DA2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

Name:- Adrija Mukhopadhyay

Reg No:- 19BDS0159

```
import pandas as pd
import matplotlib.pylab as mp
import seaborn as sb
import numpy as np
```

```
filename = "https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DA0101EN/auto.csv"
```

```
[4] headers = ["symboling","normalized-losses","make","fuel-type","aspiration", "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"]
```

```
[5] df = pd.read_csv(filename, names = headers)
```

```
[6] df.head()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepo
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	

5 rows × 26 columns

```
[7] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          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
```

```
[8] df.describe()
```

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count	205.000000	205.000000	205.000000	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	10.142537	25.219512	30.751220
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.542142	6.886443
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000

Identify missing values

Convert "?" to NaN

```
[10] # replace "?" to NaN
     df.replace("Nan", np.nan, inplace = True)
     df.head(5)
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepo
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	

5 rows × 26 columns

```
[11] missing_data = df.isnull()
     missing_data.head(5)
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False

5 rows × 26 columns

"True" stands for missing value, while "False" stands for not missing value.

Count missing values in each column

```
[12] for column in missing_data.columns.values.tolist():
         print(column)
         print (missing_data[column].value_counts())
         print("")
```

```
symboling
False    205
Name: symboling, dtype: int64

normalized-losses
False    205
Name: normalized-losses, dtype: int64

make
False    205
Name: make, dtype: int64

fuel-type
False    205
Name: fuel-type, dtype: int64

aspiration
```

```
False    205
Name: aspiration, dtype: int64

num-of-doors
False    205
Name: num-of-doors, dtype: int64

body-style
False    205
Name: body-style, dtype: int64

drive-wheels
False    205
Name: drive-wheels, dtype: int64

engine-location
False    205
Name: engine-location, dtype: int64

wheel-base
False    205
Name: wheel-base, dtype: int64

length
False    205
Name: length, dtype: int64

width
False    205
Name: width, dtype: int64

height
False    205
Name: height, dtype: int64

curb-weight
False    205
Name: curb-weight, dtype: int64

engine-type
False    205
```

Based on the summary above, each column has 205 rows of data, seven columns containing missing data:

"normalized-losses": 41 missing data

"num-of-doors": 2 missing data

"bore": 4 missing data

"stroke" : 4 missing data

"horsepower": 2 missing data

"peak-rpm": 2 missing data

"price": 4 missing data

Deal with missing data

```
[14] df['num-of-doors'].value_counts()
```

```
four    114
two      89
?         2
Name: num-of-doors, dtype: int64
```

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

```
four
```

```
[16] #replace the missing 'num-of-doors' values by the most frequent
     df["num-of-doors"].replace(np.nan, "four", inplace=True)
```

```
[17] # simply drop whole row with NaN in "price" column
     df.dropna(subset=["price"], axis=0, inplace=True)

     # reset index, because we droped two rows
     df.reset_index(drop=True, inplace=True)
```

```
[19] df.isna().sum()
```

```
symboling           0
normalized-losses    0
make                0
fuel-type           0
aspiration          0
num-of-doors        0
body-style          0
drive-wheels        0
```

```
engine-location      0
wheel-base           0
length               0
width                0
height               0
curb-weight          0
engine-type          0
num-of-cylinders     0
engine-size          0
fuel-system          0
bore                 0
stroke               0
compression-ratio    0
horsepower           0
peak-rpm             0
city-mpg             0
highway-mpg          0
price                0
dtype: int64
```

```
[21] df.dtypes
```

```
symboling              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
```

Data Standardization

```
[24] # Convert mpg to L/100km by mathematical operation (235 divided by mpg)
     df['city-L/100km'] = 235/df["city-mpg"]

     # check your transformed data
     df.head()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rp
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	mpfi	3.47	2.68	9.0	111	500
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	mpfi	3.47	2.68	9.0	111	500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	mpfi	2.68	3.47	9.0	154	500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	mpfi	3.19	3.40	10.0	102	550
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	mpfi	3.19	3.40	8.0	115	550

5 rows × 27 columns

Data Normalization

```
[26] # replace (original value) by (original value)/(maximum value)
     df['length'] = df['length']/df['length'].max()
     df['width'] = df['width']/df['width'].max()
```

```
[27] df.corr()
```

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg	city-L/100km
symboling	1.000000	-0.531954	-0.357612	-0.232919	-0.541038	-0.227691	-0.105790	-0.178515	-0.035823	0.034606	0.063165

wheel-base	-0.531954	1.000000	0.874587	0.795144	0.589435	0.776386	0.569329	0.249786	-0.470414	-0.544082	0.474040
length	-0.357612	0.874587	1.000000	0.841118	0.491029	0.877728	0.683360	0.158414	-0.670909	-0.704662	0.659165
width	-0.232919	0.795144	0.841118	1.000000	0.279210	0.867032	0.735433	0.181129	-0.642704	-0.677218	0.682850
height	-0.541038	0.589435	0.491029	0.279210	1.000000	0.295572	0.067149	0.261214	-0.048640	-0.107358	-0.002333
curb-weight	-0.227691	0.776386	0.877728	0.867032	0.295572	1.000000	0.850594	0.151362	-0.757414	-0.797465	0.791911
engine-size	-0.105790	0.569329	0.683360	0.735433	0.067149	0.850594	1.000000	0.028971	-0.653658	-0.677470	0.744952
compression-ratio	-0.178515	0.249786	0.158414	0.181129	0.261214	0.151362	0.028971	1.000000	0.324701	0.265201	-0.296964
city-mpg	-0.035823	-0.470414	-0.670909	-0.642704	-0.048640	-0.757414	-0.653658	0.324701	1.000000	0.971337	-0.950493
highway-mpg	0.034606	-0.544082	-0.704662	-0.677218	-0.107358	-0.797465	-0.677470	0.265201	0.971337	1.000000	-0.928767
city-L/100km	0.063165	0.474040	0.659165	0.682850	-0.002333	0.791911	0.744952	-0.296964	-0.950493	-0.928767	1.000000

```
[28] df_sample=df[['bore','stroke','compression-ratio','horsepower']]
     df.corr()
```

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg	city-L/100km
symboling	1.000000	-0.531954	-0.357612	-0.232919	-0.541038	-0.227691	-0.105790	-0.178515	-0.035823	0.034606	0.063165
wheel-base	-0.531954	1.000000	0.874587	0.795144	0.589435	0.776386	0.569329	0.249786	-0.470414	-0.544082	0.474040
length	-0.357612	0.874587	1.000000	0.841118	0.491029	0.877728	0.683360	0.158414	-0.670909	-0.704662	0.659165
width	-0.232919	0.795144	0.841118	1.000000	0.279210	0.867032	0.735433	0.181129	-0.642704	-0.677218	0.682850
height	-0.541038	0.589435	0.491029	0.279210	1.000000	0.295572	0.067149	0.261214	-0.048640	-0.107358	-0.002333
curb-weight	-0.227691	0.776386	0.877728	0.867032	0.295572	1.000000	0.850594	0.151362	-0.757414	-0.797465	0.791911
engine-size	-0.105790	0.569329	0.683360	0.735433	0.067149	0.850594	1.000000	0.028971	-0.653658	-0.677470	0.744952
compression-ratio	-0.178515	0.249786	0.158414	0.181129	0.261214	0.151362	0.028971	1.000000	0.324701	0.265201	-0.296964
city-mpg	-0.035823	-0.470414	-0.670909	-0.642704	-0.048640	-0.757414	-0.653658	0.324701	1.000000	0.971337	-0.950493
highway-mpg	0.034606	-0.544082	-0.704662	-0.677218	-0.107358	-0.797465	-0.677470	0.265201	0.971337	1.000000	-0.928767
city-L/100km	0.063165	0.474040	0.659165	0.682850	-0.002333	0.791911	0.744952	-0.296964	-0.950493	-0.928767	1.000000

```
[38] numerical_cols = df.select_dtypes(include=['int64','float64'])
     print(numerical_cols.columns)
```

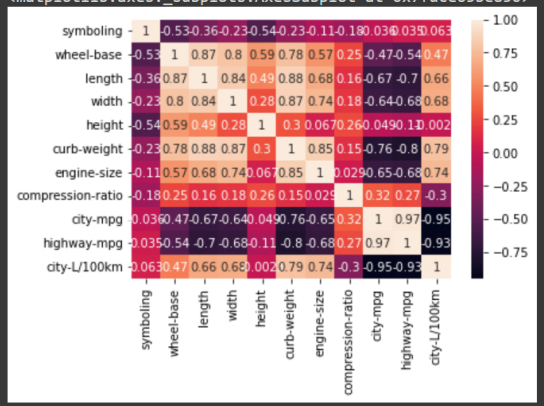
```
Index(['symboling', 'wheel-base', 'length', 'width', 'height', 'curb-weight',
       'engine-size', 'compression-ratio', 'city-mpg', 'highway-mpg',
       'city-L/100km'],
      dtype='object')
```

```
[40] categorical_cols = df.select_dtypes(include=['object'])
     print(categorical_cols.columns)
```

```
Index(['normalized-losses', 'make', 'fuel-type', 'aspiration', 'num-of-doors',
       'body-style', 'drive-wheels', 'engine-location', 'engine-type',
       'num-of-cylinders', 'fuel-system', 'bore', 'stroke', 'horsepower',
       'peak-rpm', 'price'],
      dtype='object')
```

```
[41] sb.heatmap(numerical_cols.corr(), annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdce65be850>
```




```
[46] df.dtypes
```

```
symboling              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
city-L/100km        float64
dtype: object
```

```
[52] df["normalized-losses"] = pd.to_numeric(df["normalized-losses"],errors='coerce')
     df["price"] = pd.to_numeric(df["price"],errors='coerce')
     df["peak-rpm"] = pd.to_numeric(df["peak-rpm"],errors='coerce')
```

```
[53] df.dtypes
```

```
symboling             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                 object
stroke               object
compression-ratio   float64
horsepower           object
peak-rpm            float64
city-mpg              int64
highway-mpg           int64
price               float64
city-L/100km        float64
dtype: object
```

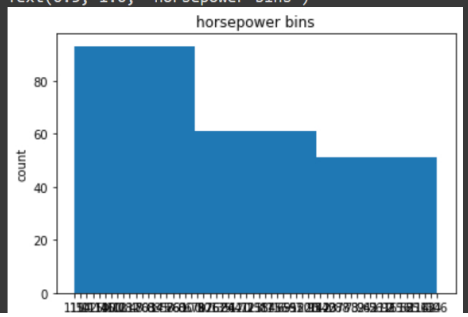
```
[54] %matplotlib inline
     import matplotlib as plt
     from matplotlib import pyplot

     a = (0,1,2)

     # draw historgram of attribute "horsepower" with bins = 3
     plt.pyplot.hist(df["horsepower"], bins = 3)

     # set x/y labels and plot title
     plt.pyplot.xlabel("horsepower")
     plt.pyplot.ylabel("count")
     plt.pyplot.title("horsepower bins")
```

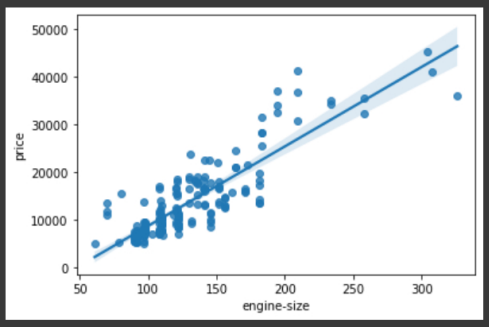
```
Text(0.5, 1.0, 'horsepower bins')
```



horsepower

[58]
```
# Engine size as potential predictor variable of price
import matplotlib.pyplot as plt
sns.regplot(x="engine-size", y="price", data=df)

plt.show()
```



[59]
```
df[["engine-size", "price"]].corr()
```

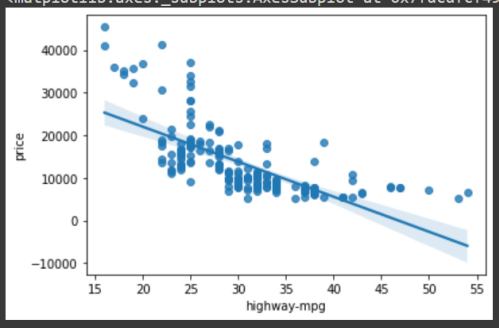
	engine-size	price
engine-size	1.000000	0.872335
price	0.872335	1.000000

As the engine-size goes up, the price goes up: this indicates a positive direct correlation between these two variables. Engine size seems like a pretty good predictor of price since the regression line is almost a perfect diagonal line.

We can examine the correlation between 'engine-size' and 'price' and see it's approximately 0.87

[60]
```
sns.regplot(x="highway-mpg", y="price", data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdcdfcf4990>
```



[61]
```
df[['highway-mpg', 'price']].corr()
```

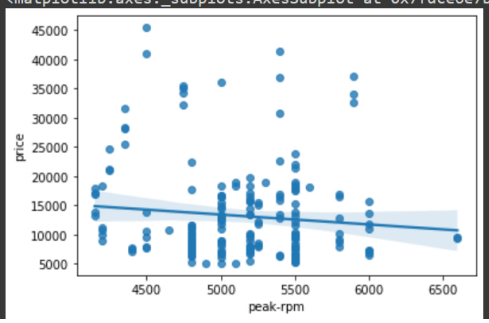
	highway-mpg	price
highway-mpg	1.000000	-0.704692
price	-0.704692	1.000000

As the highway-mpg goes up, the price goes down: this indicates an inverse/negative relationship between these two variables. Highway mpg could potentially be a predictor of price.

We can examine the correlation between 'highway-mpg' and 'price' and see it's approximately -0.704

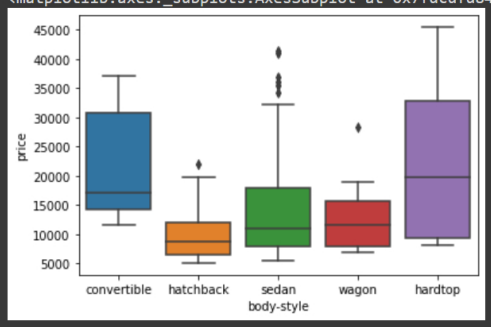
[62]
```
sns.regplot(x="peak-rpm", y="price", data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdce6e7bf10>
```



```
[63] sns.boxplot(x="body-style", y="price", data=df)
```

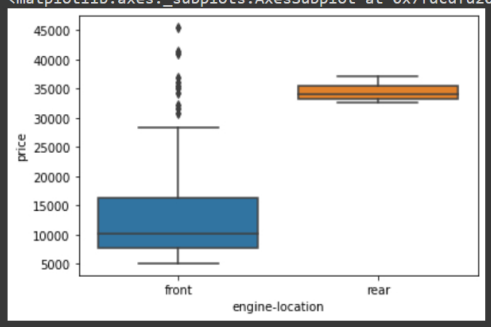
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdcdfd84ed0>
```



We see that the distributions of price between the different body-style categories have a significant overlap, and so body-style would not be a good predictor of price. Let's examine engine "engine-location" and "price":

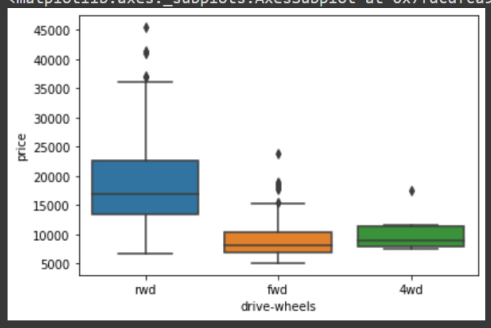
```
[64] sns.boxplot(x="engine-location", y="price", data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdcdfd20ad0>
```




```
[65] # drive-wheels
sns.boxplot(x="drive-wheels", y="price", data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdcdfca9590>
```




```
[66] df.describe()
```

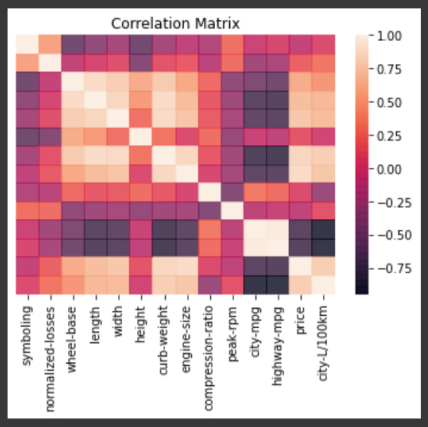
	symboling	normalized-losses	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	peak-rpm	city-mpg	highway-mpg	
count	205.000000	164.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	203.000000	205.000000	205.000000	
mean	0.834146	122.000000	98.756585	0.836373	0.911588	53.724878	2555.565854	126.907317	10.142537	5125.369458	25.219512	30.751220	1
std	1.245307	35.442168	6.021776	0.059285	0.029671	2.443522	520.680204	41.642693	3.972040	479.334560	6.542142	6.886443	
min	-2.000000	65.000000	86.600000	0.678039	0.834025	47.800000	1488.000000	61.000000	7.000000	4150.000000	13.000000	16.000000	
25%	0.000000	94.000000	94.500000	0.799135	0.886584	52.000000	2145.000000	97.000000	8.600000	4800.000000	19.000000	25.000000	
50%	1.000000	115.000000	97.000000	0.832292	0.905947	54.100000	2414.000000	120.000000	9.000000	5200.000000	24.000000	30.000000	1
75%	2.000000	150.000000	102.400000	0.879865	0.925311	55.500000	2935.000000	141.000000	9.400000	5500.000000	30.000000	34.000000	1
max	3.000000	256.000000	120.900000	1.000000	1.000000	59.800000	4066.000000	326.000000	23.000000	6600.000000	49.000000	54.000000	4

```
[67] df.describe(include=['object'])
```

	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	engine-type	num-of-cylinders	fuel-system	bore	stroke	horsepower
count	205	205	205	205	205	205	205	205	205	205	205	205	205

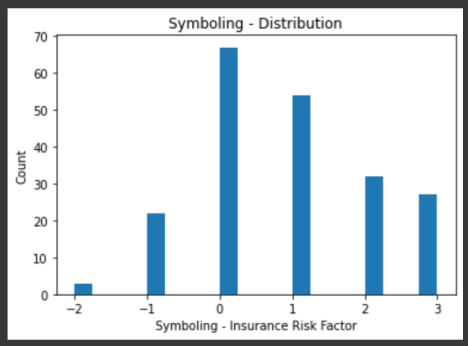
unique	22	2	2	3	5	3	2	7	7	8	39	37	60
top	toyota	gas	std	four	sedan	fwd	front	ohc	four	mpfi	3.62	3.40	68
freq	32	185	168	114	96	120	202	148	159	94	23	20	19

[68]
```
sns.heatmap(df.corr(), yticklabels=False,cmap = 'rocket', alpha = 0.8)
plt.title("Correlation Matrix")
plt.show()
```

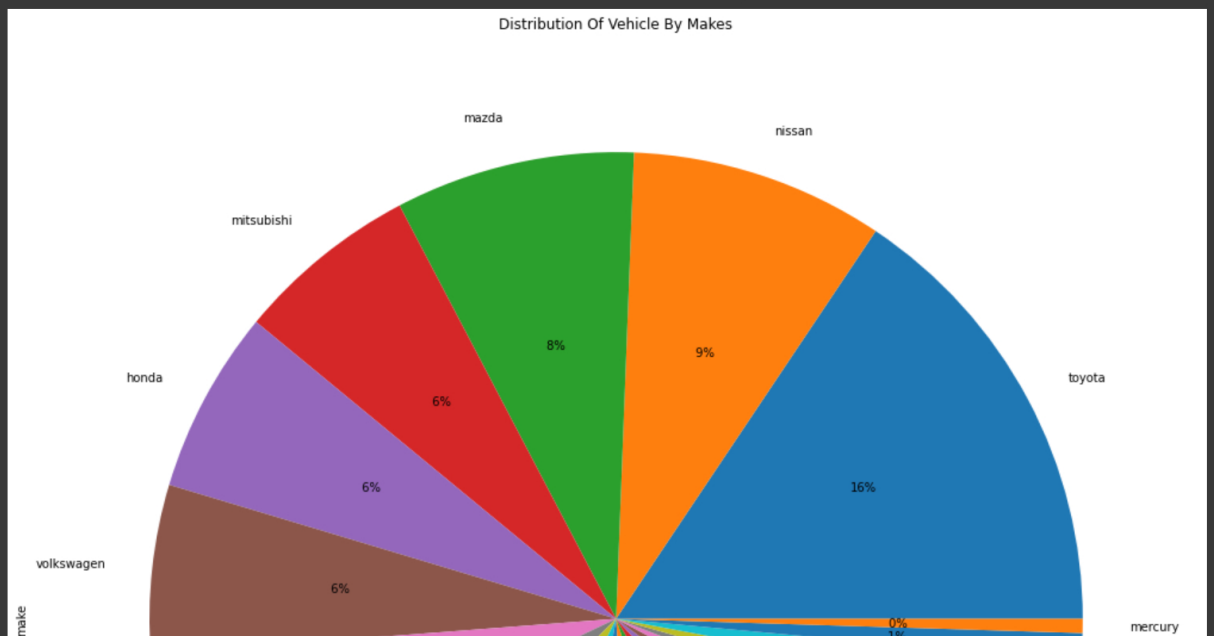


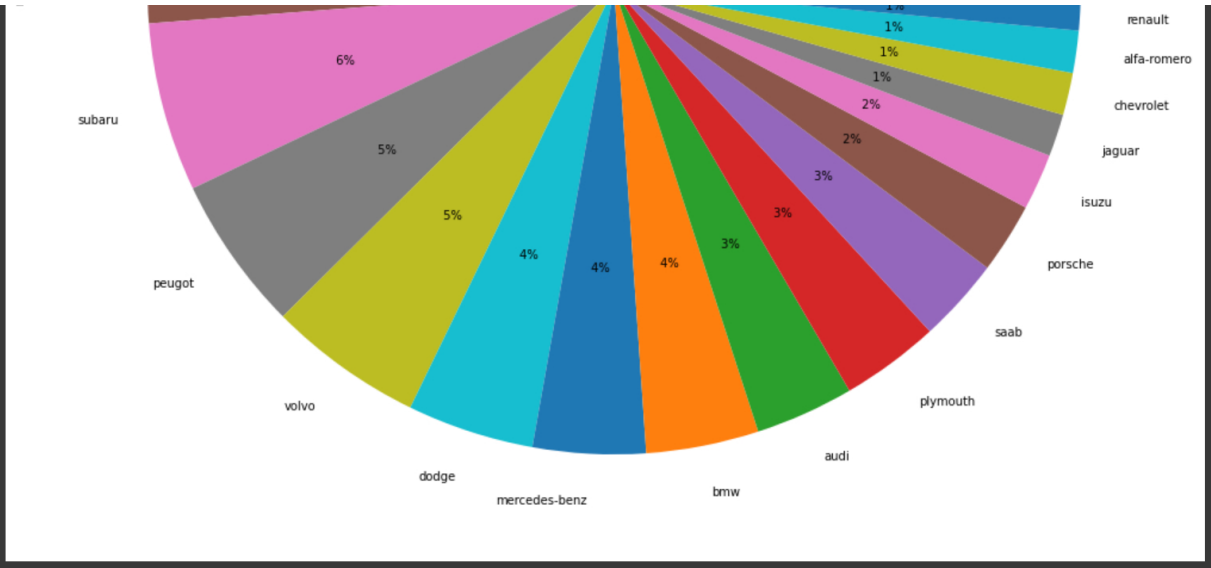
[75]
```
#Plotting a Histogram for the symboling column
df['symboling'].plot(kind='hist',bins=20)
#automobile['symboling'].value_counts().plot(kind='bar',align='center',rot=1)
plt.title('Symboling - Distribution')
plt.ylabel('Count')
plt.xlabel('Symboling - Insurance Risk Factor')

plt.show()
```

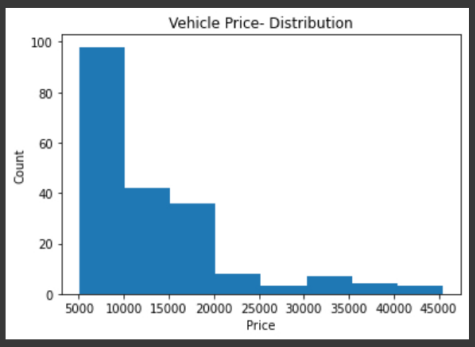


[76]
```
#Plotting a pie chart for the make attribute
df['make'].value_counts().plot(kind='pie',autopct='%2.0f%%',figsize=(18,18))
plt.title('Distribution Of Vehicle By Makes')
plt.show()
```

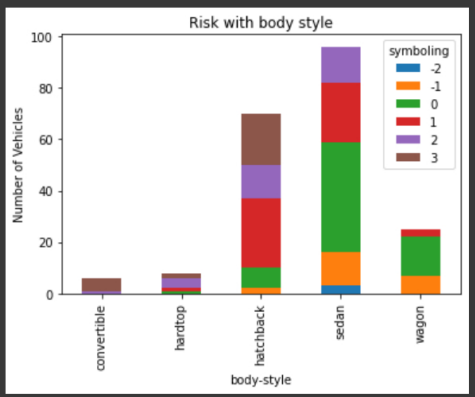




[77]
```
#plotting a histogram for the price of vehicles from the dataset
df['price'].plot(kind='hist',bins=8)
plt.title('Vehicle Price- Distribution')
plt.xlabel('Price')
plt.ylabel('Count')
plt.show()
```



[78]
```
#plotting a stacked bar chart for the body-style attribute and symboling attribute
df.groupby(['body-style','symboling']).size().unstack().plot(kind='bar',stacked=True)
plt.ylabel('Number of Vehicles')
plt.title("Risk with body style")
plt.show()
```



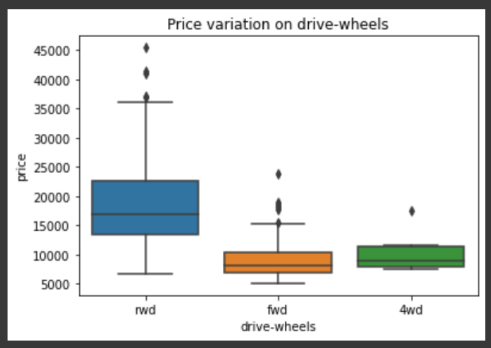
[79]
```
df[df['body-style']=='sedan']
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm
3	2	164.0	audi	gas	std	four	sedan	fwd	front	99.8	...	mpfi	3.19	3.40	10.0	102	5500.0
4	2	164.0	audi	gas	std	four	sedan	4wd	front	99.4	...	mpfi	3.19	3.40	8.0	115	5500.0
5	2	NaN	audi	gas	std	two	sedan	fwd	front	99.8	...	mpfi	3.19	3.40	8.5	110	5500.0
6	1	158.0	audi	gas	std	four	sedan	fwd	front	105.8	...	mpfi	3.19	3.40	8.5	110	5500.0
8	1	158.0	audi	gas	turbo	four	sedan	fwd	front	105.8	...	mpfi	3.13	3.40	8.3	140	5500.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

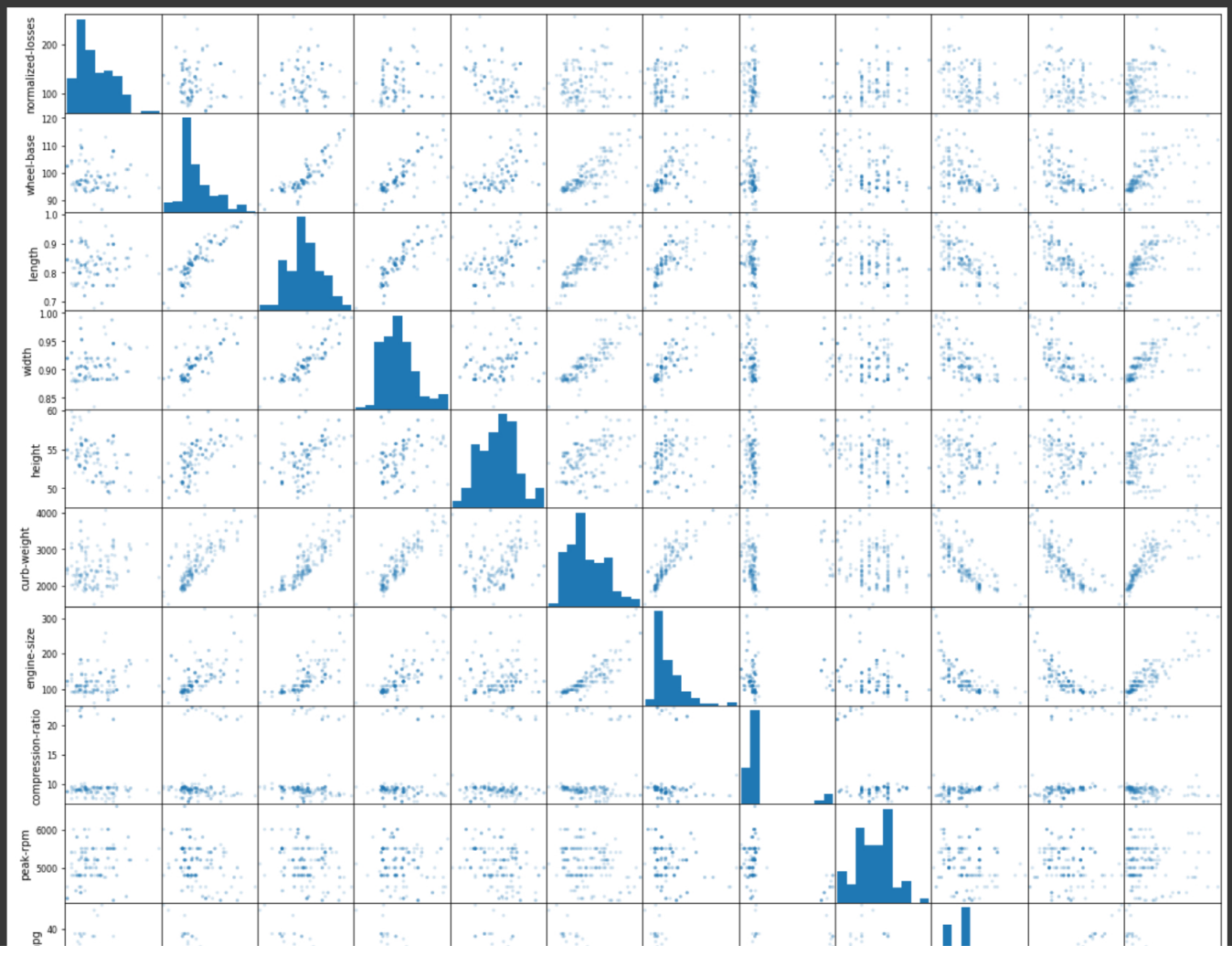
200	-1	95.0	volvo	gas	std	four	sedan	rwd	front	109.1	...	mpfi	3.78	3.15	9.5	114	5400.0
201	-1	95.0	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	mpfi	3.78	3.15	8.7	160	5300.0
202	-1	95.0	volvo	gas	std	four	sedan	rwd	front	109.1	...	mpfi	3.58	2.87	8.8	134	5500.0
203	-1	95.0	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	idi	3.01	3.40	23.0	106	4800.0
204	-1	95.0	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	mpfi	3.78	3.15	9.5	114	5400.0

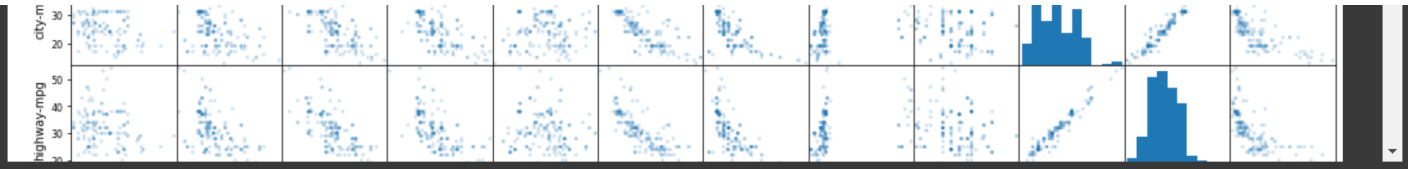
96 rows × 27 columns

[82]
```
#plotting boxplot for drive-wheel and price variables
sns.boxplot(x="drive-wheels", y="price", data=df)
plt.title('Price variation on drive-wheels')
#automobile.boxplot(column='drive-wheels',by='price')
plt.show()
```



[84]
```
#scatter plot for all the numerical variables in the dataset
from pandas.plotting import scatter_matrix
scatter_matrix(df[['normalized-losses','wheel-base','length',
                   'width','height','curb-weight','engine-size',
                   'bore','stroke','compression-ratio','horsepower',
                   'peak-rpm','city-mpg','highway-mpg','price']],alpha=0.2,figsize=(19,20),diagonal='hist')
plt.show()
```




city-m
30
20
highway-mpg
50
40
30

[ ]