The LNM Institute Of Information Technology

LNMIIT
The LNM Institute of Information Technology

Introduction to Data Science

Project report for the partial fulfillment of the requirements of the course.

Submitted by:
Aryan Singhai(20UCS036)
Arnav Arora(20UCS026)
Arjun Aggarwal(20UCS025)
Anmol Jain(20UCS022)

GitHub Repository Link:https://github.com/thetazap/IDS_Project

Department of Computer Science and Engineering, The LNM Institute of Information Technology, Jaipur.

December 2022

Description of the dataset

This dataset consists of data From 1985 Ward's Automotive Yearbook. Here are the sources

Sources:

1) 1985 Model Import Car and Truck Specifications, 1985 Ward's Automotive Yearbook.

2) Personal Auto Manuals, Insurance Services Office, 160 Water Street, New York, NY 10038

3) Insurance Collision Report, Insurance Institute for Highway Safety, Watergate 600, Washington, DC 20037

This data set consists of three types of entities: (a) the specification of an auto in terms of various characteristics, (b) its assigned insurance risk rating, (c) its normalized losses in use as compared to other cars. The second rating corresponds to the degree to which the auto is more risky than its price indicates. Cars are initially assigned a risk factor symbol associated with its price. Then, if it is more risky (or less), this symbol is adjusted by moving it up (or down) the scale. Actuarians call this process "symboling". A value of +3 indicates that the auto is risky, -3 that it is probably pretty safe.

The third factor is the relative average loss payment per insured vehicle year. This value is normalized for all autos within a particular size classification (two-door small, station wagons, sports/speciality, etc…), and represents the average loss per car per year.

Note: Several of the attributes in the database could be used as a "class" attribute.

- Number of instances: 205
- Number of attributes: 26 in total
    - 15 continuous
    - 1 integer
    - 10 nominal

Attribute Information:

1. **symboling:** -3, -2, -1, 0, 1, 2, 3.
2. **normalized-losses:** continuous values from 65 to 256.
3. **make:** manufacturer of the car (alfa-romeo, audi, bmw, chevrolet, dodge, honda, isuzu, jaguar, mazda, mercedes-benz, mercury, mitsubishi, nissan, peugeot,

plymouth, porsche, renault, saab, subaru, toyota, volkswagen, volvo)

4. **fuel-type:** the type of fuel the car uses i.e - diesel or gas.
5. **aspiration:** std, turbo.
6. **num-of-doors:** number of doors in the car- four or two.
7. **body-style:** hardtop, wagon, sedan, hatchback, convertible.
8. **drive-wheels:** 4wd, fwd, rwd.
9. **engine-location:** where the engine is located: front, rear.
10. **wheel-base:** continuous from 86.6 120.9.
11. **length:** continuous from 141.1 to 208.1.
12. **width:** continuous from 60.3 to 72.3.
13. **Height:** continuous from 47.8 to 59.8.
14. **curb-weight:** continuous from 1488 to 4066.
15. **engine-type:** dohc, dohcv, l, ohc, ohcf, ohcv, rotor.
16. **num-of-cylinders:** eight, five, four, six, three, twelve, two.
17. **engine-size:** continuous from 61 to 326.
18. **fuel-system:** 1bbl, 2bbl, 4bbl, idi, mfi, mpfi, spdi, spfi.
19. **bore:** continuous from 2.54 to 3.94.
20. **Stroke:** continuous from 2.07 to 4.17.
21. **compression-ratio:** continuous from 7 to 23.
22. **horsepower:** continuous from 48 to 288.
23. **peak-rpm:** continuous from 4150 to 6600.
24. **city-mpg:** miles per gallon in the city **(**continuous from 13 to 49).
25. **highway-mpg:** miles per gallon on highway(continuous from 16 to 54).
26. **price:** continuous from 5118 to 45400.

Source of dataset : https://archive.ics.uci.edu/ml/datasets/Automobile

The outline of this project:

1. Import and get to know the data
2. Data Cleaning
    a. Check the data type
    b. Check for the data characters mistakes
    c. Check for missing values and replace them
    d. Check for duplicate rows
    e. Statistics summary
3. Data Visualization

IMPORTING LIBRARIES AND DATA SET:

- First, we import all the necessary libraries which will be required in our code

```
In [6]:
#importing the required libraries
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sb
```

- Next, we load and read our dataset in a variable and output it.

```
In [8]:
# Collecting data
df=pd.read_csv(r"C:\Users\aryan\Downloads\Automobile_data.csv")
df=df.iloc[:,[2,3,4,5,6,7,8,10,11,12,13,14,15,16,21,23,24,25]]
df
```

Out[8]:

	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	length	width	height	curb-weight	engine-type	num-of-cylinders	engine-size	horsepower	city-mpg	highway-mpg
0	alfa-romero	gas	std	two	convertible	rwd	front	168.8	64.1	48.8	2548	dohc	four	130	111	21	27
1	alfa-romero	gas	std	two	convertible	rwd	front	168.8	64.1	48.8	2548	dohc	four	130	111	21	27
2	alfa-romero	gas	std	two	hatchback	rwd	front	171.2	65.5	52.4	2823	ohcv	six	152	154	19	26
3	audi	gas	std	four	sedan	fwd	front	176.6	66.2	54.3	2337	ohc	four	109	102	24	30
4	audi	gas	std	four	sedan	4wd	front	176.6	66.4	54.3	2824	ohc	five	136	115	18	22
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	volvo	gas	std	four	sedan	rwd	front	188.8	68.9	55.5	2952	ohc	four	141	114	23	28
201	volvo	gas	turbo	four	sedan	rwd	front	188.8	68.8	55.5	3049	ohc	four	141	160	19	25
202	volvo	gas	std	four	sedan	rwd	front	188.8	68.9	55.5	3012	ohcv	six	173	134	18	23
203	volvo	diesel	turbo	four	sedan	rwd	front	188.8	68.9	55.5	3217	ohc	six	145	106	26	27
204	volvo	gas	turbo	four	sedan	rwd	front	188.8	68.9	55.5	3062	ohc	four	141	114	19	25

205 rows × 18 columns

DATA CLEANING:

- Using the info() function we get information about our data set

Data cleaning

```
In [9]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   make              205 non-null    object
 1   fuel-type         205 non-null    object
 2   aspiration        205 non-null    object
 3   num-of-doors      205 non-null    object
 4   body-style        205 non-null    object
 5   drive-wheels      205 non-null    object
 6   engine-location   205 non-null    object
 7   length            205 non-null    float64
 8   width             205 non-null    float64
 9   height            205 non-null    float64
 10  curb-weight       205 non-null    int64
 11  engine-type       205 non-null    object
 12  num-of-cylinders  205 non-null    object
 13  engine-size       205 non-null    int64
 14  horsepower        205 non-null    object
 15  city-mpg          205 non-null    int64
 16  highway-mpg       205 non-null    int64
 17  price             205 non-null    object
dtypes: float64(3), int64(4), object(11)
memory usage: 29.0+ KB
```

- We can check the shape and columns of our dataset using .shape and .columns function

```
In [15]: df.shape
Out[15]: (205, 18)
```

```
In [16]: df.columns
Out[16]: Index(['make', 'fuel-type', 'aspiration', 'num-of-doors', 'body-style',
                'drive-wheels', 'engine-location', 'length', 'width', 'height',
                'curb-weight', 'engine-type', 'num-of-cylinders', 'engine-size',
                'horsepower', 'city-mpg', 'highway-mpg', 'price'],
               dtype='object')
```

- The describe() function returns the description of the data in the dataframe(for data frames containing numerical data only).

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

Out[11]:

	length	width	height	curb-weight	engine-size	city-mpg	highway-mpg
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	174.049268	65.907805	53.724878	2555.565854	126.907317	25.219512	30.751220
std	12.337289	2.145204	2.443522	520.680204	41.642693	6.542142	6.886443
min	141.100000	60.300000	47.800000	1488.000000	61.000000	13.000000	16.000000
25%	166.300000	64.100000	52.000000	2145.000000	97.000000	19.000000	25.000000
50%	173.200000	65.500000	54.100000	2414.000000	120.000000	24.000000	30.000000
75%	183.100000	66.900000	55.500000	2935.000000	141.000000	30.000000	34.000000
max	208.100000	72.300000	59.800000	4066.000000	326.000000	49.000000	54.000000

- Next we will search for all '?' and replace them with NaN. Then we will find the number of null values in each column and replace them with appropriate values.

```
In [17]:
#replacing ? with NaN
df.replace("?", np.nan, inplace = True)
```

```
In [18]:
#checking total number of null values in each column
df.isnull().sum()
```

```
Out[18]: make               0
fuel-type          0
aspiration         0
num-of-doors       2
body-style         0
drive-wheels       0
engine-location    0
length             0
width              0
height             0
curb-weight        0
engine-type        0
num-of-cylinders   0
engine-size        0
horsepower         2
city-mpg           0
highway-mpg        0
price              4
dtype: int64
```

```
In [21]:
avg_horsepower = df['horsepower'].astype('float').mean(axis=0)
df['horsepower'].replace(np.nan, avg_horsepower, inplace=True)
df["num-of-doors"].replace(np.nan, "four", inplace=True)
df.dropna(subset=["price"], axis=0, inplace=True)
df.reset_index(drop=True, inplace=True)
```

These null values adversely affect the performance and accuracy of any machine learning algorithm. So, it is very important to remove null values from the dataset before applying any machine learning algorithm to that dataset. Thus we do the following:

We can see that there are 2 null values in num-of-doors, 2 in horsepower and 4 in price.

We replace all the null values in column horsepower with the average horsepower of the data set.

We will replace all null values in column no-of-doors with four.(max cars have 4 doors)

And we drop all rows that had null value for price. This is done because we have no pattern yet in which we can decide to price a car.

We got rid of all null values by either replacing them or by dropping rows. Since the number of rows having null values for price is small(only 4 out of 205) we can drop them without losing major data and affecting our model.

- Next we look for duplicate rows.(if any we will eliminate them)

```
In [7]:
df.duplicated().sum()
```

```
Out[7]: 0
```

```
In [22]: df[["price"]] = df[["price"]].astype("float")
         df[["horsepower"]] = df[["horsepower"]].astype("float")

In [23]: df.dtypes

Out[23]: make                object
         fuel-type           object
         aspiration          object
         num-of-doors        object
         body-style          object
         drive-wheels        object
         engine-location     object
         length             float64
         width              float64
         height             float64
         curb-weight          int64
         engine-type         object
         num-of-cylinders    object
         engine-size          int64
         horsepower         float64
         city-mpg             int64
         highway-mpg          int64
         price              float64
         dtype: object
```

We change the datatype of price and horsepower so that it is easier to compute and compare.

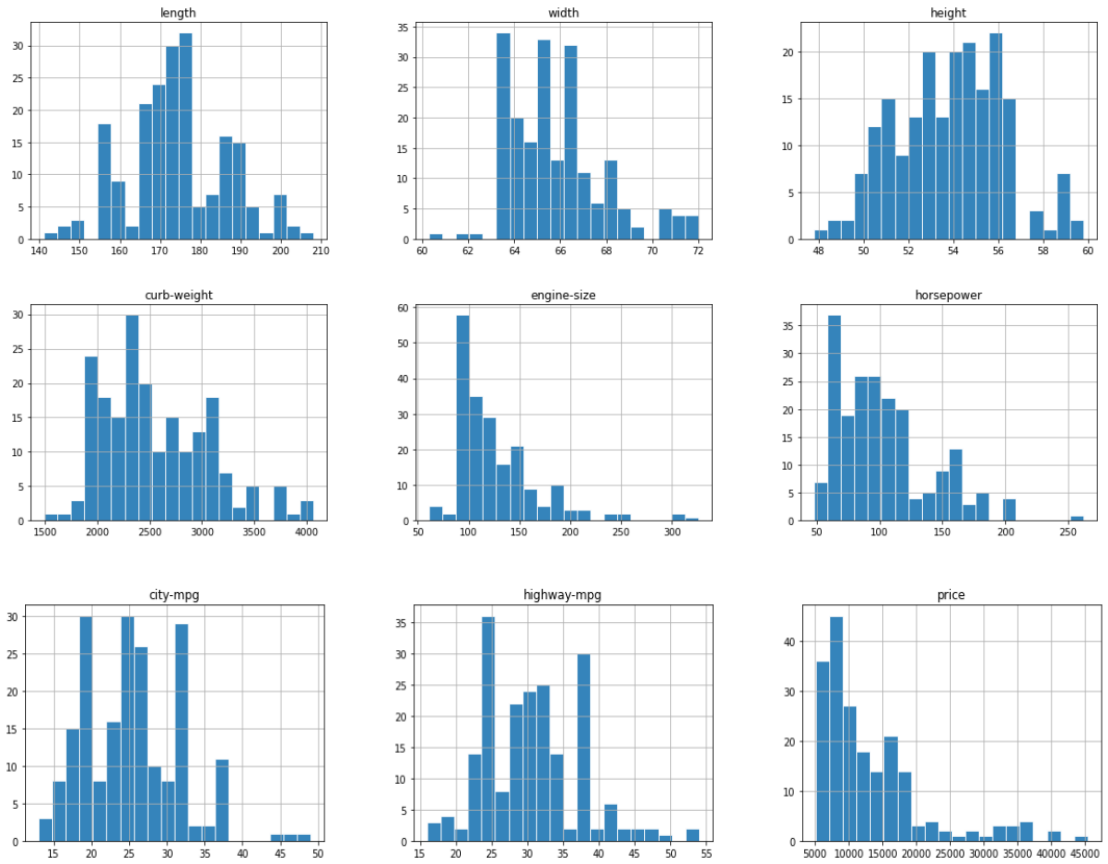
Using .dtypes function we can see the data types of all our features.

With this we are done with data cleaning.

Data Visualization

- We used hist() to plot histogram for all the attributes in the dataset

```
In [52]: df.hist(figsize = (20,15),ec='white',bins=20,alpha=0.9)
         plt.show()
```

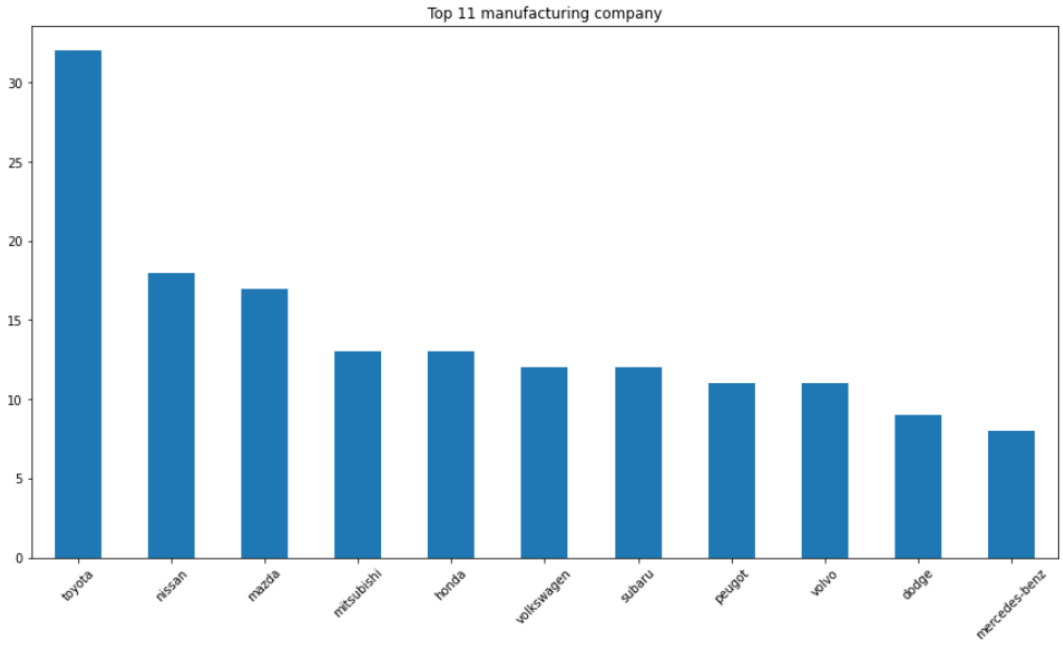


Inference:

- This graph gives the histogram plots for each attribute in our dataset that has numerical value.
- The x-value gives the range of values and the y-value gives the count(frequency).
- Length, curb-weight are nearly normal.
- Width, engine-size, price, horsepower are left-skewed.
- Height is right-skewed.

- Next we have plotted a bar graph between car brands(make) and the number of cars they manufacture

```
In [27]:
fig, ax = plt.subplots(figsize=(15,8))
df["make"].value_counts().head(11).plot.bar(ax=ax)
plt.xticks(rotation=45)
plt.title("Top 11 manufacturing company");
```

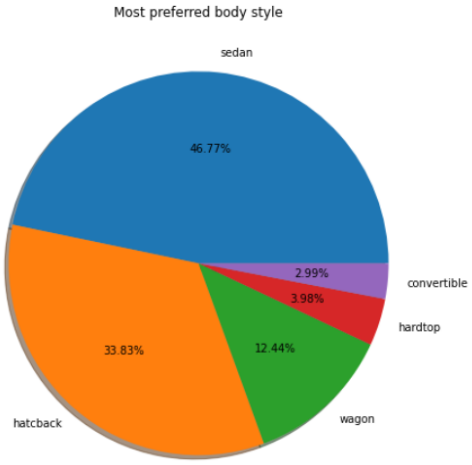


We can see that Toyota is the leading manufacturer of cars followed by Nissan and mazda.

The graph gets pretty even after that.

- Next we have plotted a pie chart that shows the preference in car body style.

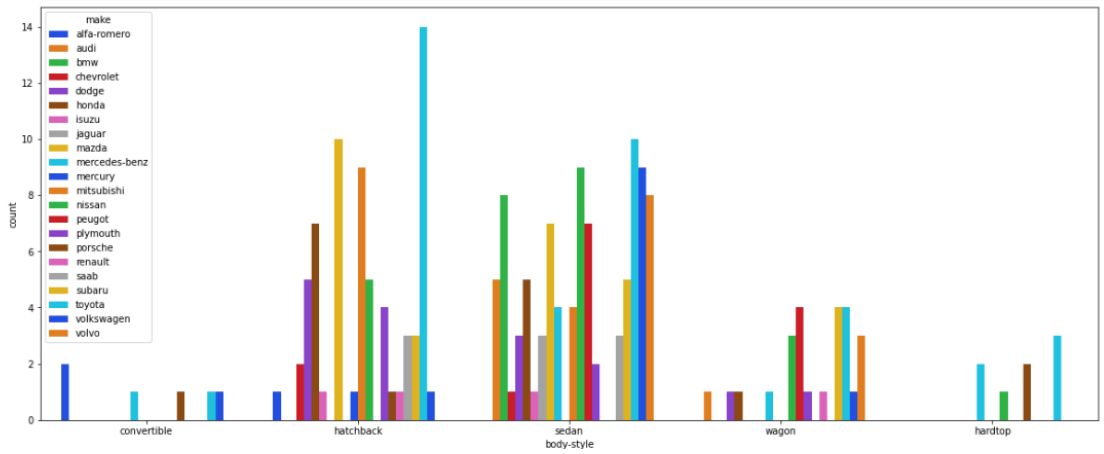
```
In [29]:
body = ['sedan', 'hatcback', 'wagon',
        'hardtop', 'convertible']
data = df["body-style"].value_counts()
fig = plt.figure(figsize =(15, 8))
plt.pie(data, labels = body,autopct='%1.2f%%', shadow=True)
plt.title("Most preferred body style");
```



We can see that most cars sold are sedan followed by hatchback.

- The next plot shows the relationship between car manufacturers and the car-body type they manufacture the most.

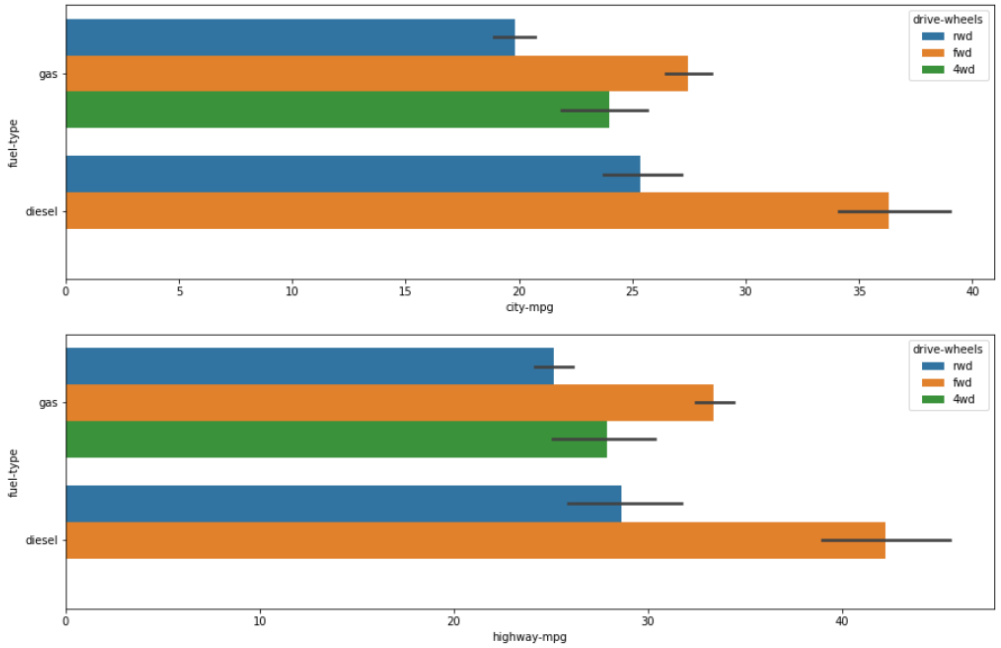
```
In [31]:
fig, ax = plt.subplots(figsize=(20,8))
sb.countplot(x='body-style', data=df, orient='h',hue='make',ax=ax,palette='bright');
```



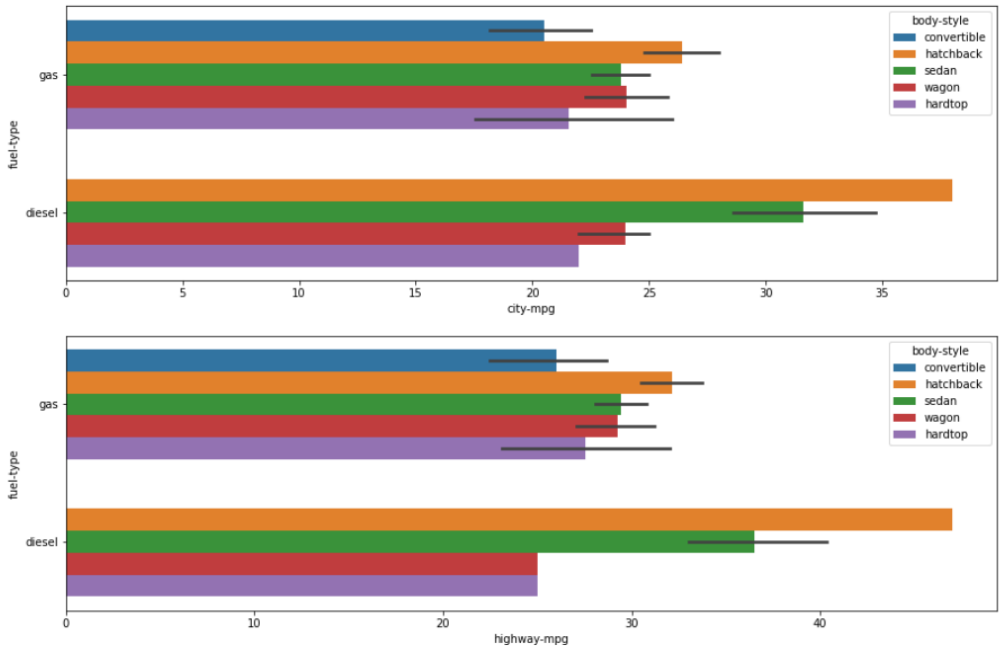
We can see that Toyota is the first manufacturer in Sedan body style. The next places are shared by Volkswagen and Nissan. Similarly we can make other observations from this graph.

- Next plot will show the relationship between fuel-type and mileage.

```
In [33]:
fig, [ax1, ax2] = plt.subplots(2, 1, figsize=(15,10))
sb.barplot(data=df, x="city-mpg", y="fuel-type",hue="drive-wheels",ax=ax1)
sb.barplot(data=df, x="highway-mpg", y="fuel-type",hue="drive-wheels",ax=ax2)
plt.show()
```



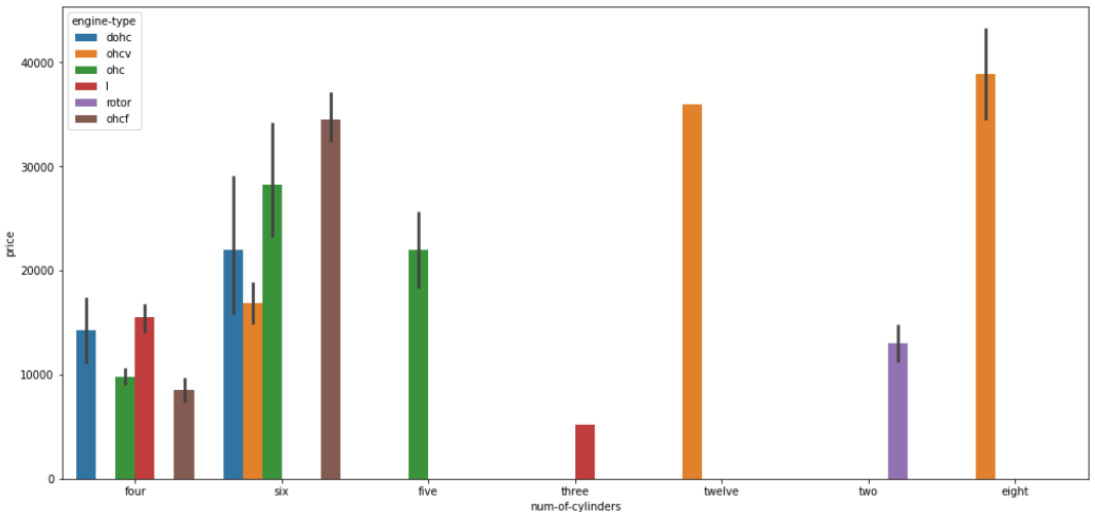
```
In [36]:
fig, [ax1, ax2] = plt.subplots(2, 1, figsize=(15,10))
sb.barplot(data=df, x="city-mpg",y="fuel-type",orient='h',hue="body-style",ax=ax1)
sb.barplot(data=df, x="highway-mpg",y="fuel-type",orient='h',hue="body-style",ax=ax2)
plt.show()
```



Here we can see that both city and highway fuel consumption averages higher in the diesel type.

- Next graph shows the relationship between price and no-of-cylinders in the car.

```
In [32]:
fig, ax = plt.subplots(figsize=(17,8))
sb.barplot(data=df,y="price",x='num-of-cylinders',hue="engine-type",ax=ax);
```

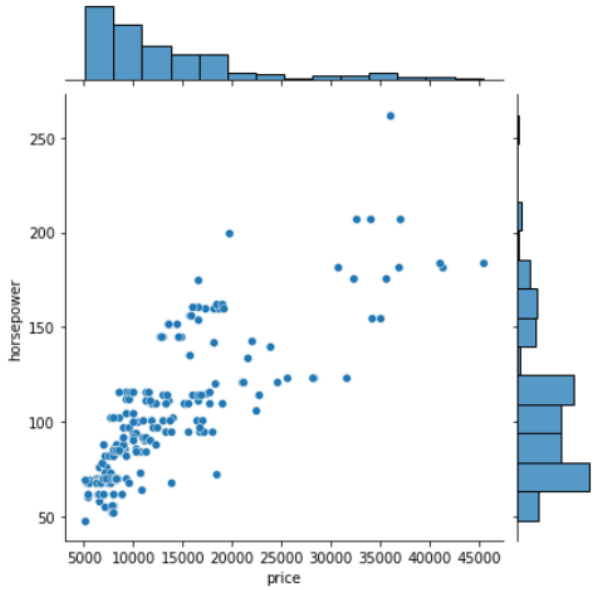


In this graph, the six cylinder car has a wide variety and high average price. But eight cylinder car has a higher average price

- Next graph shows the relationship between price and horsepower.

```
In [33]: sb.jointplot(x='price', y='horsepower',data=df)
```

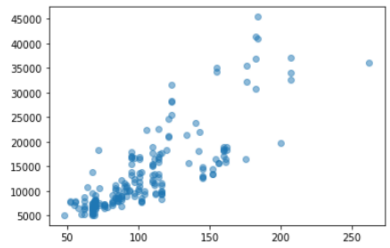
```
Out[33]: <seaborn.axisgrid.JointGrid at 0x29e10ee4910>
```



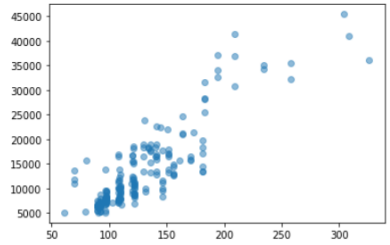
We can see price linearly increases with horsepower.

- We will look at some scatter plots between our target variable price and other variables.

```
In [35]: plt.scatter(df['horsepower'], df['price'], alpha=0.5)
plt.show()
```

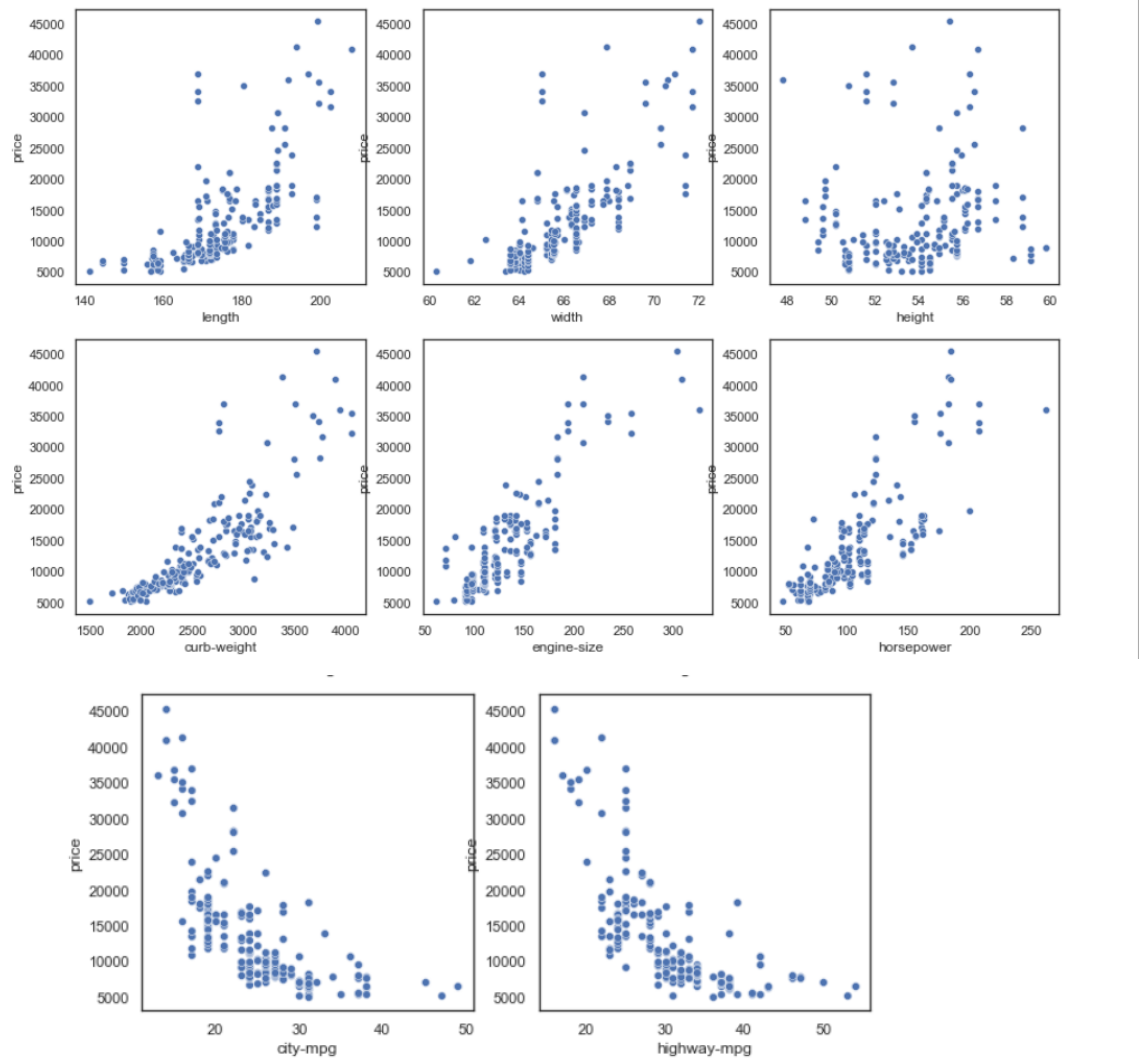



```
In [36]: plt.scatter(df['engine-size'], df['price'], alpha=0.5)
plt.show()
```



We can see that price increases linearly with both horsepower and engine size.

In [41]:
```
num_features = ['length','width','height','curb-weight','engine-size','horsepower','city-mpg','highway-mpg']
plt.figure(figsize = (15,25))
for i in enumerate (num_features):
    plt.subplot(5,3,i[0]+1)
    sb.scatterplot(x = i[1],y = 'price',data = df)
```



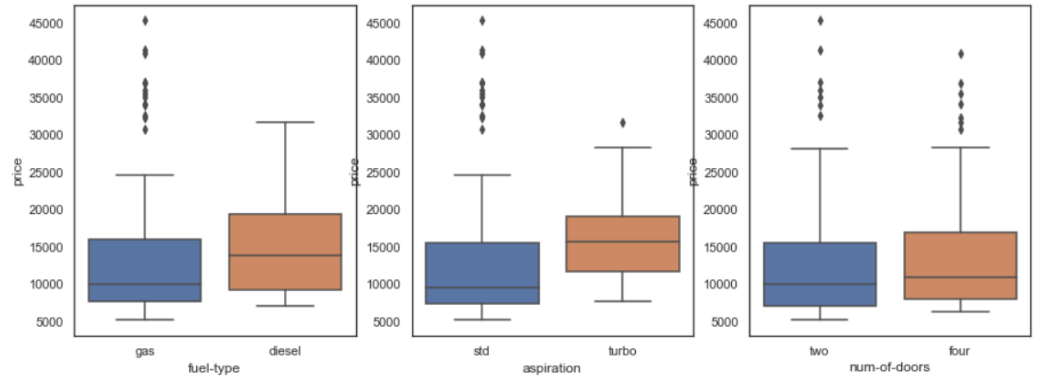
Inference:

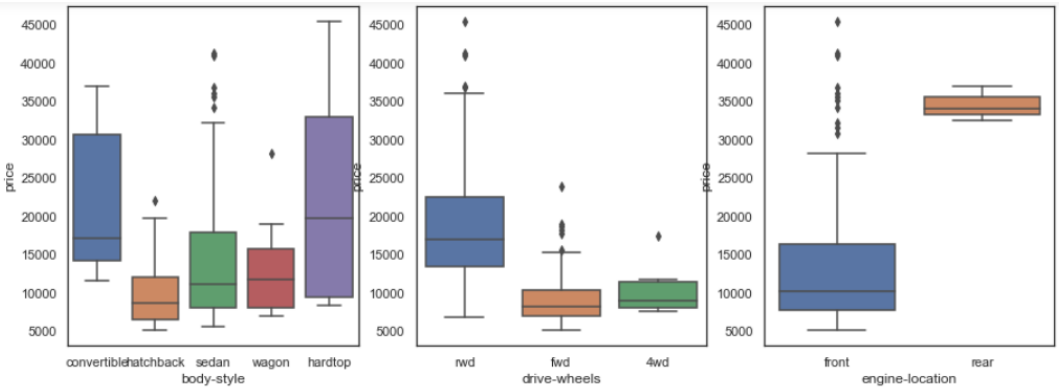
- From these scatter plots we observe that price is inversely proportional to both city-mpg and highway-mpg.
- On the other hand it is directly proportional to length, width, curb-weight, engine-size and horsepower.

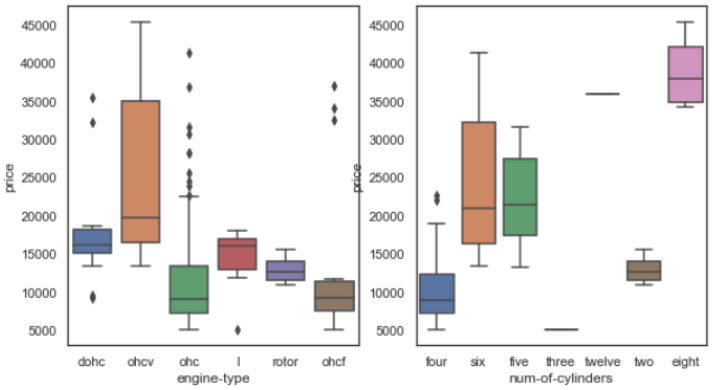
- Now we will look at some box plots between our target variable price and other variables.

In [42]:
```
cat_features = ['fuel-type', 'aspiration',
       'num-of-doors', 'body-style', 'drive-wheels', 'engine-location',
        'engine-type','num-of-cylinders']
plt.figure(figsize = (15,18))
for i in enumerate (cat_features):
    plt.subplot(3,3,i[0]+1)
    sb.boxplot(x = i[1],y = "price",data = df)
sb.catplot(y="make", x="price", kind="box", data=df)
```

Out[42]: <seaborn.axisgrid.FacetGrid at 0x29e105e4b80>

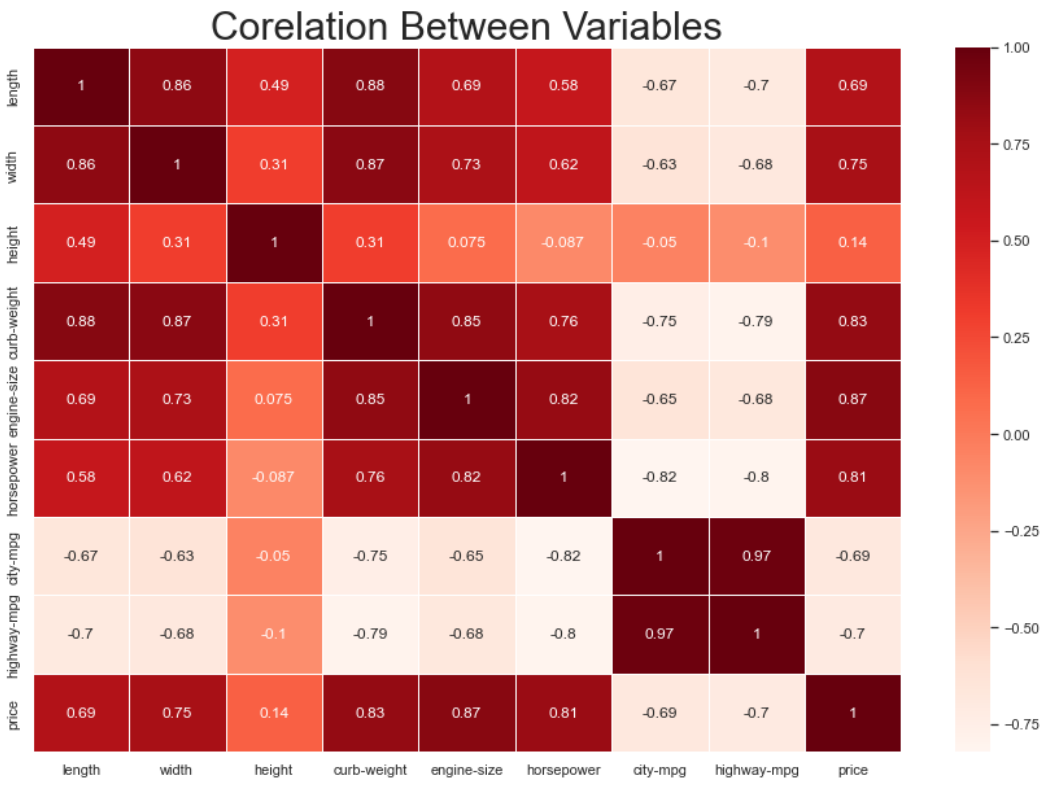






CORRELATION BETWEEN VARIABLES

```
In [37]:  sb.set(style="white")
          plt.rcParams['figure.figsize'] = (15, 10)
          sb.heatmap(df.corr(), annot = True, linewidths=.5, cmap="Reds")
          plt.title('Corelation Between Variables', fontsize = 30)
          plt.show()
```



Inference:

- We see that price has low correlation with height(0.14)
- Price has a high positive correlation with length, width, crub-weight, engine-size and horsepower.
- On the other hand, price has a high negative correlation with city-mpg and highway-mpg.

Conclusion

From this dataset we can conclude that the most common car sold is ‘sedan’.
We can also conclude that both city and highway fuel consumption averages higher in the diesel type.
We observe from both the scatter plots and correlation matrix that our target variable price has a high positive correlation with length,width, curb-weight, engine-size and horsepower and a high negative correlation with city-mpg and highway-mpg.

Acknowledgement

We would like to thank Prof. Sakthi Balan Muthiah, Prof. Aloke Datta , Prof. Subrat K. Dash for giving us an opportunity to work upon this project on Data Science. This project has been completed only because of the team effort and cooperation from Aryan Singhai, Arnav Arora, Arjun Aggarwal and Anmol Jain.

Reference

- Class lectures.
- Shubham Singh Ghar Sale. (2018). Exploratory Data Analysis on Automobile Dataset.
- https://www.w3schools.com/python/numpy/numpy_intro.asp
- https://www.w3schools.com/python/matplotlib_pyplot.asp
- https://seaborn.pydata.org/api.html
- https://towardsdatascience.com/
- Source of dataset : https://archive.ics.uci.edu/ml/datasets/Automobile