	2	94	volkswagen	gas	std	four	sedan	fwd	front
187	2	94	volkswagen	diesel	turbo	four	sedan	fwd	front
188	2	94	volkswagen	gas	std	four	sedan	fwd	front
189	3	122	volkswagen	gas	std	two	convertible	fwd	front
190	3	256	volkswagen	gas	std	two	hatchback	fwd	front
191	0	122	volkswagen	gas	std	four	sedan	fwd	front
192	0	122	volkswagen	diesel	turbo	four	sedan	fwd	front
193	0	122	volkswagen	gas	std	four	wagon	fwd	front
194	-2	103	volvo	gas	std	four	sedan	rwd	front
195	-1	74	volvo	gas	std	four	wagon	rwd	front
196	-2	103	volvo	gas	std	four	sedan	rwd	front
197	-1	74	volvo	gas	std	four	wagon	rwd	front
198	-2	103	volvo	gas	turbo	four	sedan	rwd	front
199	-1	74	volvo	gas	turbo	four	wagon	rwd	front
200	-1	95	volvo	gas	std	four	sedan	rwd	front
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front
202	-1	95	volvo	gas	std	four	sedan	rwd	front
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front

20 rows × 26 columns

In [ ]:
```
# Check the unique values, data-type of the columns with missing value
# replace missing values of columns - bore, stroke, horsepower, peak-rpm by me
an value and that of number of doors by the value with highest frequency
```

In [ ]:
```

```

In [53]:
```
#df["bore"].replace(np.nan, df["bore"].mean(),inplace=True)
```

In [54]:
```
df['bore'] = df['bore'].astype("float")
```

In [55]:
```
df["bore"].replace(np.nan, df["bore"].mean(),inplace=True)
```

In [56]:

```
df['stroke'] = df['stroke'].astype("float")
df['horsepower'] = df['horsepower'].astype("float")
df['peak-rpm'] = df['peak-rpm'].astype("float")
```

In [57]:

```
df["stroke"].replace(np.nan, df["stroke"].mean(),inplace=True)
df["horsepower"].replace(np.nan, df["horsepower"].mean(),inplace=True)
df["peak-rpm"].replace(np.nan, df["peak-rpm"].mean(),inplace=True)
```

In [58]:

```
print(df["num-of-doors"].value_counts())
df["num-of-doors"].value_counts().idxmax()
```

```
four    113
two      86
Name: num-of-doors, dtype: int64
```

Out[58]:

```
'four'
```

In [59]:

```
df["num-of-doors"].replace(np.nan, df["num-of-doors"].value_counts().idxmax(),
inplace=True)
```

In [60]:

```
df.dtypes
```

Out[60]:

```
Symbols                int64
normalized-losses    float64
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                 float64
stroke               float64
compression-ratio    float64
horsepower           float64
peak-rpm             float64
city-mpg               int64
highway-mpg            int64
price                 object
dtype: object
```

In [62]:

```
df[["bore", "stroke"]] = df[["bore", "stroke"]].astype("float")
df[["normalized-losses"]] = df[["normalized-losses"]].astype("int")
df[["price"]] = df[["price"]].astype("float")
df[["peak-rpm"]] = df[["peak-rpm"]].astype("float")
```

In [61]:

```
df['normalized-losses'].unique()
```

Out[61]:

```
array([122., 164., 158., 192., 188., 121.,  98.,  81., 118., 148., 110.,
       145., 137., 101.,  78., 106.,  85., 107., 104., 113., 150., 129.,
       115.,  93., 142., 161., 153., 125., 128., 103., 168., 108., 194.,
       231., 119., 154.,  74., 186.,  83., 102.,  89.,  87.,  77.,  91.,
       134.,  65., 197.,  90.,  94., 256.,  95.])
```

In [63]:

```
df['fuel-type'].unique()
```

Out[63]:

```
array(['gas', 'diesel'], dtype=object)
```

In [ ]:

```
#df['fuel-type'].replace(['gas','diesel'], [0,1], inplace=True)
```

In [67]:

```
dummy_variable_1 = pd.get_dummies(df["fuel-type"])
dummy_variable_1.head()
```

Out[67]:

	diesel	gas
0	0	1
1	0	1
2	0	1
3	0	1
4	0	1

In [69]:

```
dummy_variable_1.rename(columns={'gas':'fuel-type-diesel', 'diesel':'fuel-type
-gas'}, inplace=True)
dummy_variable_1.head()
```

Out[69]:

	fuel-type-diesel	fuel-type-diesel
0	0	1
1	0	1
2	0	1
3	0	1
4	0	1

In [70]:

```
df = pd.concat([df, dummy_variable_1], axis=0)
```

In [71]:

```
df.head()
```

Out[71]:

	Symbols	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
0	3	122	alfa-romero	gas	std	two	convertible	rwd	front	88.6
1	3	122	alfa-romero	gas	std	two	convertible	rwd	front	88.6
2	1	122	alfa-romero	gas	std	two	hatchback	rwd	front	94.5
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4

5 rows × 28 columns

In [72]:

```
df.drop("fuel-type", axis = 1, inplace=True)
```

In [73]:

```
df.head()
```

Out[73]:

	Symbols	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length
0	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	168.8
1	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	168.8
2	1	122	alfa-romero	std	two	hatchback	rwd	front	94.5	171.2
3	2	164	audi	std	four	sedan	fwd	front	99.8	176.6
4	2	164	audi	std	four	sedan	4wd	front	99.4	176.6

5 rows × 27 columns

In [74]:

```
df["Temp"] = (df["Symbols"] * 6 ) /3
```

In [75]:
```
df.head()
```

Out[75]:

	Symbols	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length
0	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	168.8
1	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	168.8
2	1	122	alfa-romero	std	two	hatchback	rwd	front	94.5	171.2
3	2	164	audi	std	four	sedan	fwd	front	99.8	176.6
4	2	164	audi	std	four	sedan	4wd	front	99.4	176.6

5 rows × 28 columns

In [76]:
```
from sklearn.linear_model import LinearRegression
```

In [ ]:
```

```

In [77]:
```
df.dtypes
```

Out[77]:
```
Symbols                int64
normalized-losses      int32
make                  object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                 float64
stroke               float64
compression-ratio    float64
horsepower           float64
peak-rpm             float64
city-mpg               int64
highway-mpg            int64
price                float64
fuel-type-diesel       uint8
fuel-type-diesel       uint8
Temp                 float64
dtype: object
```

In [79]:
```
x = df[['horsepower','stroke', 'engine-size', 'highway-mpg']]
x.head()
```

Out[79]:

	horsepower	stroke	engine-size	highway-mpg
0	111.0	2.68	130	27
1	111.0	2.68	130	27
2	154.0	3.47	152	26
3	102.0	3.40	109	30
4	115.0	3.40	136	22

In [80]:
```
Linear_model = LinearRegression()
```

In [81]:
```
Linear_model.fit(x, df['price'])
```

Out[81]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fals
e)
```

In [82]:
```
y_predict = Linear_model.predict(x)
print(y_predict)
Linear_model.score(x,df['price'])
```

```
[15468.2607231  15468.2607231  18570.95731511 10703.98308272
 15644.67758135 15068.16808026 15068.16808026 15068.16808026
 16252.15001326 11808.02710559 11808.02710559 19066.99689582
 19066.99689582 19446.00985016 27547.40351676 27547.40351676
 27800.07881966   296.78065028  5946.21943761  5946.21943761
  5891.14756146  6270.1605158   9331.13713236  6270.1605158
  6270.1605158   6270.1605158   9331.13713236 11681.60104394
 18154.60497641  3763.71640053  6496.10639197  4361.90518528
  7001.45699776  7001.45699776  7001.45699776  7001.45699776
  9476.39545922  9476.39545922  9476.39545922  9476.39545922
 10700.5733567  10282.06109301 10459.47201215 11945.76069191
 32402.87389907 32402.87389907 47344.80961429  7433.5602371
  6549.19667697  6549.19667697  6549.19667697  6549.19667697
  6886.67550143  6886.67550143  6886.67550143  9495.02322876
 11404.36516385 11404.36516385 11404.36516385 11404.36516385
  9351.00246239 11404.36516385 16174.13147533 11087.89584215
 21071.40666552 21071.40666552 21071.40666552 21071.40666552
 30702.50981455 30702.50981455 40987.85253778 40481.704054
 18801.85596401  6144.22180335  6523.23475769  6523.23475769
  9331.13713236 11269.13625439 11428.92574104 18230.85449652
 18230.85449652 18230.85449652 11428.92574104 11428.92574104
 11269.13625439 11269.13625439  7207.38304304  5428.10312849
  7207.38304304  7207.38304304  7207.38304304  7207.38304304
  7207.38304304  7207.38304304  7207.38304304  7207.38304304
 11259.60760038 11259.60760038 23048.13341016 23048.13341016
 22669.12045583 22985.11493062 24817.7626075  22985.11493062
 13056.73075564 15260.82260321 13056.73075564 16271.52381479
 14883.9701398  15260.82260321 14883.9701398  16271.52381479
 13056.73075564 15260.82260321 16567.59460955  5891.14756146
  9331.13713236  6270.1605158   6270.1605158   7282.45748336
 11681.60104394 18230.85449652 18569.8358209  27191.87310339
 27191.87310339 27191.87310339 12590.06015127 12590.06015127
 13420.15685269 13420.15685269 15326.39485555 13420.15685269
 15647.79762307 15647.79762307  9106.52203714 10754.36922136
 10754.36922136 10351.83709683 10857.18770262 11457.51706626
 11867.88891419 12508.01827954 10983.52535406 11583.8547177
 11362.5383084  13266.04418822  6541.14885072  6667.48650216
  6667.48650216  6793.82415361  7425.51241084  7425.51241084
  7869.04135407  7869.04135407  8350.83796505  6961.12379913
  6605.66483961  8248.05430841  8248.05430841  8248.05430841
  8248.05430841 10443.40165819 10443.40165819 15748.22308828
 15748.22308828 15748.22308828 15748.22308828 15748.22308828
 15748.22308828 11181.74863532  9401.33917833 11434.42393821
 11434.42393821 11434.42393821 21733.08164174 21733.08164174
 21535.58509499 20270.21388555  5189.16974076  9527.14421799
  5189.16974076  9527.14421799  9527.14421799  6326.50929614
 10372.30916113 10356.32902197 10356.32902197 15194.50573171
  6831.85990193 10024.65510038 15956.39746875 15956.39746875
 15956.39746875 15956.39746875 17218.48189583 17218.48189583
 15956.39746875 18152.37865318 21961.00642401 15796.32962847
 16335.41042309]
```

```
Out[82]: 0.7975597736149305
```

```
In [83]: from sklearn.metrics import mean_squared_error
```

In [84]:

```
mse = mean_squared_error(df['price'],y_predict)
mse
```

Out[84]:

```
12721678.898490274
```

In [ ]: