Car Price Prediction

Problem

A German automobile company enter the US market by setting up their manufacturing unit there and producing cars locally to give competition to their US and European counterparts.

They have contracted an automobile consulting company to understand the factors on which the pricing of cars depends. Specifically, they want to understand the factors affecting the pricing of cars in the American market, since those may be very different from the Chinese market. The company wants to know:

```
- Which variables are significant in predicting the price of a car
- How well those variables describe the price of a car
```

Based on various market surveys, the consulting firm has gathered a large dataset of different types of cars across the Americal market.

Goal

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

In [1]:
```
import warnings
warnings.filterwarnings('ignore')

#importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
from sklearn.ensemble import RandomForestRegressor
```

Step 1: Reading and Understanding the Data

1. Importing data using the pandas library
2. Understanding the structure of the data

In [2]:
```
cars = pd.read_csv('Car Price.csv')
cars.head()
```

Out[2]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	engineloc
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	

5 rows × 26 columns

Dimentions

In [3]:
```
# In simple words
print("Rows:"+str(len(cars)))
print("Colums:"+str(len(cars.columns)))
cars.shape
```

```
Rows:205
Colums:26
```

Out[3]:
```
(205, 26)
```

In [4]:
```
cars.describe()
```

Out[4]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000

In [5]:
```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   car_ID            205 non-null    int64  
 1   symboling         205 non-null    int64  
 2   CarName           205 non-null    object 
 3   fueltype          205 non-null    object 
 4   aspiration        205 non-null    object 
 5   doornumber        205 non-null    object 
 6   carbody           205 non-null    object 
 7   drivewheel        205 non-null    object 
 8   enginelocation    205 non-null    object 
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64  
 14  enginetype        205 non-null    object 
 15  cylindernumber    205 non-null    object 
 16  enginesize        205 non-null    int64  
 17  fuelsystem        205 non-null    object 
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64  
 22  peakrpm           205 non-null    int64  
 23  citympg           205 non-null    int64  
 24  highwaympg        205 non-null    int64  
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

Step 2 : Data Cleaning and Preprocessing

In [6]:
```
#Splitting company name from CarName column
def formodel(x):
    if len(x.split(' ',1)) is 1:
        return pd.NA
    return x.split(' ')[1]
CompanyName = cars['CarName'].apply(lambda x : x.split(' ')[0])
ModelName = cars['CarName'].apply(formodel)
cars.insert(3,"ModelName",ModelName)
cars.insert(3,"CompanyName",CompanyName)
cars.drop(['CarName'],axis=1,inplace=True)
cars.head()
```

Out[6]:

	car_ID	symboling	CompanyName	ModelName	fueltype	aspiration	doornumber	carbody	drive
0	1	3	alfa-romero	giulia	gas	std	two	convertible	
1	2	3	alfa-romero	stelvio	gas	std	two	convertible	
2	3	1	alfa-romero	Quadrifoglio	gas	std	two	hatchback	
3	4	2	audi	100	gas	std	four	sedan	
4	5	2	audi	100ls	gas	std	four	sedan	

5 rows × 27 columns

In [7]:
```
cars.CompanyName.unique()
```

Out[7]:
```
array(['alfa-romero', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda',
       'isuzu', 'jaguar', 'maxda', 'mazda', 'buick', 'mercury',
       'mitsubishi', 'Nissan', 'nissan', 'peugeot', 'plymouth', 'porsche',
       'porcshce', 'renault', 'saab', 'subaru', 'toyota', 'toyouta',
       'vokswagen', 'volkswagen', 'vw', 'volvo'], dtype=object)
```

Fixing invalid values

- There seems to be some spelling error in the CompanyName column.
  - `maxda` = `mazda`
  - `Nissan` = `nissan`
  - `porsche` = `porcshce`
  - `toyota` = `toyouta`
  - `vokswagen` = `volkswagen` = `vw`

In [8]:
```
cars.CompanyName = cars.CompanyName.str.lower()

def replace_name(a,b):
    cars.CompanyName.replace(a,b,inplace=True)

replace_name('maxda','mazda')
replace_name('porcshce','porsche')
replace_name('toyouta','toyota')
replace_name('vokswagen','volkswagen')
replace_name('vw','volkswagen')

cars.CompanyName.unique()
```

Out[8]:
```
array(['alfa-romero', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda',
       'isuzu', 'jaguar', 'mazda', 'buick', 'mercury', 'mitsubishi',
       'nissan', 'peugeot', 'plymouth', 'porsche', 'renault', 'saab',
       'subaru', 'toyota', 'volkswagen', 'volvo'], dtype=object)
```

Or Print Comapny Names in Images, It's not needed but it's cool

In [9]:
```
!pip install wordcloud
```

```
Requirement already satisfied: wordcloud in c:\programdata\anaconda3\lib\site-packages
(1.8.1)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-packages
(from wordcloud) (3.3.4)
Requirement already satisfied: pillow in c:\programdata\anaconda3\lib\site-packages (fro
m wordcloud) (8.2.0)
Requirement already satisfied: numpy>=1.6.1 in c:\programdata\anaconda3\lib\site-package
s (from wordcloud) (1.20.1)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-package
s (from matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\programdat
a\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\site-pa
ckages (from matplotlib->wordcloud) (1.3.1)
Requirement already satisfied: python-dateutil>=2.1 in c:\programdata\anaconda3\lib\site
-packages (from matplotlib->wordcloud) (2.8.1)
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packages (from c
ycler>=0.10->matplotlib->wordcloud) (1.15.0)
```

In [10]:
```
from wordcloud import WordCloud, ImageColorGenerator
# Create and generate a word cloud image:
```

```
wordcloud = WordCloud(max_words=200,colormap='Dark2', background_color="WHITE").generat
plt.figure(figsize=(10,6))
plt.figure(figsize=(15,10))
# Display the generated image:
plt.imshow(wordcloud, interpolation='Bilinear')
plt.axis("off")
plt.figure(1,figsize=(12, 12))
plt.show()
```


```
<Figure size 720x432 with 0 Axes>
```

In [11]:
```
#Checking for duplicates
cars.loc[cars.duplicated()]
```

Out[11]:

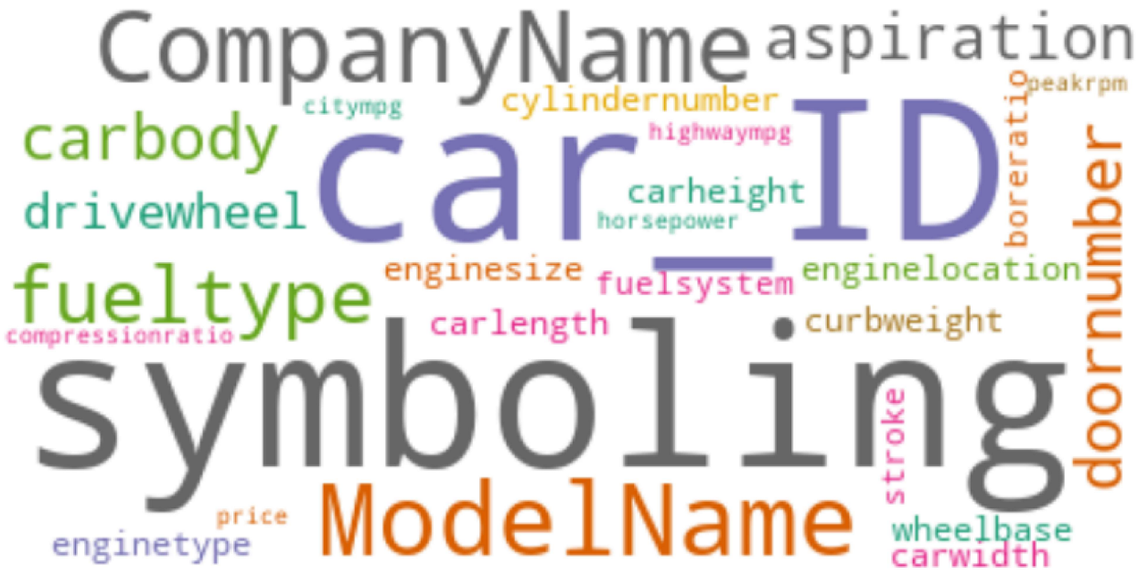
car_ID	symboling	CompanyName	ModelName	fueltype	aspiration	doornumber	carbody	drivewh

0 rows × 27 columns

In [12]:
```
cars.columns
```

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

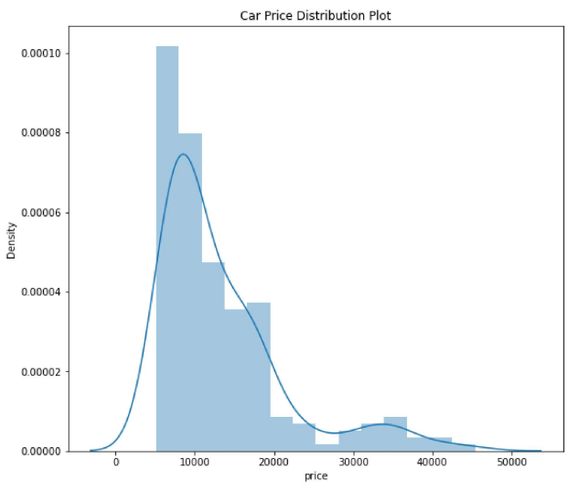
In [13]:
```
from wordcloud import WordCloud, ImageColorGenerator
# Create and generate a word cloud image:
wordcloud = WordCloud(max_words=200,colormap='Dark2', background_color="WHITE").generat
plt.figure(figsize=(10,6))
plt.figure(figsize=(15,10))
# Display the generated image:
plt.imshow(wordcloud, interpolation='Bilinear')
plt.axis("off")
plt.figure(1,figsize=(12, 12))
plt.show()
```

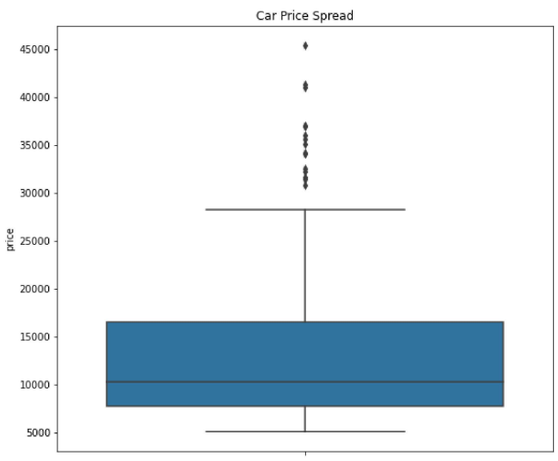


<Figure size 720x432 with 0 Axes>

Step 3: Visualizing the data

In [14]:
```
plt.figure(figsize=(20,8))

plt.subplot(1,2,1)
plt.title('Car Price Distribution Plot')
sns.distplot(cars.price)

plt.subplot(1,2,2)
plt.title('Car Price Spread')
sns.boxplot(y=cars.price)

plt.show()
```





In [15]:
```
#word cloud
from wordcloud import WordCloud, ImageColorGenerator
text = " ".join(str(each) for each in cars.CompanyName)
# Create and generate a word cloud image:
wordcloud = WordCloud(max_words=200,colormap='Dark2', background_color="white").generat
plt.figure(figsize=(10,6))
plt.figure(figsize=(15,10))
# Display the generated image:
plt.imshow(wordcloud, interpolation='Bilinear')
plt.axis("off")
plt.figure(1,figsize=(12, 12))
plt.show()
```

<Figure size 720x432 with 0 Axes>

In [16]:
```
#word cloud
from wordcloud import WordCloud, ImageColorGenerator
text = " ".join(str(each) for each in cars.carbody)
# Create and generate a word cloud image:
wordcloud = WordCloud(max_words=200,colormap='Reds', background_color="black").generate
plt.figure(figsize=(10,6))
plt.figure(figsize=(15,10))
# Display the generated image:
plt.imshow(wordcloud, interpolation='Bilinear')
plt.axis("off")
plt.figure(1,figsize=(12, 12))
plt.show()
```

<Figure size 720x432 with 0 Axes>

In [17]:
```
#word cloud
from wordcloud import WordCloud, ImageColorGenerator
text = " ".join(str(each) for each in cars.ModelName)
# Create and generate a word cloud image:
wordcloud = WordCloud(max_words=200,colormap='Reds', background_color="black").generate
plt.figure(figsize=(10,6))
plt.figure(figsize=(15,10))
# Display the generated image:
plt.imshow(wordcloud, interpolation='Bilinear')
plt.axis("off")
plt.figure(1,figsize=(12, 12))
plt.show()
```

<Figure size 720x432 with 0 Axes>

Serioulsy? "Toyota Corona" I've notieced that there are also some wrongs names in Model, and this is also no usefull untill we have CAR_ID So let's drop this column

In [18]:
```
cars.drop(['ModelName'],axis=1,inplace=True)
cars.head()
```

Out[18]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	engin
0	1	3	alfa-romero	gas	std	two	convertible	rwd	
1	2	3	alfa-romero	gas	std	two	convertible	rwd	
2	3	1	alfa-romero	gas	std	two	hatchback	rwd	
3	4	2	audi	gas	std	four	sedan	fwd	
4	5	2	audi	gas	std	four	sedan	4wd	

5 rows × 26 columns

In [19]:
```
print(cars.price.describe(percentiles = [0.25,0.50,0.75,0.85,0.90,1]))
```

```
count      205.000000
mean     13276.710571
std       7988.852332
min       5118.000000
25%       7788.000000
50%      10295.000000
75%      16503.000000
85%      18500.000000
90%      22563.000000
100%     45400.000000
max      45400.000000
Name: price, dtype: float64
```

Inference :

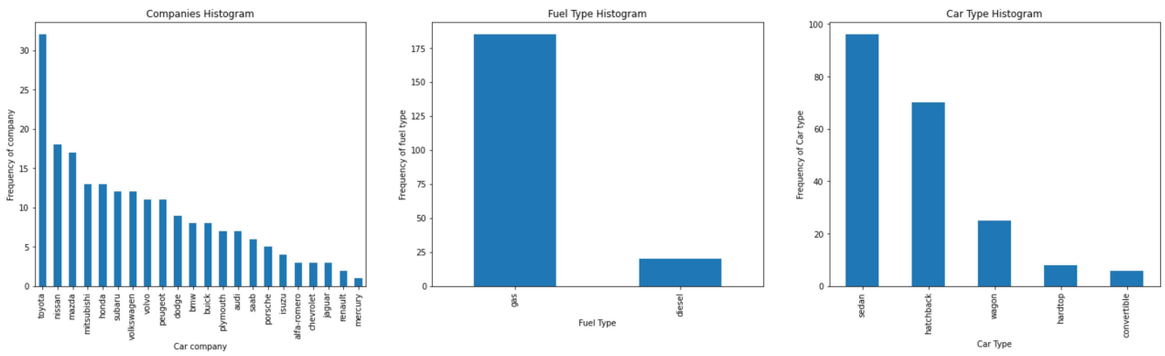
1. The plot seemed to be right-skewed, meaning that the most prices in the dataset are low(Below 15,000).
2. There is a significant difference between the mean and the median of the price distribution.
3. The data points are far spread out from the mean, which indicates a high variance in the car prices.(85% of the prices are below 18,500, whereas the remaining 15% are between 18,500 and 45,400.)

Step 3.1 : Visualising Categorical Data

- CompanyName
- Symboling
- fueltype
- enginetype
- carbody
- doornumber
- enginelocation

- fuelsystem
- cylindernumber
- aspiration
- drivewheel

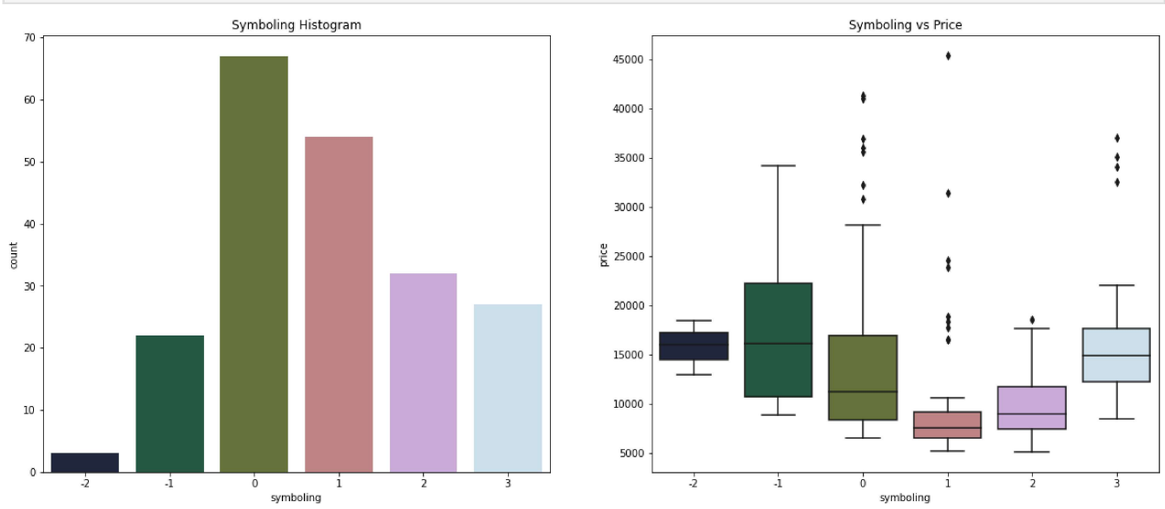
In [20]:
```
plt.figure(figsize=(25, 6))

plt.subplot(1,3,1)
plt1 = cars.CompanyName.value_counts().plot(kind='bar')
plt.title('Companies Histogram')
plt1.set(xlabel = 'Car company', ylabel='Frequency of company')

plt.subplot(1,3,2)
plt1 = cars.fueltype.value_counts().plot(kind='bar')
plt.title('Fuel Type Histogram')
plt1.set(xlabel = 'Fuel Type', ylabel='Frequency of fuel type')

plt.subplot(1,3,3)
plt1 = cars.carbody.value_counts().plot(kind='bar')
plt.title('Car Type Histogram')
plt1.set(xlabel = 'Car Type', ylabel='Frequency of Car type')

plt.show()
```



Inference :

1. `Toyota` seemed to be favored car company.
2. Number of `gas` fueled cars are more than `diesel`.
3. `sedan` is the top car type prefered.

In [21]:
```
plt.figure(figsize=(20,8))

plt.subplot(1,2,1)
plt.title('Symboling Histogram')
sns.countplot(cars.symboling, palette=("cubehelix"))

plt.subplot(1,2,2)
plt.title('Symboling vs Price')
sns.boxplot(x=cars.symboling, y=cars.price, palette=("cubehelix"))

plt.show()
```

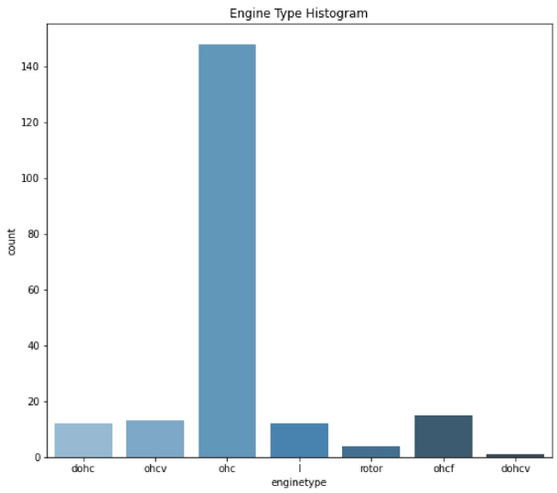


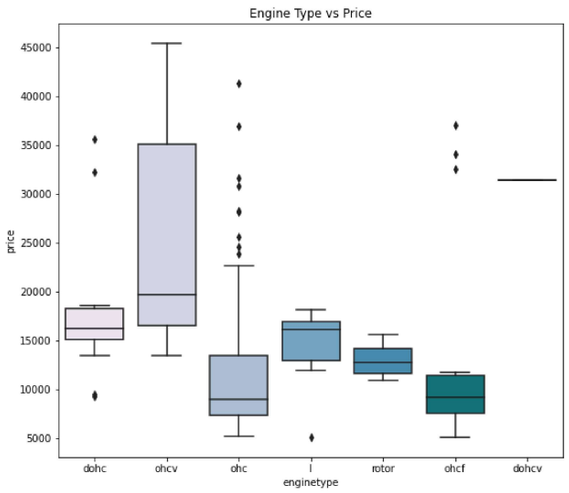
Inference :

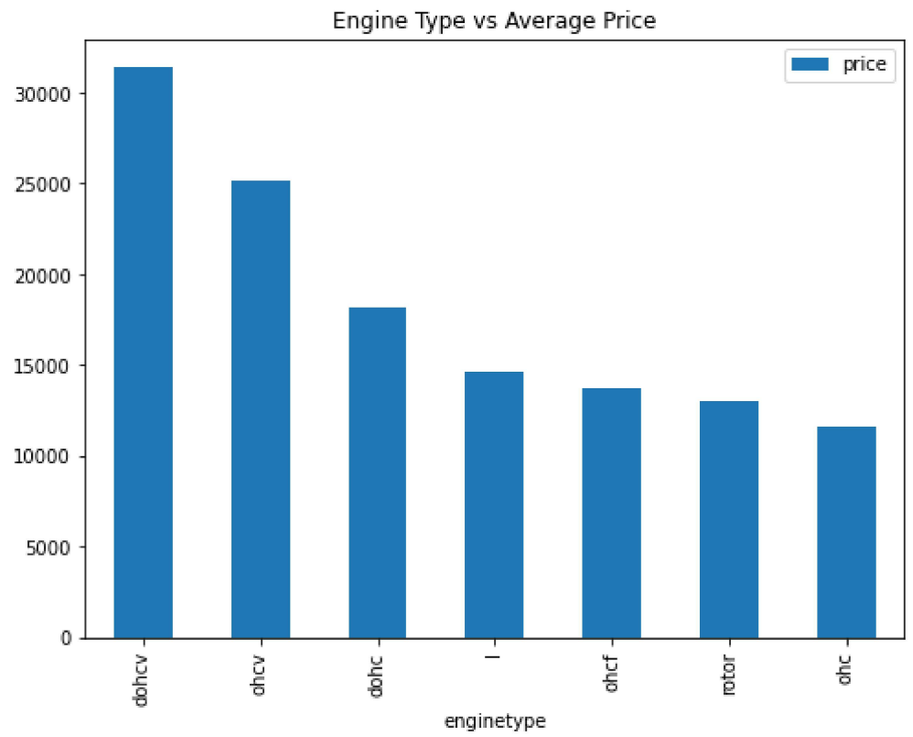
1. It seems that the symboling with `0` and `1` values have high number of rows (i.e. They are most sold.)

2. The cars with `-1` symboling seems to be high priced (as it makes sense too, insurance risk rating -1 is quite good). But it seems that symboling with `3` value has the price range similar to `-2` value. There is a dip in price at symboling `1`.

In [22]:
```
plt.figure(figsize=(20,8))

plt.subplot(1,2,1)
plt.title('Engine Type Histogram')
sns.countplot(cars.enginetype, palette=("Blues_d"))

plt.subplot(1,2,2)
plt.title('Engine Type vs Price')
sns.boxplot(x=cars.enginetype, y=cars.price, palette=("PuBuGn"))

plt.show()

df = pd.DataFrame(cars.groupby(['enginetype'])['price'].mean().sort_values(ascending =
df.plot.bar(figsize=(8,6))
plt.title('Engine Type vs Average Price')
plt.show()
```







In [23]:
```
cars.head()
```

Out[23]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	engin
0	1	3	alfa-romero	gas	std	two	convertible	rwd	
1	2	3	alfa-romero	gas	std	two	convertible	rwd	
2	3	1	alfa-romero	gas	std	two	hatchback	rwd	
3	4	2	audi	gas	std	four	sedan	fwd	
4	5	2	audi	gas	std	four	sedan	4wd	

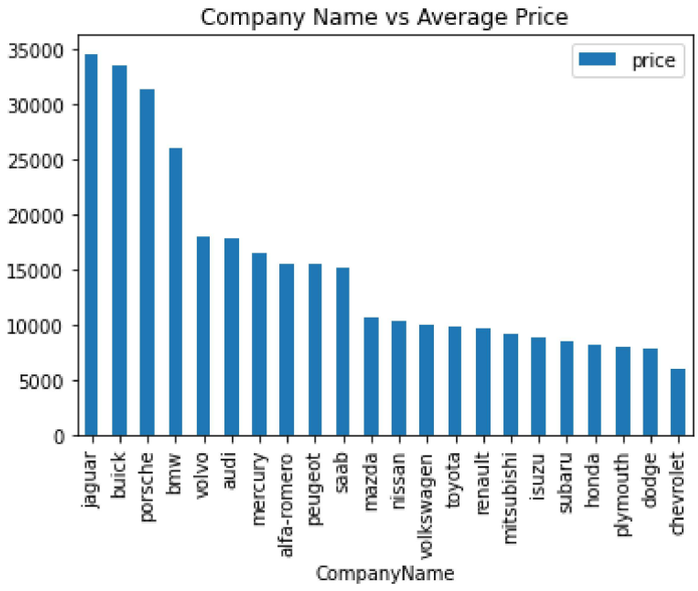
5 rows × 26 columns

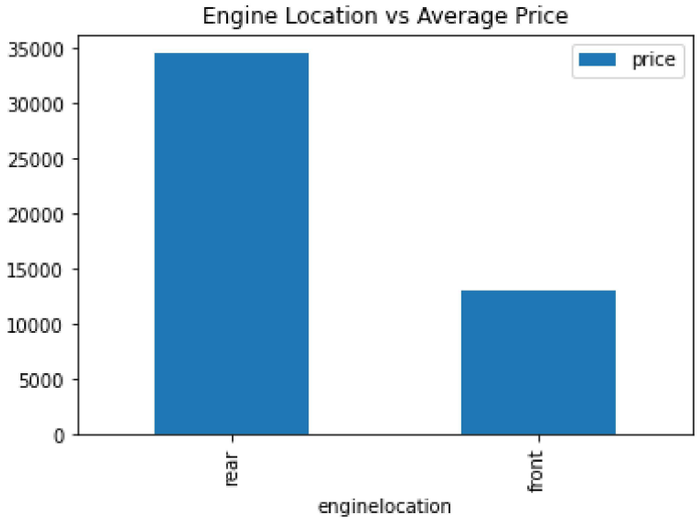
Inference :

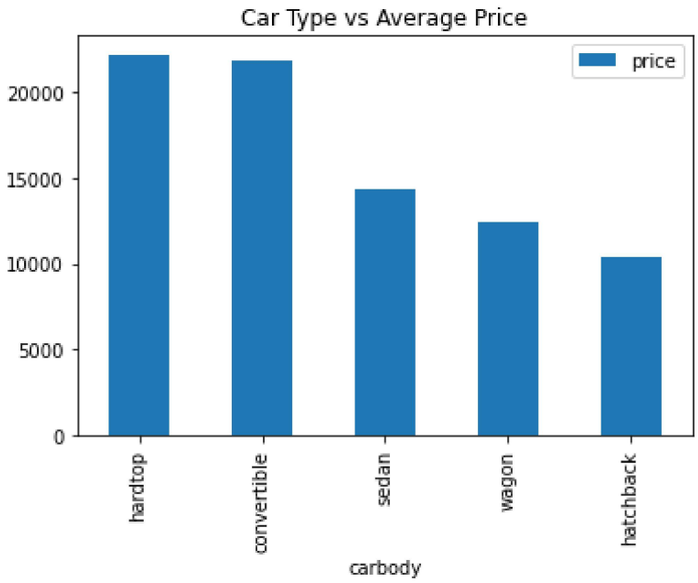
1. `ohc` Engine type seems to be most favored type.
2. `ohcv` has the highest price range (While `dohcv` has only one row), `ohc` and `ohcf` have the low price range.

In [24]:
```
plt.figure(figsize=(25, 6))

df = pd.DataFrame(cars.groupby(['CompanyName'])['price'].mean().sort_values(ascending =
df.plot.bar()
plt.title('Company Name vs Average Price')
plt.show()

df = pd.DataFrame(cars.groupby(['enginelocation'])['price'].mean().sort_values(ascendin
df.plot.bar()
plt.title('Engine Location vs Average Price')
plt.show()

df = pd.DataFrame(cars.groupby(['carbody'])['price'].mean().sort_values(ascending = Fal
df.plot.bar()
plt.title('Car Type vs Average Price')
plt.show()
```

```
<Figure size 1800x432 with 0 Axes>
```







In [25]:
```
cars.columns
```

Out[25]:
```
Index(['car_ID', 'symboling', 'CompanyName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price'],
      dtype='object')
```

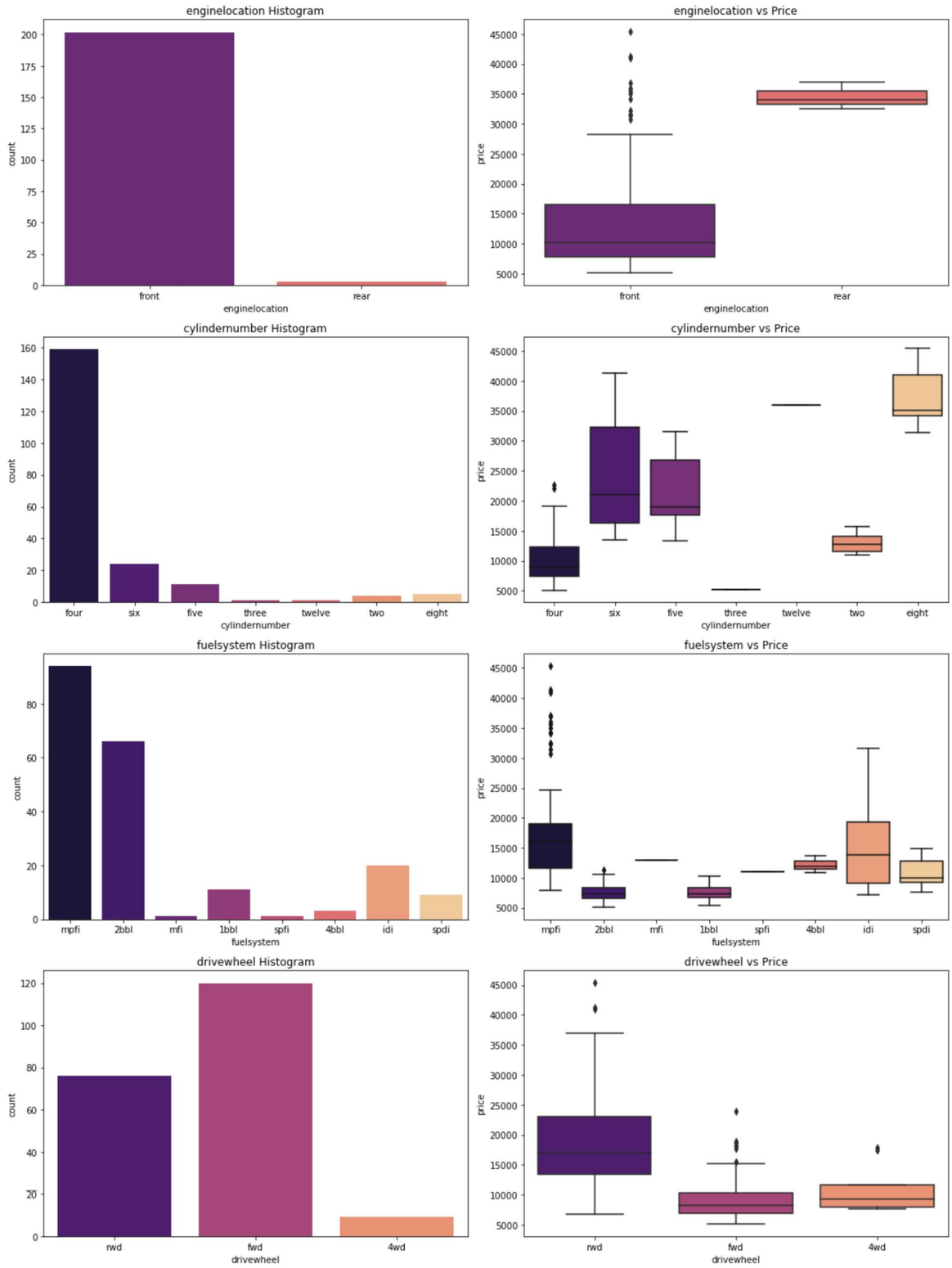
Inference :

1. `Jaguar` and `Buick` seem to have highest average price.
2. `rare engine car` has higher average price than front engine.
3. `hardtop` and `convertible` have higher average price.

Inference :

1. Low Budget Cars has better Fuel Economy than Expensive Cars
2. `Turbo` have higher price range than the `Standard`

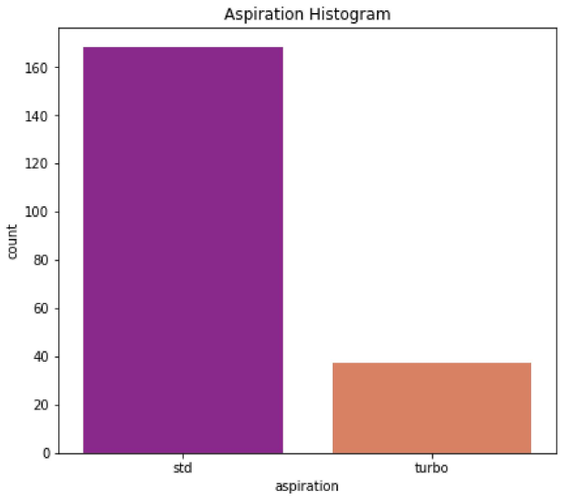
In [26]:
```
def plot_count(x,fig):
    plt.subplot(4,2,fig)
    plt.title(x+' Histogram')
    sns.countplot(cars[x],palette=("magma"))
    plt.subplot(4,2,(fig+1))
    plt.title(x+' vs Price')
    sns.boxplot(x=cars[x], y=cars.price, palette=("magma"))

plt.figure(figsize=(15,20))

plot_count('enginelocation', 1)
plot_count('cylindernumber', 3)
plot_count('fuelsystem', 5)
plot_count('drivewheel', 7)

plt.tight_layout()
```

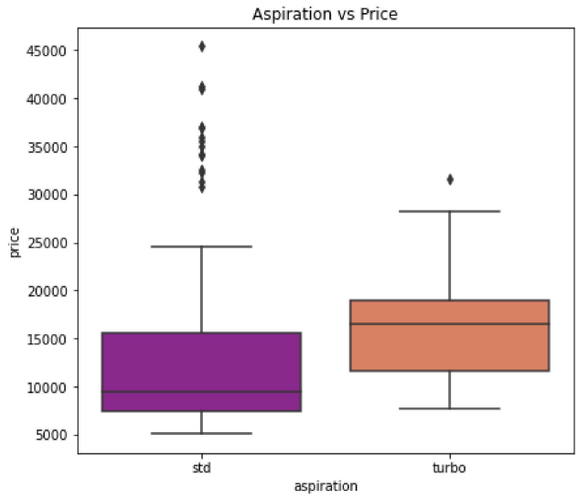


Inference :

1. Very few datapoints for `enginelocation` categories to make an inference.
2. Most common number of cylinders are `four` , `six` and `five` . Though `eight` cylinders have the highest price range.
3. `mpfi` and `2bbl` are most common type of fuel systems. `mpfi` and `idi` having the highest price range. But there are few data for other categories to derive any meaningful inference
4. A very significant difference in drivewheel category. Most high ranged cars seeme to prefer `rwd` drivewheel.

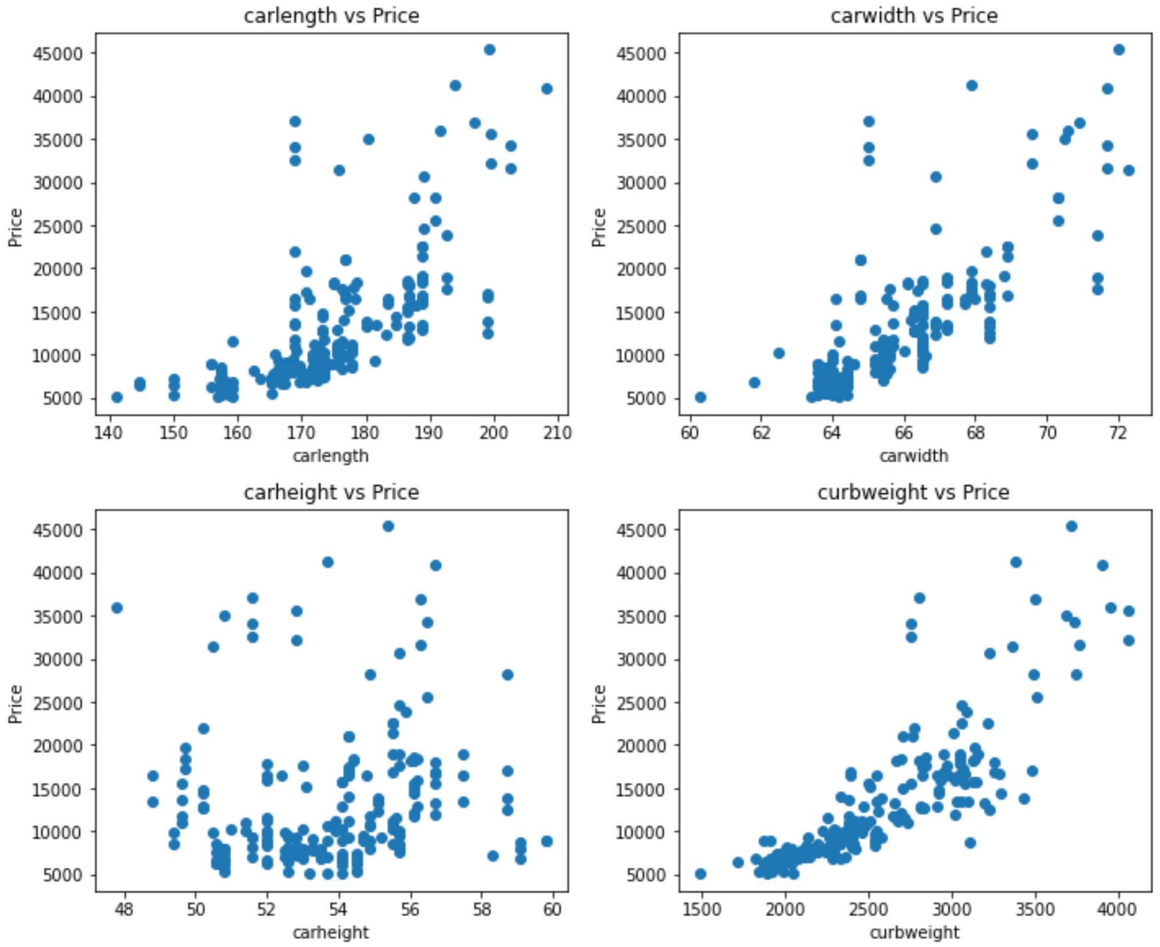
In [27]:
```
plt.figure(figsize=(15,6))

plt.subplot(1,2,1)
plt.title('Aspiration Histogram')
sns.countplot(cars.aspiration, palette=("plasma"))

plt.subplot(1,2,2)
plt.title('Aspiration vs Price')
sns.boxplot(x=cars.aspiration, y=cars.price, palette=("plasma"))

plt.show()
```





Step 3.2 : Visualising numerical data

In [28]:
```
def scatter(x,fig):
    plt.subplot(5,2,fig)
    plt.scatter(cars[x],cars['price'])
    plt.title(x+' vs Price')
    plt.ylabel('Price')
    plt.xlabel(x)

plt.figure(figsize=(10,20))

scatter('carlength', 1)
scatter('carwidth', 2)
scatter('carheight', 3)
scatter('curbweight', 4)

plt.tight_layout()
```

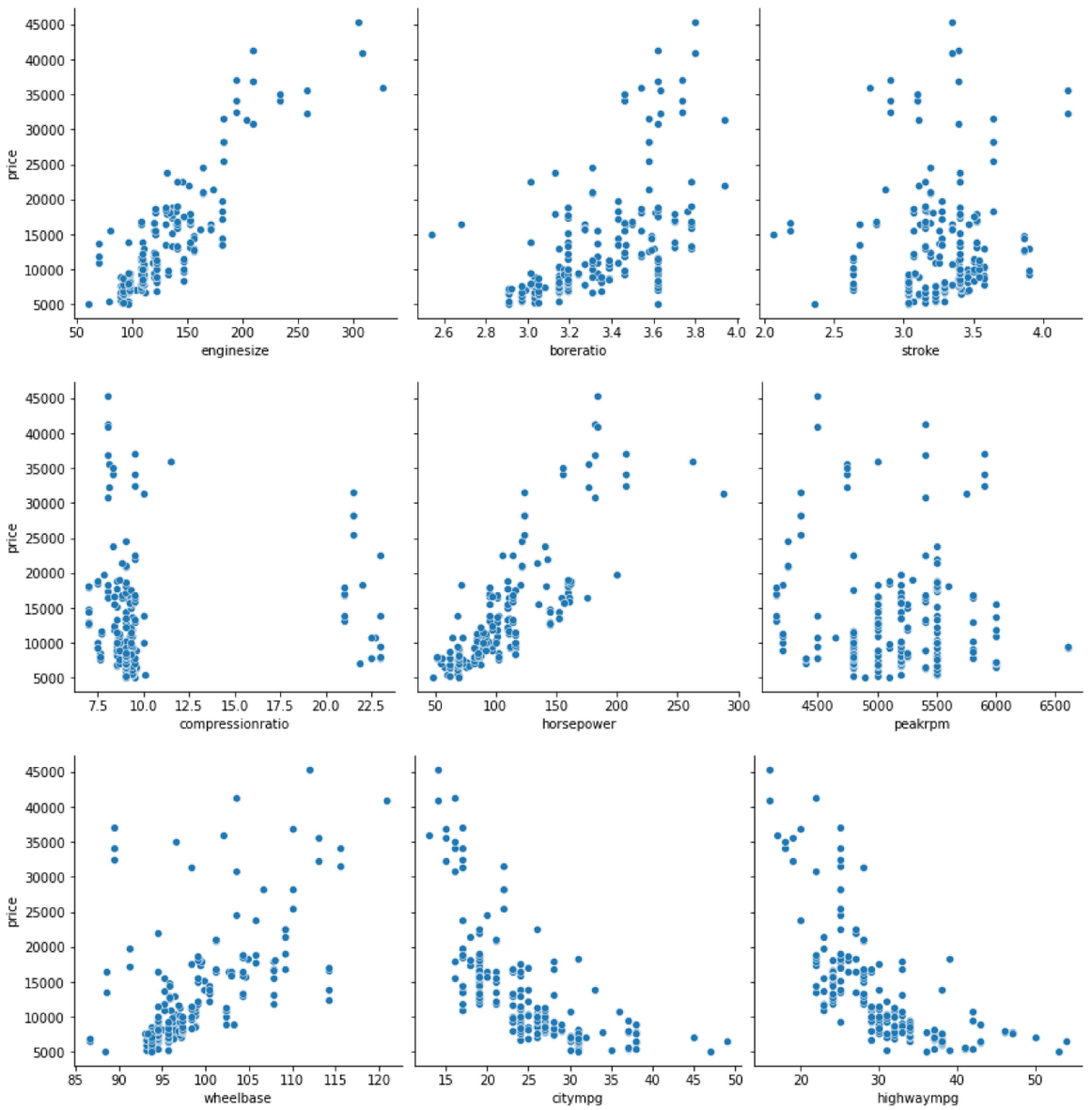


Inference :

1. `carwidth` , `carlength` and `curbweight` seems to have a poitive correlation with `price` .
2. `carheight` doesn't show any significant trend with price.

In [29]:
```
def pp(x,y,z):
    sns.pairplot(cars, x_vars=[x,y,z], y_vars='price',size=4, aspect=1, kind='scatter')
    plt.show()
```

```
pp('enginesize', 'boreratio', 'stroke')
pp('compressionratio', 'horsepower', 'peakrpm')
pp('wheelbase', 'citympg', 'highwaympg')
```



Inference :

1. `enginesize` , `boreratio` , `horsepower` , `wheelbase` - seem to have a significant positive correlation with price.
2. `citympg` , `highwaympg` - seem to have a significant negative correlation with price.

In [30]:
```
np.corrcoef(cars['carlength'], cars['carwidth'])[0, 1]
```

Out[30]:
```
0.841118268481846
```

Step 4 : Deriving new features

In [31]:
```
#Fuel economy
cars['fueleconomy'] = (0.55 * cars['citympg']) + (0.45 * cars['highwaympg'])
```

In [32]:
```
#Binning the Car Companies based on avg prices of each Company.
cars['price'] = cars['price'].astype('int')
temp = cars.copy()
table = temp.groupby(['CompanyName'])['price'].mean()
temp = temp.merge(table.reset_index(), how='left',on='CompanyName')
bins = [0,10000,20000,40000]
cars_bin=['Budget','Medium','Highend']
cars['carsrange'] = pd.cut(temp['price_y'],bins,right=False,labels=cars_bin)
cars.head()
```

Out[32]:

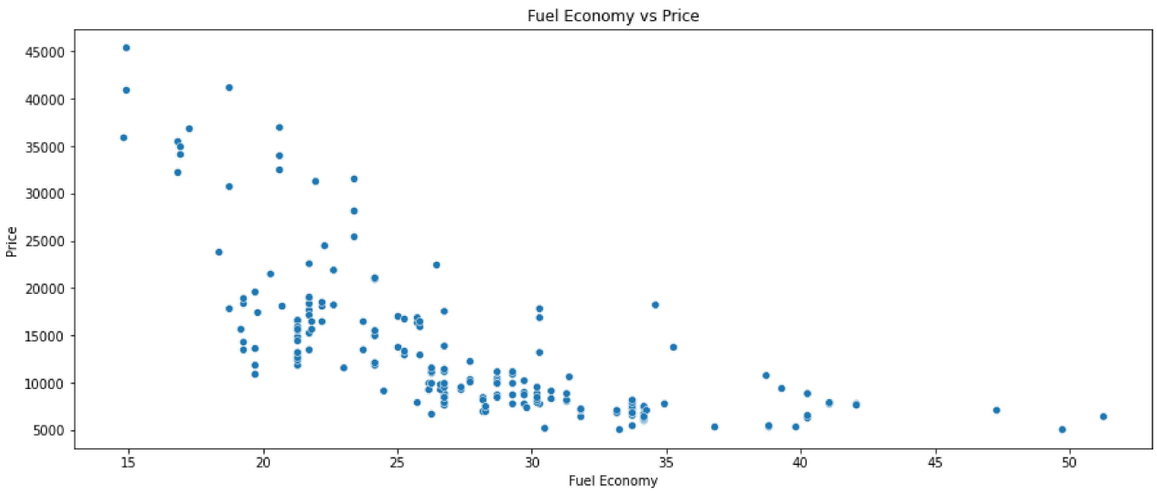
	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	engin
0	1	3	alfa-romero	gas	std	two	convertible	rwd	
1	2	3	alfa-romero	gas	std	two	convertible	rwd	

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	engin
2	3	1	alfa-romero	gas	std	two	hatchback	rwd	
3	4	2	audi	gas	std	four	sedan	fwd	
4	5	2	audi	gas	std	four	sedan	4wd	

5 rows × 28 columns

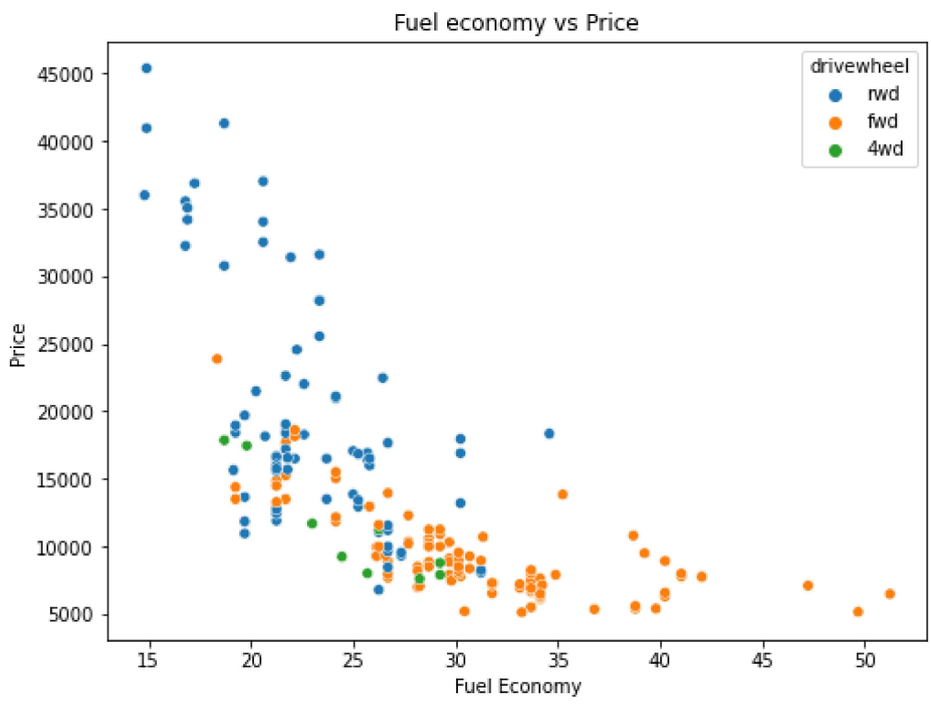
In [33]:
```
plt.figure(figsize = (15,6))
plt.title('Fuel Economy vs Price')
sns.scatterplot(x = cars['fueleconomy'], y = cars['price'])
plt.xlabel('Fuel Economy')
plt.ylabel('Price')
```

Out[33]:
```
Text(0, 0.5, 'Price')
```



Step 5 : Bivariate Analysis

In [34]:
```
plt.figure(figsize=(8,6))

plt.title('Fuel economy vs Price')
sns.scatterplot(x=cars['fueleconomy'],y=cars['price'],hue=cars['drivewheel'])
plt.xlabel('Fuel Economy')
plt.ylabel('Price')

plt.show()
plt.tight_layout()
```



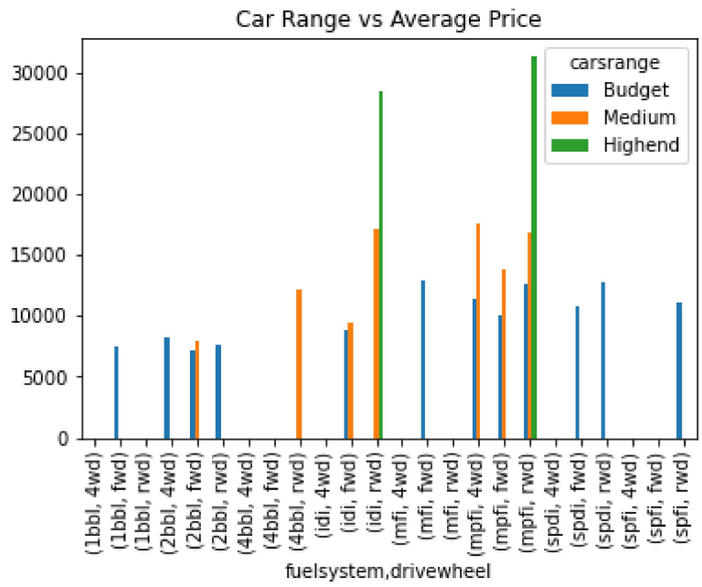

```
<Figure size 432x288 with 0 Axes>
```

Inference :

1. `fueleconomy` has an obvios `negative correlation` with price and is significant.

In [35]:
```
plt.figure(figsize=(25, 6))

df = pd.DataFrame(cars.groupby(['fuelsystem','drivewheel','carsrange'])['price'].mean()
df.plot.bar()
plt.title('Car Range vs Average Price')
plt.show()
```

```
<Figure size 1800x432 with 0 Axes>
```



Inference :

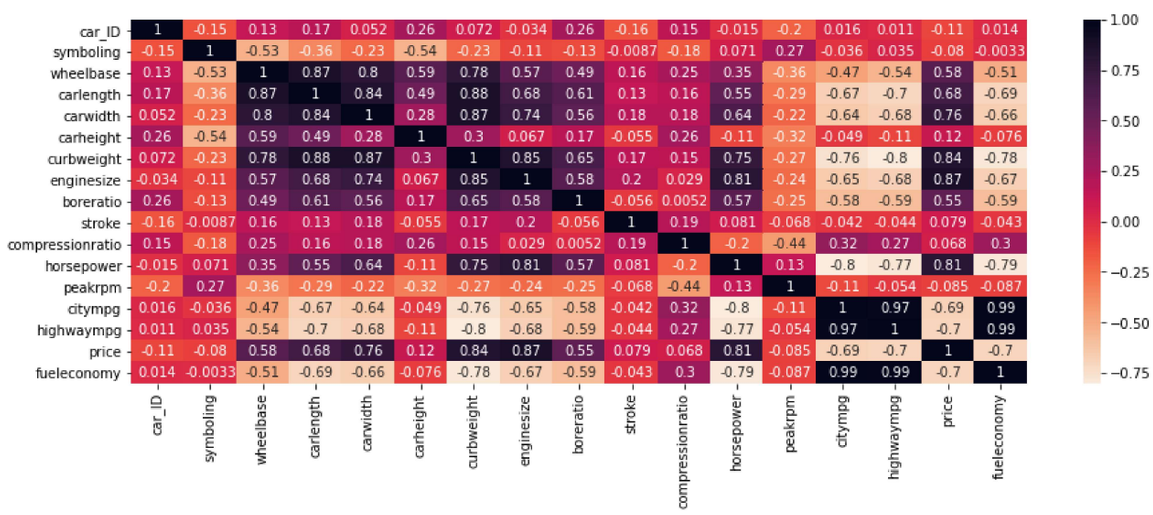
1. High ranged cars prefer `rwd` drivewheel with `idi` or `mpfi` fuelsystem.

List of significant variables after Visual analysis :

- Car Range
- Engine Type
- Fuel type
- Car Body
- Aspiration
- Cylinder Number
- Drivewheel
- Curbweight
- Car Length
- Car width
- Engine Size
- Boreratio
- Horse Power
- Wheel base
- Fuel Economy

In [36]:
```
plt.figure(figsize=(15,5))
sns.heatmap(cars.corr(), annot = True, cmap=sns.color_palette("rocket_r", as_cmap=True)
```

Out[36]:
```
<AxesSubplot:>
```



In [37]:
```
cars_lr = cars[['price', 'fueltype', 'aspiration','carbody', 'drivewheel','wheelbase',
                'curbweight', 'enginetype', 'cylindernumber', 'enginesize', 'borerati
                 'fueleconomy', 'carlength','carwidth', 'carsrange']]
cars_lr.head()
```

Out[37]:

	price	fueltype	aspiration	carbody	drivewheel	wheelbase	curbweight	enginetype	cylindernum
0	13495	gas	std	convertible	rwd	88.6	2548	dohc	
1	16500	gas	std	convertible	rwd	88.6	2548	dohc	
2	16500	gas	std	hatchback	rwd	94.5	2823	ohcv	
3	13950	gas	std	sedan	fwd	99.8	2337	ohc	
4	17450	gas	std	sedan	4wd	99.4	2824	ohc	

In [38]:
```
templs = ['price', 'fueltype', 'aspiration','carbody', 'drivewheel','wheelbase',
                'curbweight', 'enginetype', 'cylindernumber', 'enginesize', 'borerati
                 'fueleconomy', 'carlength','carwidth', 'carsrange']
tempcols = list(cars.columns)
for i in templs:
    tempcols.remove(i)
print("We have ignored these columns")
tempcols
```

```
We have ignored these columns
```

Out[38]:
```
['car_ID',
 'symboling',
 'CompanyName',
 'doornumber',
 'enginelocation',
 'carheight',
 'fuelsystem',
 'stroke',
 'compressionratio',
 'peakrpm',
 'citympg',
 'highwaympg']
```

In [39]:
```
cars.head()
```

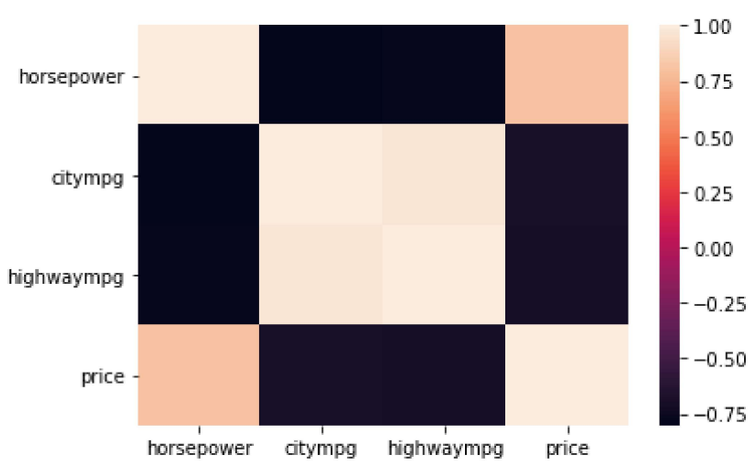
Out[39]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	engin
0	1	3	alfa-romero	gas	std	two	convertible	rwd	
1	2	3	alfa-romero	gas	std	two	convertible	rwd	
2	3	1	alfa-romero	gas	std	two	hatchback	rwd	
3	4	2	audi	gas	std	four	sedan	fwd	
4	5	2	audi	gas	std	four	sedan	4wd	

5 rows × 28 columns

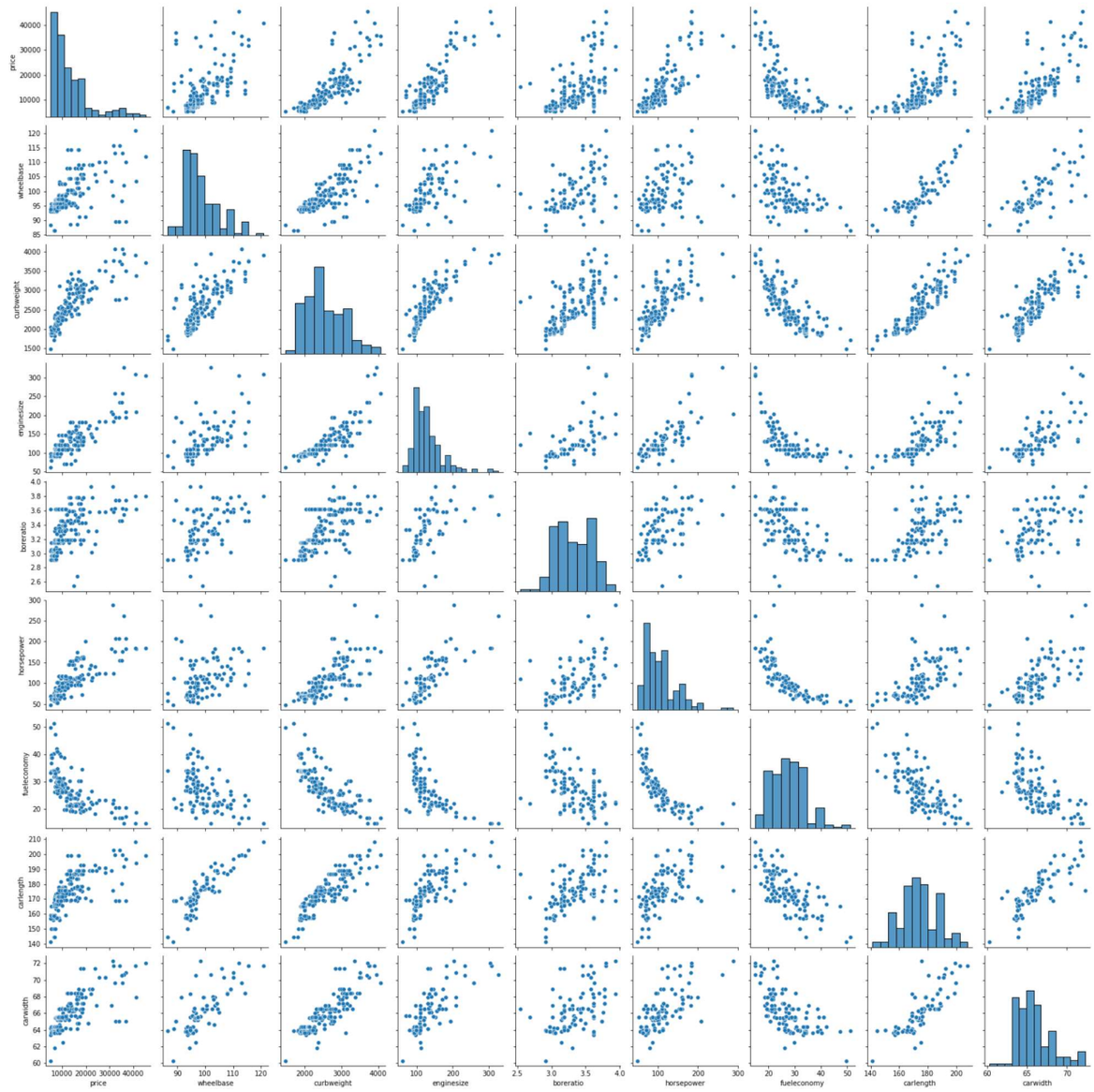
In [40]:
```
col=['horsepower','citympg','highwaympg','price']
sns.heatmap(cars[col].corr())
```

Out[40]:
```
<AxesSubplot:>
```



Step 6 : Dummy Variables

In [41]:
```
sns.pairplot(cars_lr)
plt.show()
```



In [42]:
```
# Defining the map function
def dummies(x,df):
    temp = pd.get_dummies(df[x], drop_first = True)
    df = pd.concat([df, temp], axis = 1)
    df.drop([x], axis = 1, inplace = True)
    return df
# Applying the function to the cars_lr

cars_lr = dummies('fueltype',cars_lr)
cars_lr = dummies('aspiration',cars_lr)
cars_lr = dummies('carbody',cars_lr)
cars_lr = dummies('drivewheel',cars_lr)
cars_lr = dummies('enginetype',cars_lr)
cars_lr = dummies('cylindernumber',cars_lr)
cars_lr = dummies('carsrange',cars_lr)

cars_lr.head()
```

Out[42]:

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	fueleconomy	carlength	carwidth
0	13495	88.6	2548	130	3.47	111	23.70	168.8	64.1
1	16500	88.6	2548	130	3.47	111	23.70	168.8	64.1
2	16500	94.5	2823	152	2.68	154	22.15	171.2	65.5
3	13950	99.8	2337	109	3.19	102	26.70	176.6	66.2
4	17450	99.4	2824	136	3.19	115	19.80	176.6	66.4

5 rows × 31 columns

In [43]:
```
cars_lr.shape
```

Out[43]:
```
(205, 31)
```

Step 7 : Train-Test Split and feature scaling

In [44]:
```
from sklearn.model_selection import train_test_split

np.random.seed(0)
df_train, df_test = train_test_split(cars_lr, train_size = 0.7, test_size = 0.3, random
```

In [45]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
num_vars = ['wheelbase', 'curbweight', 'enginesize', 'boreratio', 'horsepower','fueleco
df_train[num_vars] = scaler.fit_transform(df_train[num_vars])

df_train.head()
```

Out[45]:

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	fueleconomy	carlength	car
122	0.068818	0.244828	0.272692	0.139623	0.230159	0.083333	0.530864	0.426016	0.2
125	0.466890	0.272414	0.500388	0.339623	1.000000	0.395833	0.213992	0.452033	0.6
166	0.122110	0.272414	0.314973	0.139623	0.444444	0.266667	0.344307	0.448780	0.3
1	0.314446	0.068966	0.411171	0.260377	0.626984	0.262500	0.244170	0.450407	0.3
199	0.382131	0.610345	0.647401	0.260377	0.746032	0.475000	0.122085	0.775610	0.5

5 rows × 31 columns

In [46]:
```
df_train.describe()
```

Out[46]:

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	fueleconomy	carleng
count	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.0000
mean	0.219309	0.411141	0.407878	0.241351	0.497946	0.227302	0.358265	0.5254
std	0.215682	0.205581	0.211269	0.154619	0.207140	0.165511	0.185980	0.2048
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	0.067298	0.272414	0.245539	0.135849	0.305556	0.091667	0.198903	0.3991
50%	0.140343	0.341379	0.355702	0.184906	0.500000	0.191667	0.344307	0.5024
75%	0.313479	0.503448	0.559542	0.301887	0.682540	0.283333	0.512346	0.6699
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000

8 rows × 31 columns

Step 7 : Model Building

Highly correlated variables to price are - `curbweight` , `enginesize` , `horsepower` , `carwidth` and `highend` .

In [47]:
```
#Dividing data into X and y variables
y_train = df_train.pop('price')
X_train = df_train
```

LINEAR AND RANDOM MODEL

In [48]:
```
#RFE
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

Step 8 : Model Building

Feature Selection

In [49]:
```
cars.columns
```

Out[49]:
```
Index(['car_ID', 'symboling', 'CompanyName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price', 'fueleconomy', 'carsrange'],
      dtype='object')
```

In [50]:
```
numerical_cols=cars.select_dtypes(exclude=['object']).columns
X=cars[numerical_cols].drop('price',axis=1)
y=cars['price']
cars.head(2)
```

Out[50]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	engin
0	1	3	alfa-romero	gas	std	two	convertible	rwd	
1	2	3	alfa-romero	gas	std	two	convertible	rwd	

2 rows × 28 columns

Recursive feature elimination (RFE) with random forest

In [51]:
```
data = cars
X = data.apply(lambda col: preprocessing.LabelEncoder().fit_transform(col))
X=X.drop(['CompanyName','price'],axis=1)
y=data['price']
```

In [52]:
```
# Create the RFE object and rank each pixel
clf_rf_3 = RandomForestRegressor()
rfe = RFE(estimator=clf_rf_3, n_features_to_select=15, step=1)
rfe = rfe.fit(X, y)
print('Chosen best 15 feature by rfe:',X.columns[rfe.support_])
```

```
Chosen best 15 feature by rfe: Index(['car_ID', 'wheelbase', 'carlength', 'carwidth', 'c
arheight',
       'curbweight', 'enginesize', 'boreratio', 'compressionratio',
       'horsepower', 'peakrpm', 'citympg', 'highwaympg', 'fueleconomy',
       'carsrange'],
      dtype='object')
```

In [53]:
```
features=list(X.columns[rfe.support_])
```

Data Modelling and Evaluation

In [54]:
```
x = X[features]
y = data.price
x_train,x_test,y_train,y_test = train_test_split(x,y, random_state = 0)
```

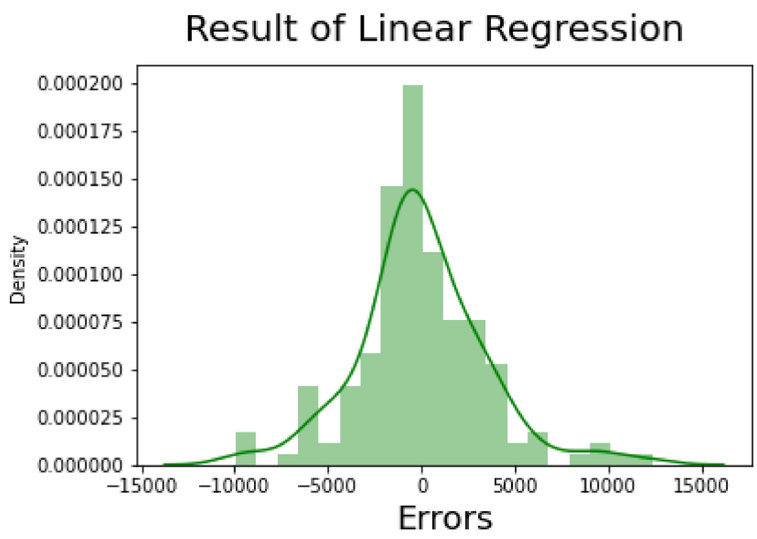
Linear Regression

In [55]:
```
lreg = LinearRegression()
lreg.fit(x_train,y_train)
y_train_pred = lreg.predict(x_train)
y_test_pred = lreg.predict(x_test)
lreg.score(x_test,y_test)
```

Out[55]: 0.7413223561624112

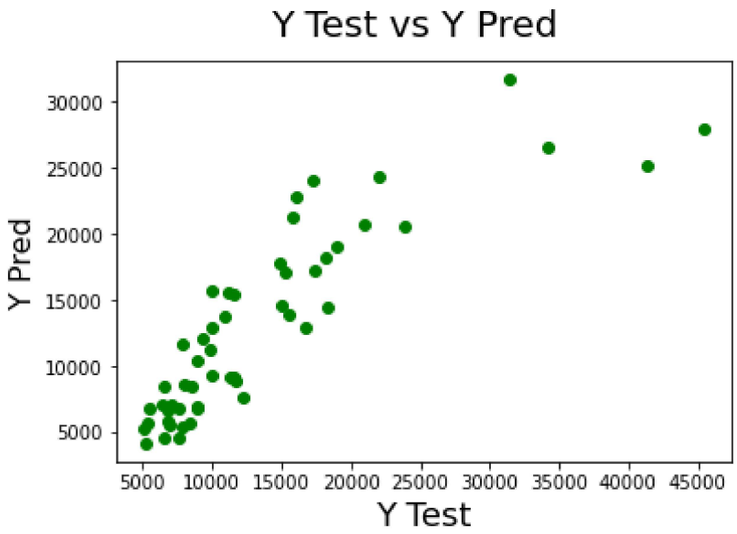
In [56]:
```
# Plot the histogram of the error terms
fig = plt.figure()
sns.distplot((y_train - y_train_pred), bins = 20,color='green')
fig.suptitle('Result of Linear Regression', fontsize = 20)                    # Plot head
plt.xlabel('Errors', fontsize = 18)
```

Out[56]: Text(0.5, 0, 'Errors')



In [57]:
```
#EVALUATION OF THE MODEL
# Plotting y_test and y_pred to understand the spread.
fig = plt.figure()
plt.scatter(y_test,y_test_pred,color='green')
fig.suptitle('Y Test vs Y Pred', fontsize=20)                  # Plot heading
plt.xlabel('Y Test', fontsize=18)                              # X-label
plt.ylabel('Y Pred', fontsize=16)
```

Out[57]: Text(0, 0.5, 'Y Pred')



Random Forest regressor

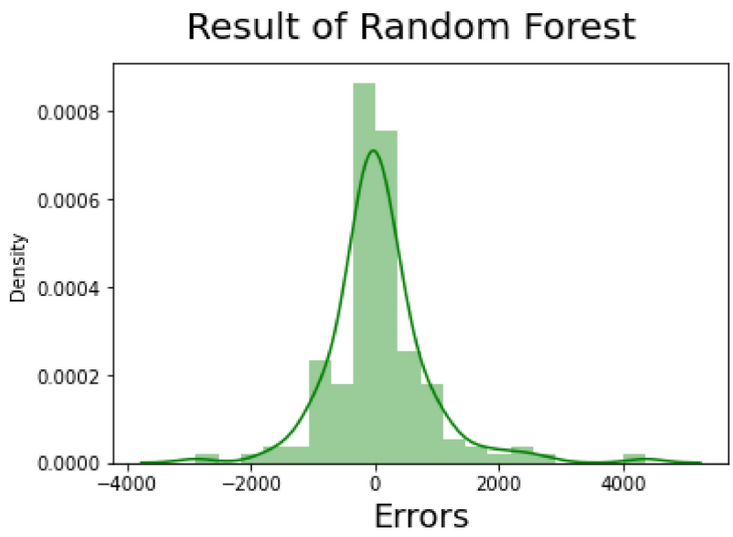
In [59]:
```
Rf = RandomForestRegressor(n_estimators = 15,
                           criterion = 'mse',
                           random_state = 20,
                           n_jobs = -1)
Rf.fit(x_train,y_train)
Rf_train_pred = Rf.predict(x_train)
Rf_test_pred = Rf.predict(x_test)

from sklearn.metrics import r2_score
r2_score(y_test,Rf_test_pred)
```

Out[59]: 0.9061630914758874

In [60]:
```
# Plot the histogram of the error terms
fig = plt.figure()
```

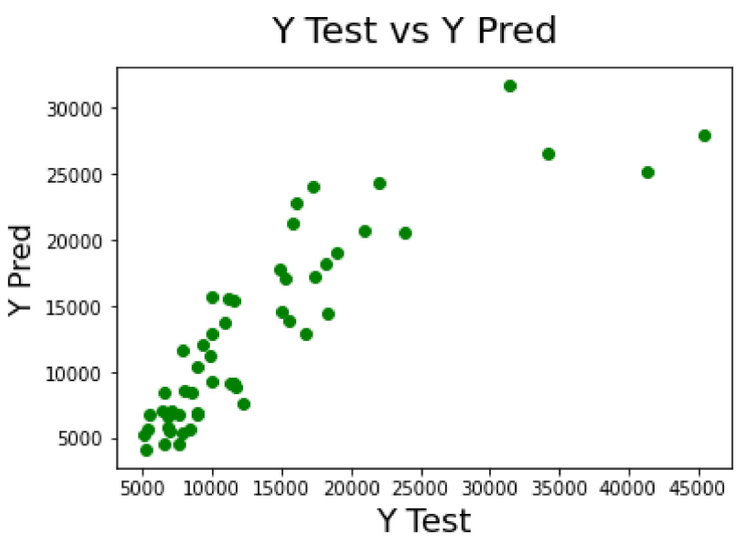
```
sns.distplot((y_train - Rf_train_pred), bins = 20,color='green')
fig.suptitle('Result of Random Forest', fontsize = 20)                    # Plot heading
plt.xlabel('Errors', fontsize = 18)
```

Out[60]: Text(0.5, 0, 'Errors')



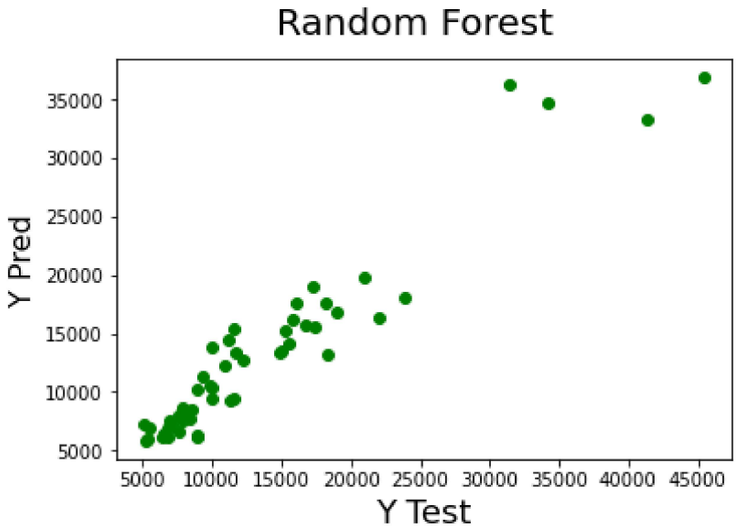
In [61]:
```
#EVALUATION OF THE MODEL
# Plotting y_test and y_pred to understand the spread.
fig = plt.figure()
plt.scatter(y_test,y_test_pred,color='green')
fig.suptitle('Y Test vs Y Pred', fontsize=20)                  # Plot heading
plt.xlabel('Y Test', fontsize=18)                              # X-label
plt.ylabel('Y Pred', fontsize=16)
```

Out[61]: Text(0, 0.5, 'Y Pred')



In [62]:
```
#EVALUATION OF THE MODEL
# Plotting y_test and y_pred to understand the spread.
fig = plt.figure()
plt.scatter(y_test,Rf_test_pred,color='green')
fig.suptitle('Random Forest', fontsize=20)                  # Plot heading
plt.xlabel('Y Test', fontsize=18)                              # X-label
plt.ylabel('Y Pred', fontsize=16)
```

Out[62]: Text(0, 0.5, 'Y Pred')


Random Forest
Y Pred
Y Test
35000
30000
25000
20000
15000
10000
5000
5000
10000
15000
20000
25000
30000
35000
40000
45000

In [ ]:

In [ ]: