Vaibhav Kumar

Roll No : 19

Multivarient Regression - mpg dataset

In [45]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,OneHotEncoder
```

In [46]:
```
data=pd.read_fwf("D:\\vk\\TRIM 3\\ML\\DATASET\\auto-mpg.data")
data
```

Out[46]:

	18.0	8	307.0	130.0	3504.	12.0	70	1	"chevrolet chevelle malibu"
0	15.0	8	350.0	165.0	3693.0	11.5	70	1	"buick skylark 320"
1	18.0	8	318.0	150.0	3436.0	11.0	70	1	"plymouth satellite"
2	16.0	8	304.0	150.0	3433.0	12.0	70	1	"amc rebel sst"
3	17.0	8	302.0	140.0	3449.0	10.5	70	1	"ford torino"
4	15.0	8	429.0	198.0	4341.0	10.0	70	1	"ford galaxie 500"
...	...	...	...	...	...	...	...	...	...
392	27.0	4	140.0	86.00	2790.0	15.6	82	1	"ford mustang gl"
393	44.0	4	97.0	52.00	2130.0	24.6	82	2	"vw pickup"
394	32.0	4	135.0	84.00	2295.0	11.6	82	1	"dodge rampage"
395	28.0	4	120.0	79.00	2625.0	18.6	82	1	"ford ranger"
396	31.0	4	119.0	82.00	2720.0	19.4	82	1	"chevy s-10"

397 rows × 9 columns

In [47]:
```
col_names=['mpg','cylinder','displacement','horsepower','weight','acceleration','model
```

In [48]:
```
data.columns=col_names
```

In [49]:
```
data
```

Out[49]:

	mpg	cylinder	displacement	horsepower	weight	acceleration	modelyear	origin	carname
0	15.0	8	350.0	165.0	3693.0	11.5	70	1	"buick skylark 320"
1	18.0	8	318.0	150.0	3436.0	11.0	70	1	"plymouth satellite"
2	16.0	8	304.0	150.0	3433.0	12.0	70	1	"amc rebel sst"
3	17.0	8	302.0	140.0	3449.0	10.5	70	1	"ford torino"
4	15.0	8	429.0	198.0	4341.0	10.0	70	1	"ford galaxie 500"
...	...	...	...	...	...	...	...	...	...
392	27.0	4	140.0	86.00	2790.0	15.6	82	1	"ford mustang gl"
393	44.0	4	97.0	52.00	2130.0	24.6	82	2	"vw pickup"
394	32.0	4	135.0	84.00	2295.0	11.6	82	1	"dodge rampage"
395	28.0	4	120.0	79.00	2625.0	18.6	82	1	"ford ranger"
396	31.0	4	119.0	82.00	2720.0	19.4	82	1	"chevy s-10"

397 rows × 9 columns

In [50]:
```
data.head()
```

Out[50]:

	mpg	cylinder	displacement	horsepower	weight	acceleration	modelyear	origin	carname
0	15.0	8	350.0	165.0	3693.0	11.5	70	1	"buick skylark 320"
1	18.0	8	318.0	150.0	3436.0	11.0	70	1	"plymouth satellite"
2	16.0	8	304.0	150.0	3433.0	12.0	70	1	"amc rebel sst"
3	17.0	8	302.0	140.0	3449.0	10.5	70	1	"ford torino"
4	15.0	8	429.0	198.0	4341.0	10.0	70	1	"ford galaxie 500"

In [51]:
```
data.describe()
```

Out[51]:

	mpg	cylinder	displacement	weight	acceleration	modelyear	origin
count	397.000000	397.000000	397.000000	397.000000	397.000000	397.000000	397.000000
mean	23.528463	5.448363	193.139798	2969.080605	15.577078	76.025189	1.574307
std	7.820926	1.698329	104.244898	847.485218	2.755326	3.689922	0.802549
min	9.000000	3.000000	68.000000	1613.000000	8.000000	70.000000	1.000000
25%	17.500000	4.000000	104.000000	2223.000000	13.900000	73.000000	1.000000
50%	23.000000	4.000000	146.000000	2800.000000	15.500000	76.000000	1.000000
75%	29.000000	8.000000	262.000000	3609.000000	17.200000	79.000000	2.000000
max	46.600000	8.000000	455.000000	5140.000000	24.800000	82.000000	3.000000

In [52]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 397 entries, 0 to 396
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           397 non-null    float64
 1   cylinder      397 non-null    int64  
 2   displacement  397 non-null    float64
 3   horsepower    397 non-null    object 
 4   weight        397 non-null    float64
 5   acceleration  397 non-null    float64
 6   modelyear     397 non-null    int64  
 7   origin        397 non-null    int64  
 8   carname       397 non-null    object 
dtypes: float64(4), int64(3), object(2)
memory usage: 28.0+ KB
```

In [53]:
```
data.shape
```

Out[53]:
```
(397, 9)
```

In [54]:
```
data.isnull()
```

Out[54]:

	mpg	cylinder	displacement	horsepower	weight	acceleration	modelyear	origin	carname
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
392	False	False	False	False	False	False	False	False	False
393	False	False	False	False	False	False	False	False	False
394	False	False	False	False	False	False	False	False	False
395	False	False	False	False	False	False	False	False	False
396	False	False	False	False	False	False	False	False	False

397 rows × 9 columns

In [55]:
```
data['carname'].value_counts()
```

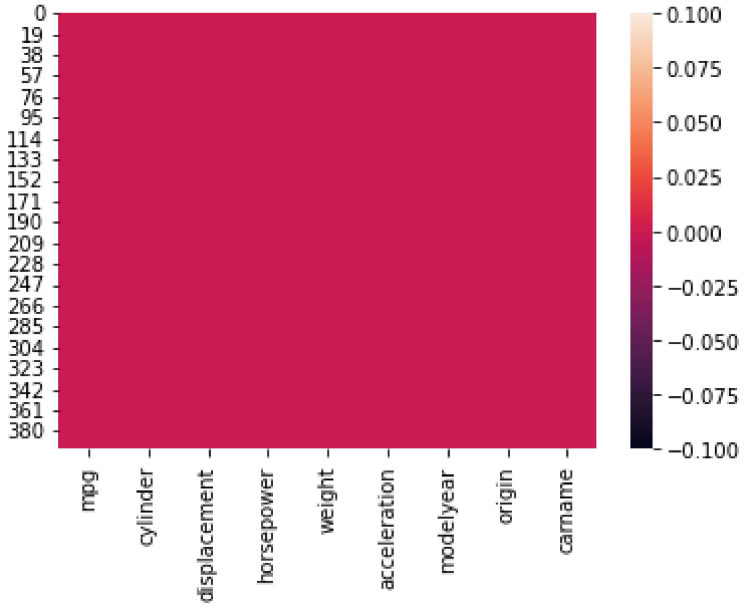
Out[55]:
```
"ford pinto"             6
"amc matador"            5
"toyota corolla"         5
"ford maverick"          5
"amc hornet"             4
                        ..
"chevrolet monza 2+2"    1
"ford mustang ii"        1
"pontiac astro"          1
"amc pacer"              1
"chevy s-10"             1
Name: carname, Length: 305, dtype: int64
```

In [56]:
```
type(data)
```

Out[56]:
```
pandas.core.frame.DataFrame
```

In [57]:
```
sns.heatmap(data.isnull())
```

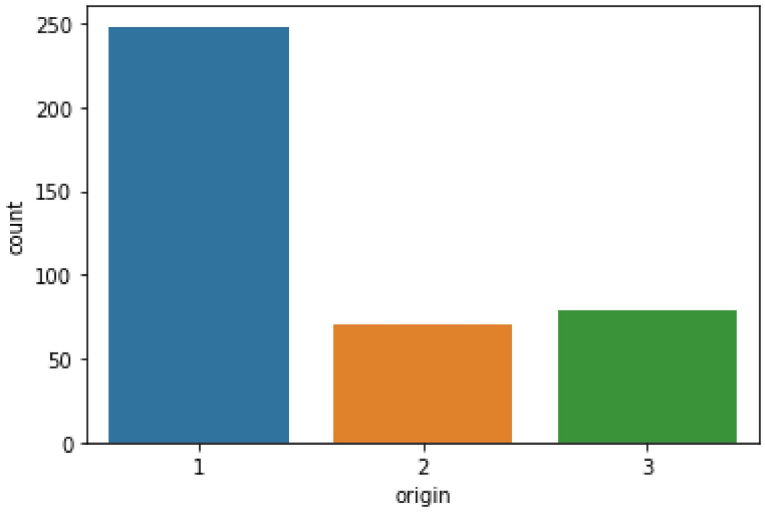
Out[57]:
```
<AxesSubplot:>
```



no null value is there

In [58]:
```
sns.countplot(x='origin',data=data)
```

Out[58]:
```
<AxesSubplot:xlabel='origin', ylabel='count'>
```



In [59]:
```
data['carname'].unique()
```

Out[59]:
```
array(['"buick skylark 320"', '"plymouth satellite"', '"amc rebel sst"',
       '"ford torino"', '"ford galaxie 500"', '"chevrolet impala"',
       '"plymouth fury iii"', '"pontiac catalina"',
       '"amc ambassador dpl"', '"dodge challenger se"',
       '"plymouth \'cuda 340"', '"chevrolet monte carlo"',
       '"buick estate wagon (sw)"', '"toyota corona mark ii"',
       '"plymouth duster"', '"amc hornet"', '"ford maverick"',
       '"datsun pl510"', '"volkswagen 1131 deluxe sedan"',
       '"peugeot 504"', '"audi 100 ls"', '"saab 99e"', '"bmw 2002"',
       '"amc gremlin"', '"ford f250"', '"chevy c20"', '"dodge d200"',
       '"hi 1200d"', '"chevrolet vega 2300"', '"toyota corona"',
       '"ford pinto"', '"plymouth satellite custom"',
       '"chevrolet chevelle malibu"', '"ford torino 500"',
       '"amc matador"', '"pontiac catalina brougham"',
       '"dodge monaco (sw)"', '"ford country squire (sw)"',
       '"pontiac safari (sw)"', '"amc hornet sportabout (sw)"',
       '"chevrolet vega (sw)"', '"pontiac firebird"', '"ford mustang"',
       '"mercury capri 2000"', '"opel 1900"', '"peugeot 304"',
       '"fiat 124b"', '"toyota corolla 1200"', '"datsun 1200"',
       '"volkswagen model 111"', '"plymouth cricket"',
       '"toyota corona hardtop"', '"dodge colt hardtop"',
       '"volkswagen type 3"', '"chevrolet vega"', '"ford pinto runabout"',
       '"amc ambassador sst"', '"mercury marquis"',
       '"buick lesabre custom"', '"oldsmobile delta 88 royale"',
       '"chrysler newport royal"', '"mazda rx2 coupe"',
       '"amc matador (sw)"', '"chevrolet chevelle concours (sw)"',
       '"ford gran torino (sw)"', '"plymouth satellite custom (sw)"',
       '"volvo 145e (sw)"', '"volkswagen 411 (sw)"', '"peugeot 504 (sw)"',
       '"renault 12 (sw)"', '"ford pinto (sw)"', '"datsun 510 (sw)"',
       '"toyouta corona mark ii (sw)"', '"dodge colt (sw)"',
       '"toyota corolla 1600 (sw)"', '"buick century 350"',
       '"chevrolet malibu"', '"ford gran torino"',
       '"dodge coronet custom"', '"mercury marquis brougham"',
       '"chevrolet caprice classic"', '"ford ltd"',
       '"plymouth fury gran sedan"', '"chrysler new yorker brougham"',
       '"buick electra 225 custom"', '"amc ambassador brougham"',
       '"plymouth valiant"', '"chevrolet nova custom"',
       '"volkswagen super beetle"', '"ford country"',
       '"plymouth custom suburb"', '"oldsmobile vista cruiser"',
       '"toyota carina"', '"datsun 610"', '"maxda rx3"',
       '"mercury capri v6"', '"fiat 124 sport coupe"',
       '"chevrolet monte carlo s"', '"pontiac grand prix"', '"fiat 128"',
       '"opel manta"', '"audi 100ls"', '"volvo 144ea"',
       '"dodge dart custom"', '"saab 99le"', '"toyota mark ii"',
       '"oldsmobile omega"', '"chevrolet nova"', '"datsun b210"',
       '"chevrolet chevelle malibu classic',
       '"plymouth satellite sebring"', '"buick century luxus (sw)"',
       '"dodge coronet custom (sw)"', '"audi fox"', '"volkswagen dasher"',
       '"datsun 710"', '"dodge colt"', '"fiat 124 tc"', '"honda civic"',
       '"subaru"', '"fiat x1.9"', '"plymouth valiant custom"',
       '"mercury monarch"', '"chevrolet bel air"',
       '"plymouth grand fury"', '"buick century"',
       '"chevroelt chevelle malibu"', '"plymouth fury"',
       '"buick skyhawk"', '"chevrolet monza 2+2"', '"ford mustang ii"',
       '"toyota corolla"', '"pontiac astro"', '"volkswagen rabbit"',
       '"amc pacer"', '"volvo 244dl"', '"honda civic cvcc"', '"fiat 131"',
       '"capri ii"', '"renault 12tl"', '"dodge coronet brougham"',
       '"chevrolet chevette"', '"chevrolet woody"', '"vw rabbit"',
       '"dodge aspen se"', '"ford granada ghia"', '"pontiac ventura sj"',
       '"amc pacer d/l"', '"datsun b-210"', '"volvo 245"',
```

```
'"plymouth volare premier v8"', '"mercedes-benz 280s"',
'"cadillac seville"', '"chevy c10"', '"ford f108"', '"dodge d100"',
'"honda accord cvcc"', '"buick opel isuzu deluxe"',
'"renault 5 gtl"', '"plymouth arrow gs"',
'"datsun f-10 hatchback"', '"oldsmobile cutlass supreme"',
'"dodge monaco brougham"', '"mercury cougar brougham"',
'"chevrolet concours"', '"buick skylark"',
'"plymouth volare custom"', '"ford granada"',
'"pontiac grand prix lj"', '"chevrolet monte carlo landau"',
'"chrysler cordoba"', '"ford thunderbird"',
'"volkswagen rabbit custom"', '"pontiac sunbird coupe"',
'"toyota corolla liftback"', '"ford mustang ii 2+2"',
'"dodge colt m/m"', '"subaru dl"', '"datsun 810"', '"bmw 320i"',
'"mazda rx-4"', '"volkswagen rabbit custom diesel"',
'"ford fiesta"', '"mazda glc deluxe"', '"datsun b210 gx"',
'"oldsmobile cutlass salon brougham', '"dodge diplomat"',
'"mercury monarch ghia"', '"pontiac phoenix lj"',
'"ford fairmont (auto)"', '"ford fairmont (man)"',
'"plymouth volare"', '"amc concord"', '"buick century special"',
'"mercury zephyr"', '"dodge aspen"', '"amc concord d/l"',
'"buick regal sport coupe (turbo)"', '"ford futura"',
'"dodge magnum xe"', '"datsun 510"', '"dodge omni"',
'"toyota celica gt liftback"', '"plymouth sapporo"',
'"oldsmobile starfire sx"', '"datsun 200-sx"', '"audi 5000"',
'"volvo 264gl"', '"saab 99gle"', '"peugeot 604sl"',
'"volkswagen scirocco"', '"honda accord lx"',
'"pontiac lemans v6"', '"mercury zephyr 6"', '"ford fairmont 4"',
'"amc concord dl 6"', '"dodge aspen 6"', '"ford ltd landau"',
'"mercury grand marquis"', '"dodge st. regis"',
'"chevrolet malibu classic (sw)"',
'"chrysler lebaron town @ country (', '"vw rabbit custom"',
'"maxda glc deluxe"', '"dodge colt hatchback custom"',
'"amc spirit dl"', '"mercedes benz 300d"', '"cadillac eldorado"',
'"plymouth horizon"', '"plymouth horizon tc3"', '"datsun 210"',
'"fiat strada custom"', '"buick skylark limited"',
'"chevrolet citation"', '"oldsmobile omega brougham"',
'"pontiac phoenix"', '"toyota corolla tercel"', '"datsun 310"',
'"ford fairmont"', '"audi 4000"', '"toyota corona liftback"',
'"mazda 626"', '"datsun 510 hatchback"', '"mazda glc"',
'"vw rabbit c (diesel)"', '"vw dasher (diesel)"',
'"audi 5000s (diesel)"', '"mercedes-benz 240d"',
'"honda civic 1500 gl"', '"renault lecar deluxe"',
'"vokswagen rabbit"', '"datsun 280-zx"', '"mazda rx-7 gs"',
'"triumph tr7 coupe"', '"ford mustang cobra"', '"honda accord"',
'"plymouth reliant"', '"dodge aries wagon (sw)"',
'"toyota starlet"', '"plymouth champ"', '"honda civic 1300"',
'"datsun 210 mpg"', '"toyota tercel"', '"mazda glc 4"',
'"plymouth horizon 4"', '"ford escort 4w"', '"ford escort 2h"',
'"volkswagen jetta"', '"renault 18i"', '"honda prelude"',
'"datsun 200sx"', '"peugeot 505s turbo diesel"', '"volvo diesel"',
'"toyota cressida"', '"datsun 810 maxima"',
'"oldsmobile cutlass ls"', '"ford granada gl"',
'"chrysler lebaron salon"', '"chevrolet cavalier"',
'"chevrolet cavalier wagon"', '"chevrolet cavalier 2-door"',
'"pontiac j2000 se hatchback"', '"dodge aries se"',
'"ford fairmont futura"', '"amc concord dl"',
'"volkswagen rabbit l"', '"mazda glc custom l"',
'"mazda glc custom"', '"plymouth horizon miser"',
'"mercury lynx l"', '"nissan stanza xe"', '"honda civic (auto)"',
'"datsun 310 gx"', '"buick century limited"',
```

```
       '"oldsmobile cutlass ciera (diesel)',
       '"chrysler lebaron medallion"', '"ford granada l"',
       '"toyota celica gt"', '"dodge charger 2.2"', '"chevrolet camaro"',
       '"ford mustang gl"', '"vw pickup"', '"dodge rampage"',
       '"ford ranger"', '"chevy s-10"'], dtype=object)
```

In [60]:
```
data['carname']=[i[0]for i in data['carname'].str.split(' ')]
```

In [61]:
```
data['carname'].unique()
```

Out[61]:
```
array(['"buick', '"plymouth', '"amc', '"ford', '"chevrolet', '"pontiac',
       '"dodge', '"toyota', '"datsun', '"volkswagen', '"peugeot', '"audi',
       '"saab', '"bmw', '"chevy', '"hi', '"mercury', '"opel', '"fiat',
       '"oldsmobile', '"chrysler', '"mazda', '"volvo', '"renault',
       '"toyouta', '"maxda', '"honda', '"subaru"', '"chevroelt', '"capri',
       '"vw', '"mercedes-benz', '"cadillac', '"subaru', '"mercedes',
       '"vokswagen', '"triumph', '"nissan'], dtype=object)
```

In [62]:
```
data['carname']=data['carname'].replace(['"chevrolet','"chevy','"chevroelt'],'chevrole
data['carname']=data['carname'].replace(['"volkswagen','"vokswagen','"vw'],'volkswager
data['carname']=data['carname'].replace('"maxda','mazda')
data['carname']=data['carname'].replace('"toyouta','toyota')
data['carname']=data['carname'].replace('"mercedes','mercedes-benz')
data['carname']=data['carname'].replace('"nissan','datsun')
data['carname']=data['carname'].replace('"capri','ford')
```

In [64]:
```
data['carname'].unique()
```

Out[64]:
```
array(['"buick', '"plymouth', '"amc', '"ford', 'chevrolet', '"pontiac',
       '"dodge', '"toyota', '"datsun', 'volkswagen', '"peugeot', '"audi',
       '"saab', '"bmw', '"hi', '"mercury', '"opel', '"fiat',
       '"oldsmobile', '"chrysler', '"mazda', '"volvo', '"renault',
       'toyota', 'mazda', '"honda', '"subaru"', 'ford', '"mercedes-benz',
       '"cadillac', '"subaru', 'mercedes-benz', '"triumph', 'datsun'],
      dtype=object)
```

In [65]:
```
org=pd.get_dummies(data.origin,prefix='org')
org
```

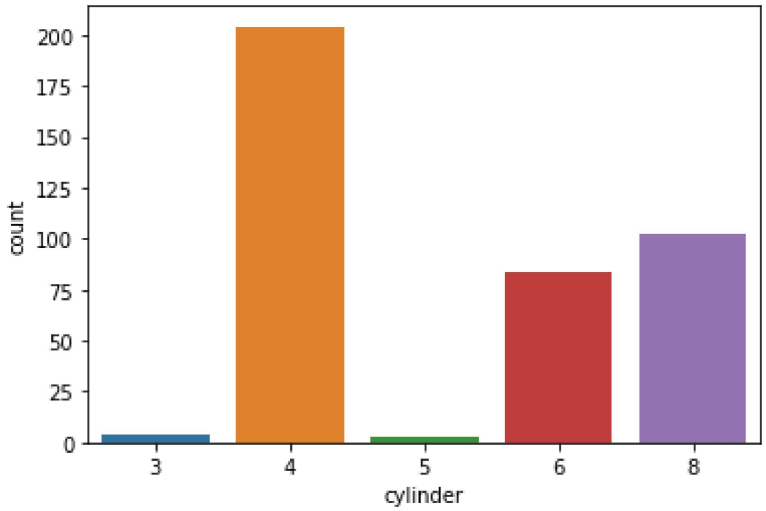
Out[65]:

	org_1	org_2	org_3
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
...	...	...	...
392	1	0	0
393	0	1	0
394	1	0	0
395	1	0	0
396	1	0	0

397 rows × 3 columns

In [66]:
```
sns.countplot(x='cylinder',data=data)
```

Out[66]:
```
<AxesSubplot:xlabel='cylinder', ylabel='count'>
```



In [67]:
```
cyl=pd.get_dummies(data.cylinder,prefix='cyl')
cyl
```

Out[67]:

	cyl_3	cyl_4	cyl_5	cyl_6	cyl_8
0	0	0	0	0	1
1	0	0	0	0	1
2	0	0	0	0	1
3	0	0	0	0	1
4	0	0	0	0	1
...	...	...	...	...	...
392	0	1	0	0	0
393	0	1	0	0	0
394	0	1	0	0	0
395	0	1	0	0	0
396	0	1	0	0	0

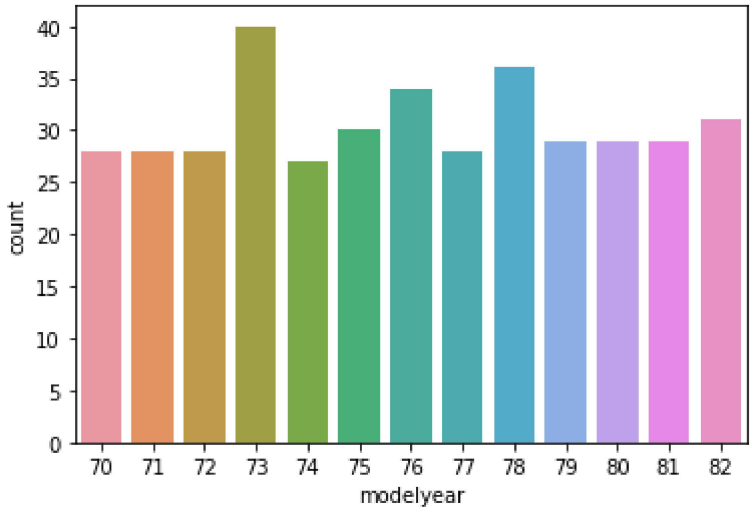
397 rows × 5 columns

In [68]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 397 entries, 0 to 396
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           397 non-null    float64
 1   cylinder      397 non-null    int64  
 2   displacement  397 non-null    float64
 3   horsepower    397 non-null    object 
 4   weight        397 non-null    float64
 5   acceleration  397 non-null    float64
 6   modelyear     397 non-null    int64  
 7   origin        397 non-null    int64  
 8   carname       397 non-null    object 
dtypes: float64(4), int64(3), object(2)
memory usage: 28.0+ KB
```

In [69]:
```
sns.countplot(x='modelyear',data=data)
```

Out[69]:
```
<AxesSubplot:xlabel='modelyear', ylabel='count'>
```



In [70]:
```
year=pd.get_dummies(data.modelyear,prefix='year')
year
```

Out[70]:

	year_70	year_71	year_72	year_73	year_74	year_75	year_76	year_77	year_78	year_79	year_
0	1	0	0	0	0	0	0	0	0	0	
1	1	0	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	
3	1	0	0	0	0	0	0	0	0	0	
4	1	0	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	
392	0	0	0	0	0	0	0	0	0	0	
393	0	0	0	0	0	0	0	0	0	0	
394	0	0	0	0	0	0	0	0	0	0	
395	0	0	0	0	0	0	0	0	0	0	
396	0	0	0	0	0	0	0	0	0	0	

397 rows × 13 columns

In [71]:
```
cn=pd.get_dummies(data['carname'],prefix='cn')
cn
```

Out[71]:

	cn_"amc	cn_"audi	cn_"bmw	cn_"buick	cn_"cadillac	cn_"chrysler	cn_"datsun	cn_"dodge	cn_
0	0	0	0	1	0	0	0	0	
1	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	
392	0	0	0	0	0	0	0	0	
393	0	0	0	0	0	0	0	0	
394	0	0	0	0	0	0	0	1	
395	0	0	0	0	0	0	0	0	
396	0	0	0	0	0	0	0	0	

397 rows × 34 columns

In [72]:
```
data
```

Out[72]:

	mpg	cylinder	displacement	horsepower	weight	acceleration	modelyear	origin	carname
0	15.0	8	350.0	165.0	3693.0	11.5	70	1	"buick
1	18.0	8	318.0	150.0	3436.0	11.0	70	1	"plymouth
2	16.0	8	304.0	150.0	3433.0	12.0	70	1	"amc
3	17.0	8	302.0	140.0	3449.0	10.5	70	1	"ford
4	15.0	8	429.0	198.0	4341.0	10.0	70	1	"ford
...	...	...	...	...	...	...	...	...	...
392	27.0	4	140.0	86.00	2790.0	15.6	82	1	"ford
393	44.0	4	97.0	52.00	2130.0	24.6	82	2	volkswagen
394	32.0	4	135.0	84.00	2295.0	11.6	82	1	"dodge
395	28.0	4	120.0	79.00	2625.0	18.6	82	1	"ford
396	31.0	4	119.0	82.00	2720.0	19.4	82	1	chevrolet

397 rows × 9 columns

In [77]:
```
data.drop(['origin','cylinder','modelyear','carname'],axis=1,inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [77], in <cell line: 1>()
----> 1 data.drop(['origin','cylinder','modelyear','carname'],axis=1,inplace=True)
      2 data

File D:\anaconda\lib\site-packages\pandas\util\_decorators.py:311, in deprecate_nonke
yword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    305 if len(args) > num_allow_args:
    306     warnings.warn(
    307         msg.format(arguments=arguments),
    308         FutureWarning,
    309         stacklevel=stacklevel,
    310     )
--> 311 return func(*args, **kwargs)

File D:\anaconda\lib\site-packages\pandas\core\frame.py:4954, in DataFrame.drop(self,
labels, axis, index, columns, level, inplace, errors)
   4806 @deprecate_nonkeyword_arguments(version=None, allowed_args=["self", "labels"
])
   4807 def drop(
   4808     self,
   (...)
   4815     errors: str = "raise",
   4816 ):
   4817     """
   4818     Drop specified labels from rows or columns.
   4819
   (...)
   4952             weight  1.0     0.8
   4953     """
-> 4954     return super().drop(
   4955         labels=labels,
   4956         axis=axis,
   4957         index=index,
   4958         columns=columns,
   4959         level=level,
   4960         inplace=inplace,
   4961         errors=errors,
   4962     )

File D:\anaconda\lib\site-packages\pandas\core\generic.py:4267, in NDFrame.drop(self,
labels, axis, index, columns, level, inplace, errors)
   4265 for axis, labels in axes.items():
   4266     if labels is not None:
-> 4267         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4269 if inplace:
   4270     self._update_inplace(obj)

File D:\anaconda\lib\site-packages\pandas\core\generic.py:4311, in NDFrame._drop_axis
(self, labels, axis, level, errors, consolidate, only_slice)
   4309         new_axis = axis.drop(labels, level=level, errors=errors)
   4310     else:
-> 4311         new_axis = axis.drop(labels, errors=errors)
   4312     indexer = axis.get_indexer(new_axis)
   4314 # Case for non-unique axis
   4315 else:

File D:\anaconda\lib\site-packages\pandas\core\indexes\base.py:6644, in Index.drop(se
lf, labels, errors)
```

```
   6642 if mask.any():
   6643     if errors != "ignore":
-> 6644         raise KeyError(f"{list(labels[mask])} not found in axis")
   6645     indexer = indexer[~mask]
   6646 return self.delete(indexer)

KeyError: "['origin', 'cylinder', 'modelyear', 'carname'] not found in axis"
```

In [78]:
```
data
```

Out[78]:

	mpg	displacement	horsepower	weight	acceleration	cn_"amc	cn_"audi	cn_"bmw	cn_"buick
0	15.0	350.0	165.0	3693.0	11.5	0	0	0	1
1	18.0	318.0	150.0	3436.0	11.0	0	0	0	0
2	16.0	304.0	150.0	3433.0	12.0	1	0	0	0
3	17.0	302.0	140.0	3449.0	10.5	0	0	0	0
4	15.0	429.0	198.0	4341.0	10.0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
392	27.0	140.0	86.00	2790.0	15.6	0	0	0	0
393	44.0	97.0	52.00	2130.0	24.6	0	0	0	0
394	32.0	135.0	84.00	2295.0	11.6	0	0	0	0
395	28.0	120.0	79.00	2625.0	18.6	0	0	0	0
396	31.0	119.0	82.00	2720.0	19.4	0	0	0	0

397 rows × 60 columns

In [74]:
```
data=pd.concat([data,cn,year,cyl,org],axis=1)
data
```

Out[74]:

	mpg	displacement	horsepower	weight	acceleration	cn_"amc	cn_"audi	cn_"bmw	cn_"buick
0	15.0	350.0	165.0	3693.0	11.5	0	0	0	1
1	18.0	318.0	150.0	3436.0	11.0	0	0	0	0
2	16.0	304.0	150.0	3433.0	12.0	1	0	0	0
3	17.0	302.0	140.0	3449.0	10.5	0	0	0	0
4	15.0	429.0	198.0	4341.0	10.0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
392	27.0	140.0	86.00	2790.0	15.6	0	0	0	0
393	44.0	97.0	52.00	2130.0	24.6	0	0	0	0
394	32.0	135.0	84.00	2295.0	11.6	0	0	0	0
395	28.0	120.0	79.00	2625.0	18.6	0	0	0	0
396	31.0	119.0	82.00	2720.0	19.4	0	0	0	0

397 rows × 60 columns

In [79]:
```
data.shape
```

Out[79]:
```
(397, 60)
```

In [84]:
```
data[['displacement','horsepower','weight','acceleration']]=StandardScaler().fit_trans
```

```
C:\Users\studentadmin\AppData\Local\Temp\ipykernel_6952\2376225795.py:1: SettingWithC
opyWarning:
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/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  data[['displacement','horsepower','weight','acceleration']]=StandardScaler().fit_tr
ansform(data[['displacement','horsepower','weight','acceleration']])
```

In [81]:
```
sum(data.horsepower=='?')
```

Out[81]:
```
6
```

In [82]:
```
data=data[data.horsepower!='?']
```

In [83]:
```
data
```

Out[83]:

	mpg	displacement	horsepower	weight	acceleration	cn_"amc	cn_"audi	cn_"bmw	cn_"buick
0	15.0	350.0	165.0	3693.0	11.5	0	0	0	1
1	18.0	318.0	150.0	3436.0	11.0	0	0	0	0
2	16.0	304.0	150.0	3433.0	12.0	1	0	0	0
3	17.0	302.0	140.0	3449.0	10.5	0	0	0	0
4	15.0	429.0	198.0	4341.0	10.0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
392	27.0	140.0	86.00	2790.0	15.6	0	0	0	0
393	44.0	97.0	52.00	2130.0	24.6	0	0	0	0
394	32.0	135.0	84.00	2295.0	11.6	0	0	0	0
395	28.0	120.0	79.00	2625.0	18.6	0	0	0	0
396	31.0	119.0	82.00	2720.0	19.4	0	0	0	0

391 rows × 60 columns

In [85]:
```
y=data.pop('mpg')
```

In [86]:
```
y
```

Out[86]:
```
0      15.0
1      18.0
2      16.0
3      17.0
4      15.0
       ...
392    27.0
393    44.0
394    32.0
395    28.0
396    31.0
Name: mpg, Length: 391, dtype: float64
```

In [87]:
```
x=data
```

In [88]:
```
x.head(269)
```

Out[88]:

	displacement	horsepower	weight	acceleration	cn_"amc	cn_"audi	cn_"bmw	cn_"buick	cn
0	1.491799	1.575170	0.844259	-1.471246	0	0	0	1	
1	1.185545	1.185250	0.541544	-1.652864	0	0	0	0	
2	1.051559	1.185250	0.538010	-1.289628	1	0	0	0	
3	1.032419	0.925303	0.556856	-1.834482	0	0	0	0	
4	2.247862	2.432994	1.607524	-2.016100	0	0	0	0	
...	...	...	...	...	...	...	...	...	
266	-0.575413	-0.244456	-0.490278	-0.490508	0	0	0	0	
267	-0.718969	-0.192467	-0.796526	-0.308890	0	0	0	0	
268	-0.852955	-0.764349	-0.878977	-0.381537	0	0	0	0	
269	-0.575413	-0.244456	-0.543282	-0.272566	0	0	0	0	
270	-0.364863	0.015490	-0.272370	0.417582	0	0	0	0	

269 rows × 59 columns

dividing 80:20

In [89]:
```
trainx,testx,trainy,testy=train_test_split(x,y,test_size=0.2,random_state=69)
```

In [90]:
```
LR=LinearRegression()
LR.fit(trainx,trainy)
```

Out[90]:
```
LinearRegression()
```

In [91]:
```
pred=LR.predict(testx)
mean_squared_error(pred,testy)
```

Out[91]:
```
6.538767175795156
```

In [93]:
```
df=pd.DataFrame({'Actual':testy,'Predicted':pred})
```

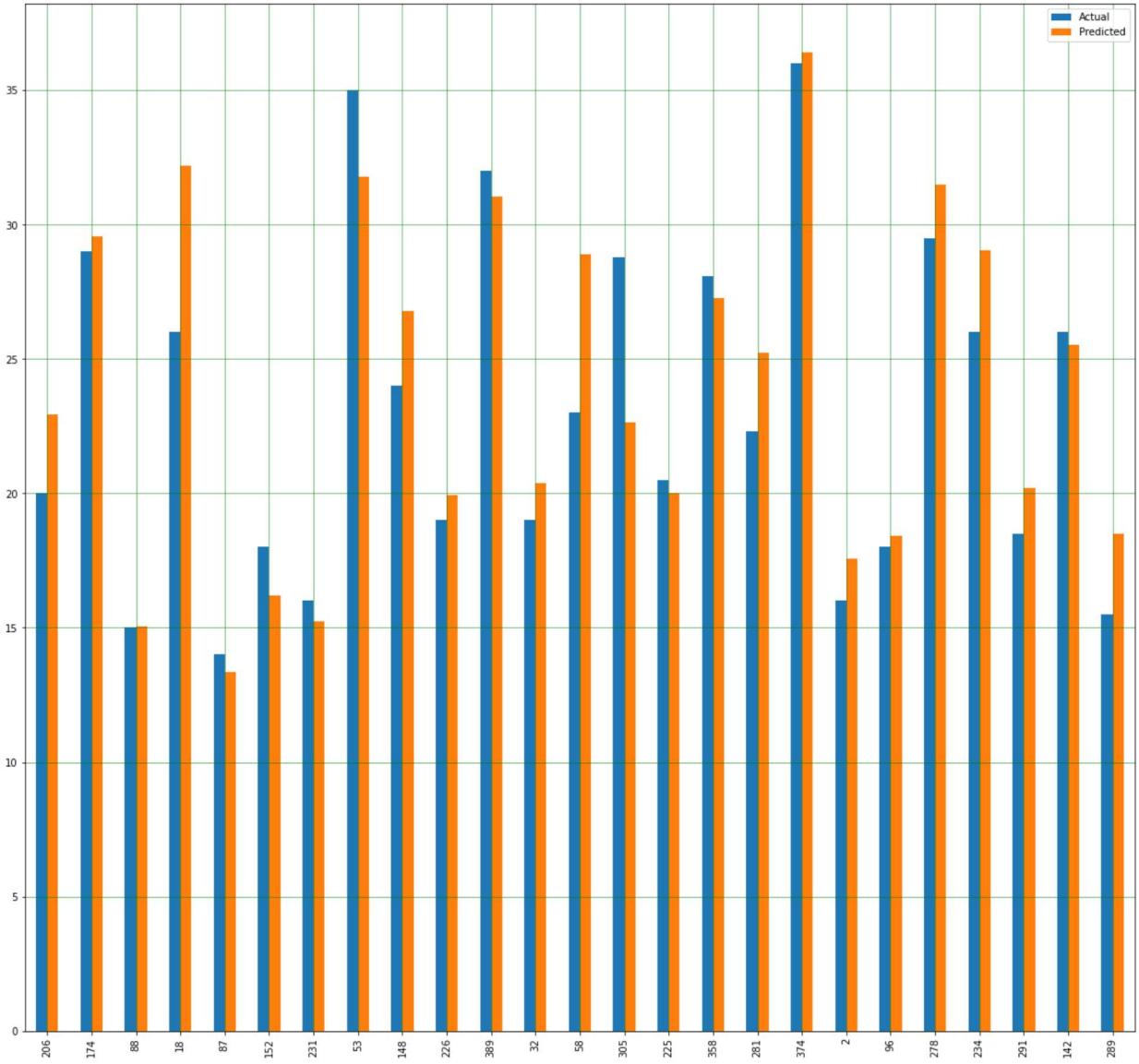
In [94]:
```
df
```

Out[94]:

	Actual	Predicted
206	20.0	22.947266
174	29.0	29.570312
88	15.0	15.038086
18	26.0	32.182617
87	14.0	13.333008
...	...	...
105	12.0	12.300781
211	16.5	14.423828
61	13.0	12.058594
199	18.0	17.632812
21	25.0	27.762695

79 rows × 2 columns

In [95]:
```
df1=df.head(25)
df1.plot(y=['Actual','Predicted'],kind='bar',figsize=(20,19));
plt.grid(which='major',ls='-',linewidth=0.5,color='g')
plt.grid(which='minor',ls=':',linewidth=0.5,color='b')
plt.show()
```



In [96]:
```
print('Mean Absolute Error',mean_absolute_error(testy,pred))
print('Mean Squared Error',mean_squared_error(testy,pred))
print('Root Mean Sqaured Error',np.sqrt(mean_squared_error(testy,pred)))
```

```
Mean Absolute Error 2.012801621835443
Mean Squared Error 6.538767175795156
Root Mean Sqaured Error 2.5571013229426707
```

divding 70:30

In [98]:
```
trainx,testx,trainy,testy=train_test_split(x,y,test_size=0.3,random_state=69)
```

In [99]:
```
LR=LinearRegression()
LR.fit(trainx,trainy)
```

Out[99]:
```
LinearRegression()
```

In [100…
```
pred=LR.predict(testx)
mean_squared_error(pred,testy)
```

Out[100]:
```
3.0270984824050786e+25
```

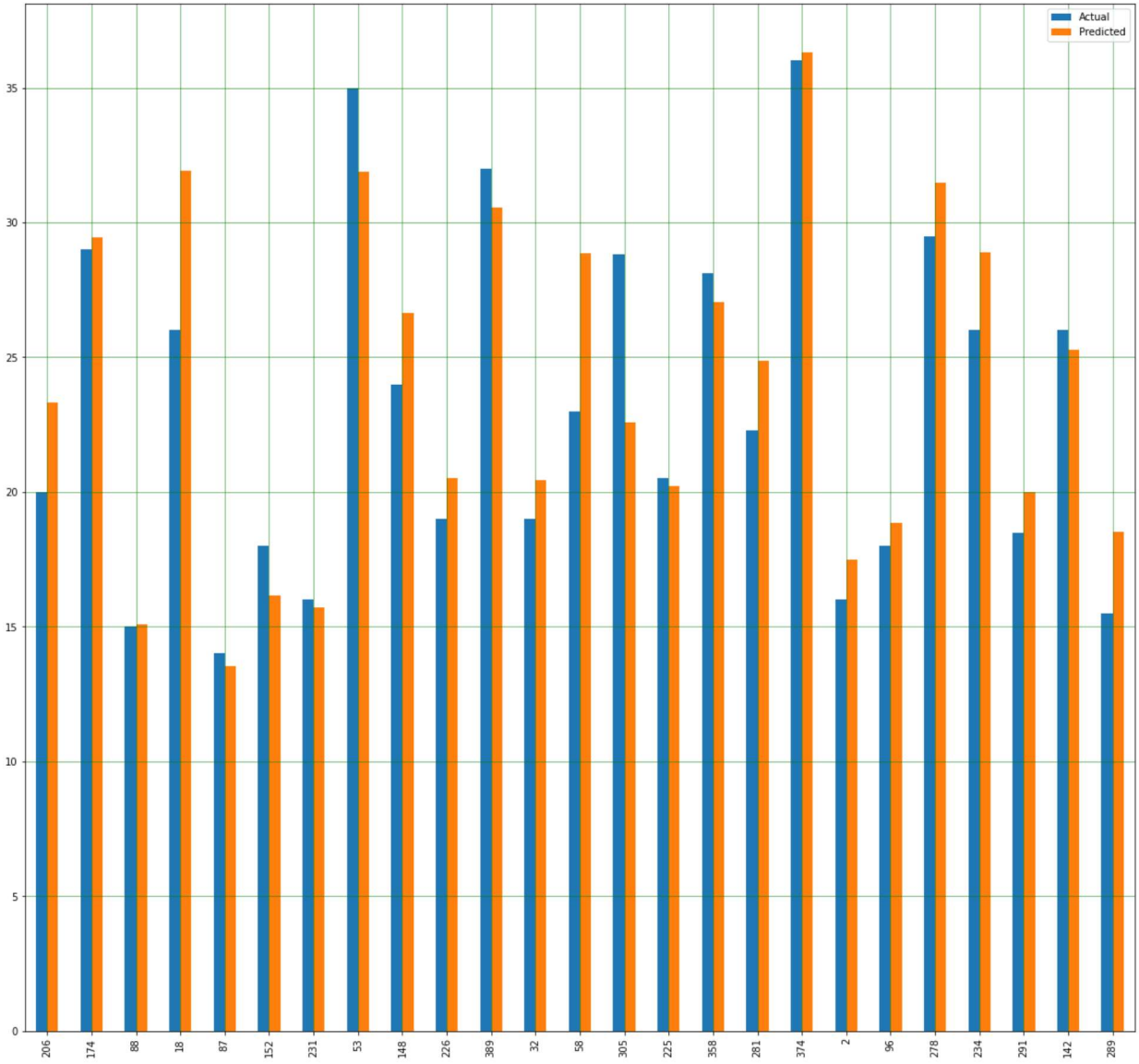
In [101…
```
df=pd.DataFrame({'Actual':testy,'Predicted':pred})
df
```

Out[101]:

	Actual	Predicted
206	20.0	23.314453
174	29.0	29.451172
88	15.0	15.099609
18	26.0	31.904297
87	14.0	13.521484
…	…	…
136	13.0	13.380859
380	36.0	35.531250
240	22.0	25.265625
64	14.0	13.130859
77	21.0	21.888672

118 rows × 2 columns

In [102…
```
df1=df.head(25)
df1.plot(y=['Actual','Predicted'],kind='bar',figsize=(20,19));
plt.grid(which='major',ls='-',linewidth=0.5,color='g')
plt.grid(which='minor',ls=':',linewidth=0.5,color='b')
plt.show()
```



In [103…
```
print('Mean Absolute Error',mean_absolute_error(testy,pred))
print('Mean Squared Error',mean_squared_error(testy,pred))
print('Root Mean Sqaured Error',np.sqrt(mean_squared_error(testy,pred)))
```

```
Mean Absolute Error 1054299011303.7745
Mean Squared Error 3.0270984824050786e+25
Root Mean Sqaured Error 5501907380540.933
```

dividing 90:10

In [104…
```
trainx,testx,trainy,testy=train_test_split(x,y,test_size=0.1,random_state=69)
```

In [105…
```
LR=LinearRegression()
LR.fit(trainx,trainy)
```

Out[105]:
```
LinearRegression()
```

In [106…
```
pred=LR.predict(testx)
mean_squared_error(pred,testy)
```

Out[106]:
```
6.1197006835937495
```

In [107…
```
df=pd.DataFrame({'Actual':testy,'Predicted':pred})
df
```

Out[107]:

	Actual	Predicted
206	20.0	23.484375
174	29.0	29.437500
88	15.0	14.718750
18	26.0	31.078125
87	14.0	13.250000
152	18.0	16.703125
231	16.0	15.218750
53	35.0	31.734375
148	24.0	27.156250
226	19.0	19.796875
389	32.0	31.125000
32	19.0	20.515625
58	23.0	27.968750
305	28.8	23.203125
225	20.5	19.468750
358	28.1	27.781250
281	22.3	24.687500
374	36.0	36.046875
2	16.0	16.812500
96	18.0	18.562500
278	29.5	31.640625
234	26.0	28.617188
291	18.5	18.125000
142	26.0	25.843750
289	15.5	18.562500
119	19.0	22.640625
149	26.0	25.359375
218	25.5	27.781250
11	15.0	17.132812
75	18.0	23.140625
268	30.9	27.593750
10	14.0	18.601562
3	17.0	17.406250

	Actual	Predicted
17	27.0	28.234375
25	10.0	10.273438
304	28.4	27.171875
188	15.5	16.015625
15	18.0	19.125000
5	14.0	11.203125
252	20.5	20.453125

In [108…
```
print('Mean Absolute Error',mean_absolute_error(testy,pred))
print('Mean Squared Error',mean_squared_error(testy,pred))
print('Root Mean Sqaured Error',np.sqrt(mean_squared_error(testy,pred)))
```

```
Mean Absolute Error 1.8791406249999998
Mean Squared Error 6.1197006835937495
Root Mean Sqaured Error 2.473802878887837
```

80:20 ::: Mean Absolute Error 2.012801621835443 Mean Squared Error 6.538767175795156 Root Mean Sqaured Error 2.5571013229426707

90:10 :: Mean Absolute Error 1.8791406249999998 Mean Squared Error 6.1197006835937495 Root Mean Sqaured Error 2.473802878887837

70:30 Mean Absolute Error 1054299011303.7745 Mean Squared Error 3.0270984824050786e+25 Root Mean Sqaured Error 5501907380540.933

most least error is in 90 :10 split

In [ ]: