Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
import seaborn as sns
import pylab
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
sns.set(style="darkgrid",font_scale=1.5)
pd.set_option("display.max.columns",None)
pd.set_option("display.max.rows",None)

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor,
from xgboost import XGBRegressor
from catboost import CatBoostRegressor
from lightgbm import LGBMRegressor

from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.preprocessing import StandardScaler
```

Loading datasets

In [2]:
```
df = pd.read_csv("CarPrice_Assignment.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	eng
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	

In [4]:
```
df.tail()
```

Out[4]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	engin
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	

In [5]:
```
df.shape
```

Out[5]:
```
(205, 26)
```

In [6]:
```
df.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
```

In [7]:
```
df.describe()
```

Out[7]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	er
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	20
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	12
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	4
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	6
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	9
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	12
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	14
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	32

In [8]:
```
df.isnull().sum().to_frame().rename(columns={0:"Total No. of Missing Values"})
```

Out[8]:

	Total No. of Missing Values
car_ID	0
symboling	0
CarName	0
fueltype	0
aspiration	0
doornumber	0
carbody	0
drivewheel	0
enginelocation	0
wheelbase	0
carlength	0
carwidth	0
carheight	0
curbweight	0
enginetype	0
cylindernumber	0
enginesize	0
fuelsystem	0
boreratio	0
stroke	0
compressionratio	0
horsepower	0
peakrpm	0
citympg	0
highwaympg	0
price	0

In [9]:
```
print("Duplicate Values =",df.duplicated().sum())
```

```
Duplicate Values = 0
```

In [10]:
```
df.select_dtypes(include="object").head()
```

Out[10]:

	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	enginety
0	alfa-romero giulia	gas	std	two	convertible	rwd	front	do
1	alfa-romero stelvio	gas	std	two	convertible	rwd	front	do
2	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	ol
3	audi 100 ls	gas	std	four	sedan	fwd	front	
4	audi 100ls	gas	std	four	sedan	4wd	front	

In [11]:
```
df.select_dtypes(include=["int","float"]).head()
```

Out[11]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	borera
0	1	3	88.6	168.8	64.1	48.8	2548	130	3.
1	2	3	88.6	168.8	64.1	48.8	2548	130	3.
2	3	1	94.5	171.2	65.5	52.4	2823	152	2.
3	4	2	99.8	176.6	66.2	54.3	2337	109	3.
4	5	2	99.4	176.6	66.4	54.3	2824	136	3.

Data Cleaning

In [12]:
```
df.head()
```

Out[12]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	eng
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	

In [13]:
```
Company_Name = df["CarName"].apply(lambda x: x.split(" ")[0])
df.insert(2,"CompanyName",Company_Name)

# Now we can drop the CarName Feature.
df.drop(columns=["CarName"],inplace=True)
```

In [14]:
```
df.head()
```

Out[14]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel
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

In [15]:
```
df["CompanyName"].unique()
```

Out[15]:
```
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)
```

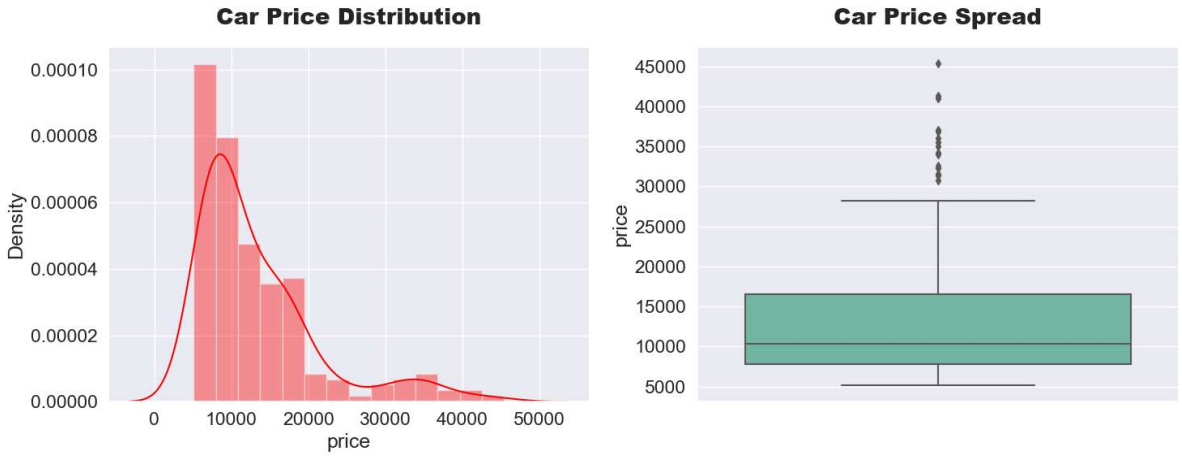
In [16]:
```
def replace(a,b):
    df["CompanyName"].replace(a,b,inplace=True)

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

In [17]:
```
df["CompanyName"].unique()
```

Out[17]:
```
array(['alfa-romero', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda',
       'isuzu', 'jaguar', 'mazda', 'buick', 'mercury', 'mitsubishi',
       'Nissan', 'nissan', 'peugeot', 'plymouth', 'porsche', 'renault',
       'saab', 'subaru', 'toyota', 'volkswagen', 'volvo'], dtype=object)
```

Exploratory Data Analysis(EDA)

In [18]:
```
plt.figure(figsize=(15,6))
plt.subplot(1,2,1)
sns.distplot(df["price"],color="red",kde=True)
plt.title("Car Price Distribution",fontweight="black",pad=20,fontsize=20)

plt.subplot(1,2,2)
sns.boxplot(y=df["price"],palette="Set2")
plt.title("Car Price Spread",fontweight="black",pad=20,fontsize=20)
plt.tight_layout()
plt.show()
```

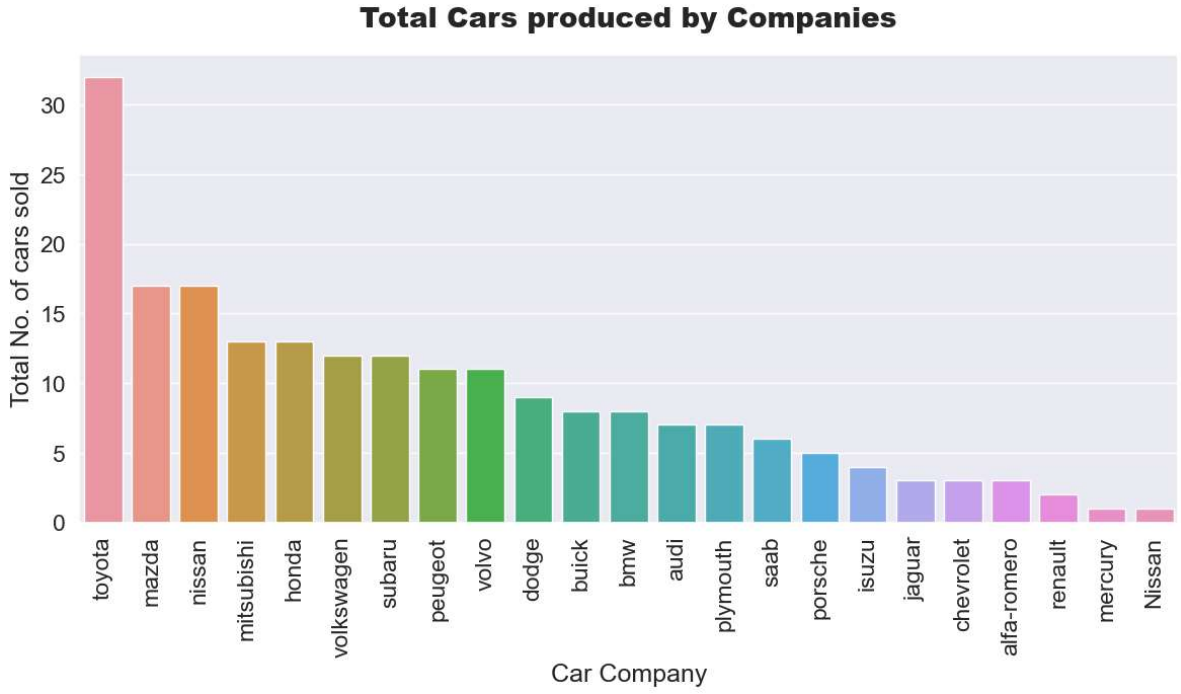


In [19]:
```
df["price"].agg(["min","mean","median","max","std","skew"]).to_frame().T
```

Out[19]:

	min	mean	median	max	std	skew
price	5118.0	13276.710571	10295.0	45400.0	7988.852332	1.777678

In [20]:
```
plt.figure(figsize=(14,6))
counts = df["CompanyName"].value_counts()
sns.barplot(x=counts.index, y=counts.values)
plt.xlabel("Car Company")
plt.ylabel("Total No. of cars sold")
plt.title("Total Cars produced by Companies", pad=20, fontweight="black", font
plt.xticks(rotation=90)
plt.show()
```



In [21]:
```
df[df["CompanyName"]=="mercury"]
```

Out[21]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel
75	76	1	mercury	gas	turbo	two	hatchback	rwd

In [22]:
```
df[df["CompanyName"]=="Nissan"]
```

Out[22]:

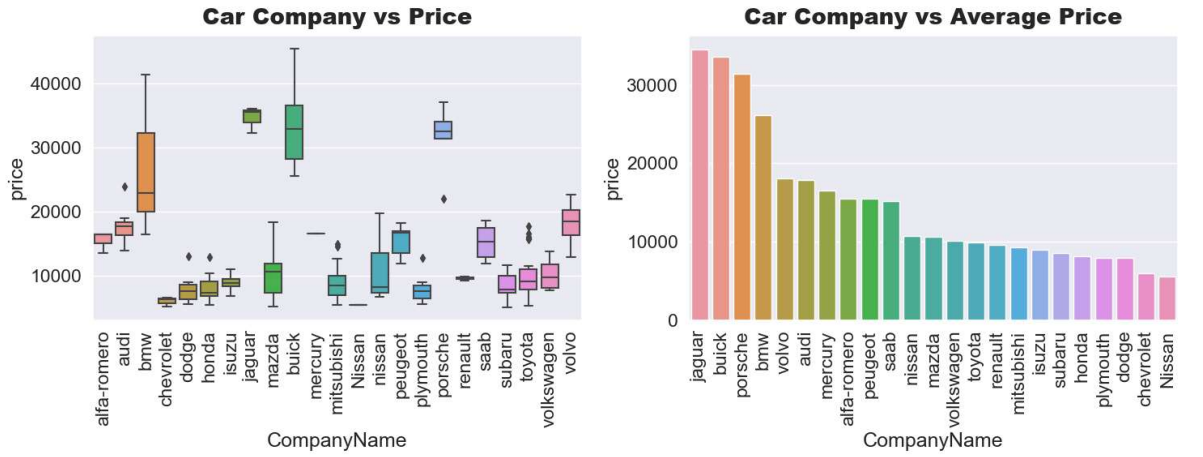
	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	e
89	90	1	Nissan	gas	std	two	sedan	fwd	

In [23]:
```
df[df["CompanyName"]=="renault"]
```

Out[23]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel
130	131	0	renault	gas	std	four	wagon	fwd
131	132	2	renault	gas	std	two	hatchback	fwd

In [24]:
```
plt.figure(figsize=(15,6))

plt.subplot(1,2,1)
sns.boxplot(x="CompanyName",y="price",data=df)
plt.xticks(rotation=90)
plt.title("Car Company vs Price", pad=10, fontweight="black", fontsize=20)

plt.subplot(1,2,2)
x = pd.DataFrame(df.groupby("CompanyName")["price"].mean().sort_values(ascendi
sns.barplot(x=x.index,y="price",data=x)
plt.xticks(rotation=90)
plt.title("Car Company vs Average Price", pad=10, fontweight="black", fontsize
plt.tight_layout()
plt.show()
```



In [25]:
```
df[df["CompanyName"]=="mercury"]
```

Out[25]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel
75	76	1	mercury	gas	turbo	two	hatchback	rwd

In [26]:
```
df[df["CompanyName"]=="Nissan"]
```

Out[26]:

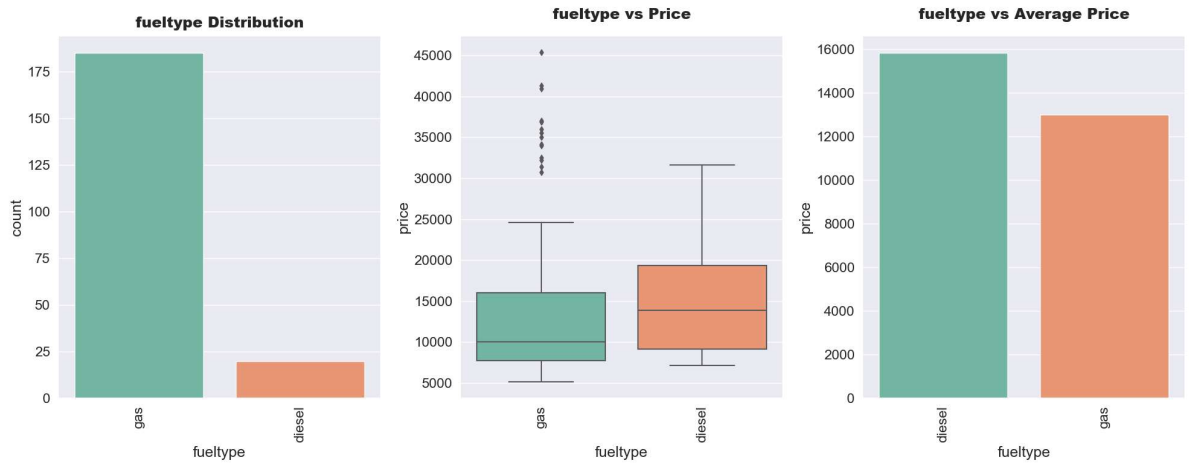
	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	e
89	90	1	Nissan	gas	std	two	sedan	fwd	

In [27]:
```
df[df["CompanyName"]=="renault"]
```

Out[27]:

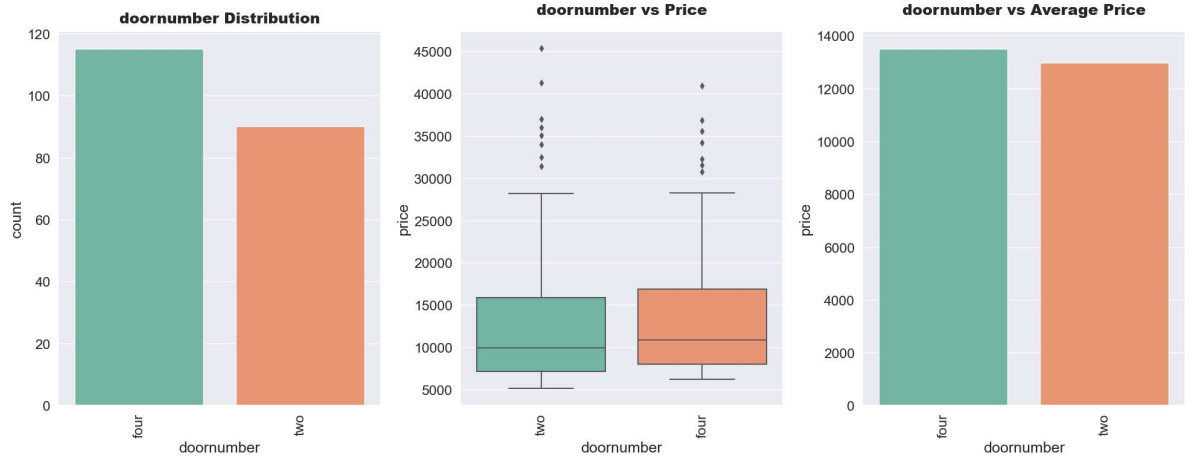
	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel
130	131	0	renault	gas	std	four	wagon	fwd
131	132	2	renault	gas	std	two	hatchback	fwd

In [28]:
```
def categorical_visualization(cols):
    plt.figure(figsize=(20,8))
    plt.subplot(1,3,1)
    sns.countplot(x=cols,data=df,palette="Set2",order=df[cols].value_counts().
    plt.title(f"{cols} Distribution",pad=10,fontweight="black",fontsize=18)
    plt.xticks(rotation=90)

    plt.subplot(1,3,2)
    sns.boxplot(x=cols,y="price",data=df,palette="Set2")
    plt.title(f"{cols} vs Price",pad=20,fontweight="black",fontsize=18)
    plt.xticks(rotation=90)

    plt.subplot(1,3,3)
    x=pd.DataFrame(df.groupby(cols)["price"].mean().sort_values(ascending=Fals
    sns.barplot(x=x.index,y="price",data=x,palette="Set2")
    plt.title(f"{cols} vs Average Price",pad=20,fontweight="black",fontsize=18
    plt.xticks(rotation=90)
    plt.tight_layout()
    plt.show()

categorical_visualization("fueltype")
```

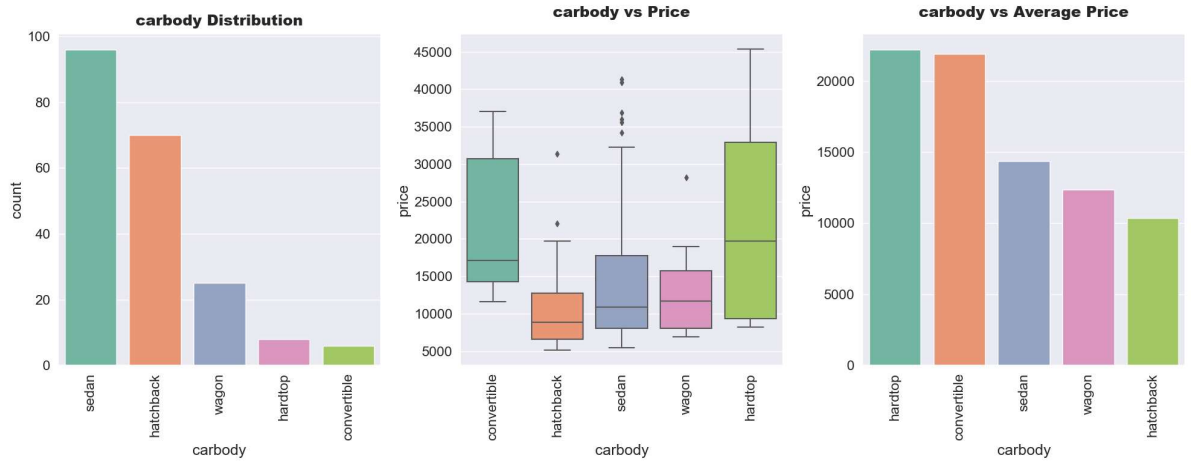


In [29]:
```
categorical_visualization("aspiration")
```

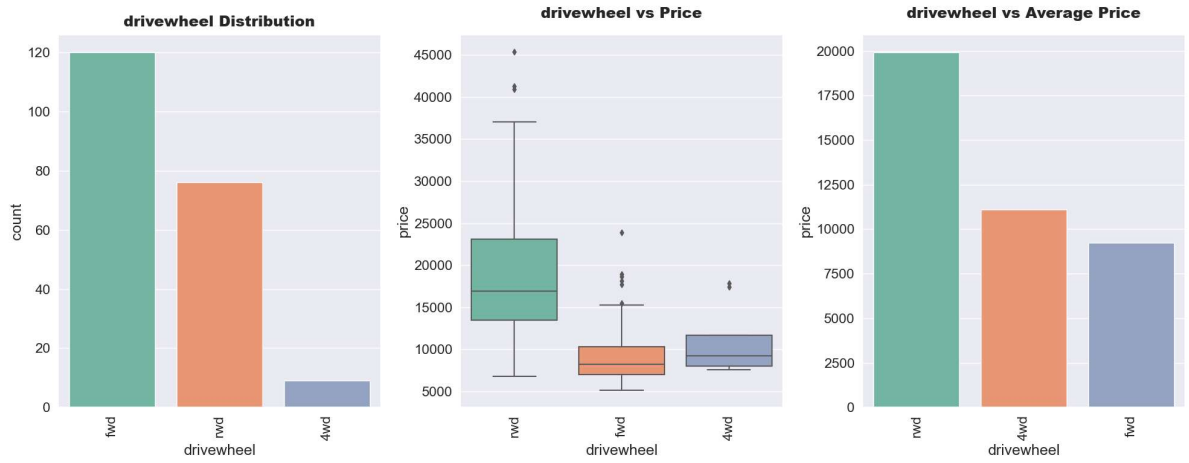
In [30]:
```
categorical_visualization("doornumber")
```



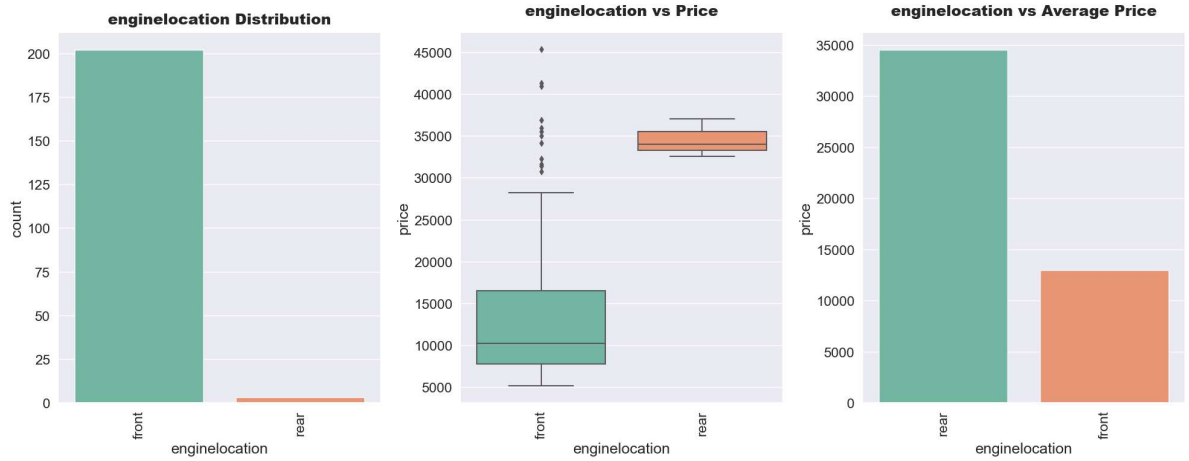
In [31]:
```
categorical_visualization("carbody")
```



In [32]:
```
categorical_visualization("drivewheel")
```



In [33]:
```
categorical_visualization("enginelocation")
```

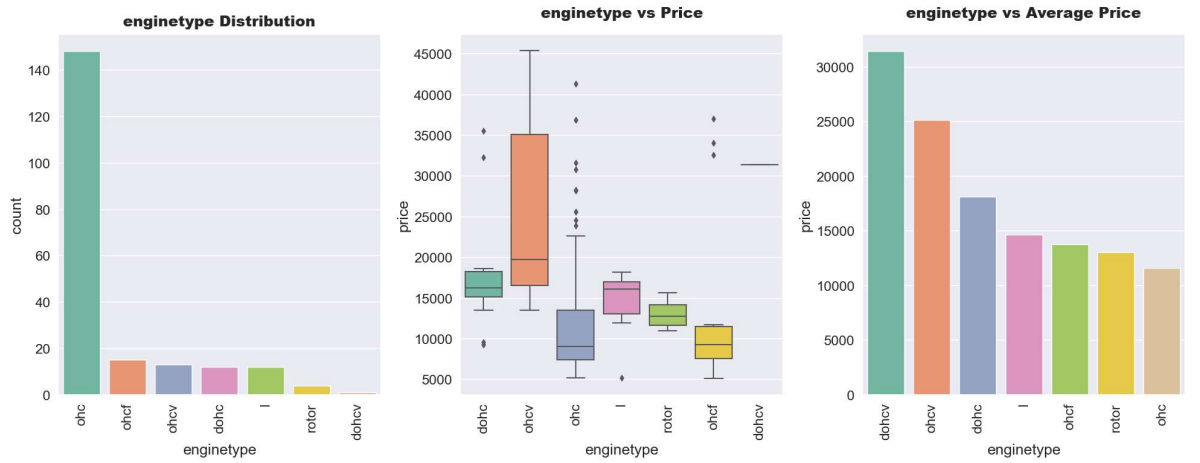


In [34]:
```
df[df["enginelocation"]=="rear"]
```

Out[34]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel
126	127	3	porsche	gas	std	two	hardtop	rwd
127	128	3	porsche	gas	std	two	hardtop	rwd
128	129	3	porsche	gas	std	two	convertible	rwd

In [35]:
```
categorical_visualization("enginetype")
```



In [36]:
```
df[df["enginetype"]=="dohcv"]
```

Out[36]:

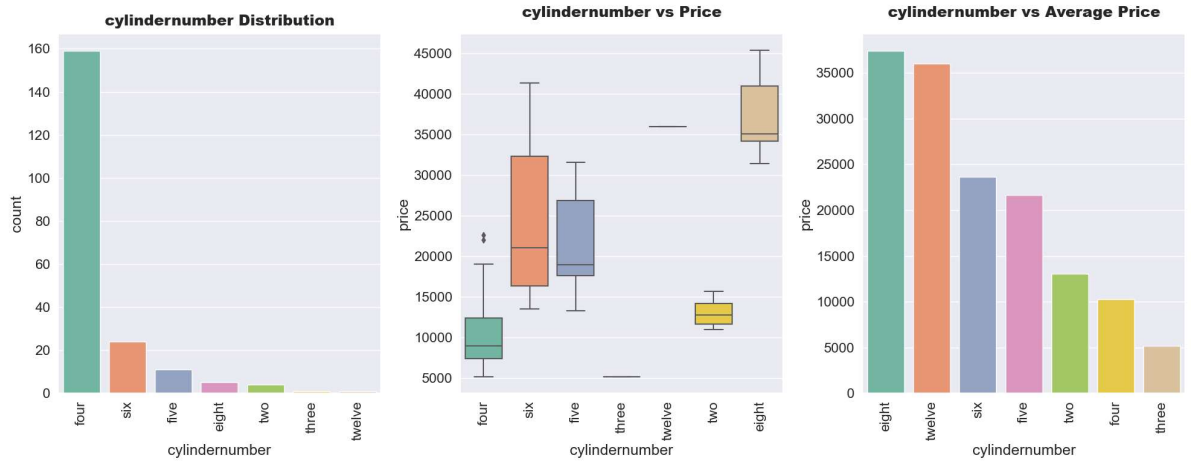
	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel
129	130	1	porsche	gas	std	two	hatchback	rwd

In [37]:
```
df[df["enginetype"]=="rotor"]
```

Out[37]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel
55	56	3	mazda	gas	std	two	hatchback	rwd
56	57	3	mazda	gas	std	two	hatchback	rwd
57	58	3	mazda	gas	std	two	hatchback	rwd
58	59	3	mazda	gas	std	two	hatchback	rwd

In [38]:
```
categorical_visualization("cylindernumber")
```



In [39]:
```
df[df["cylindernumber"]=="three"]
```

Out[39]:

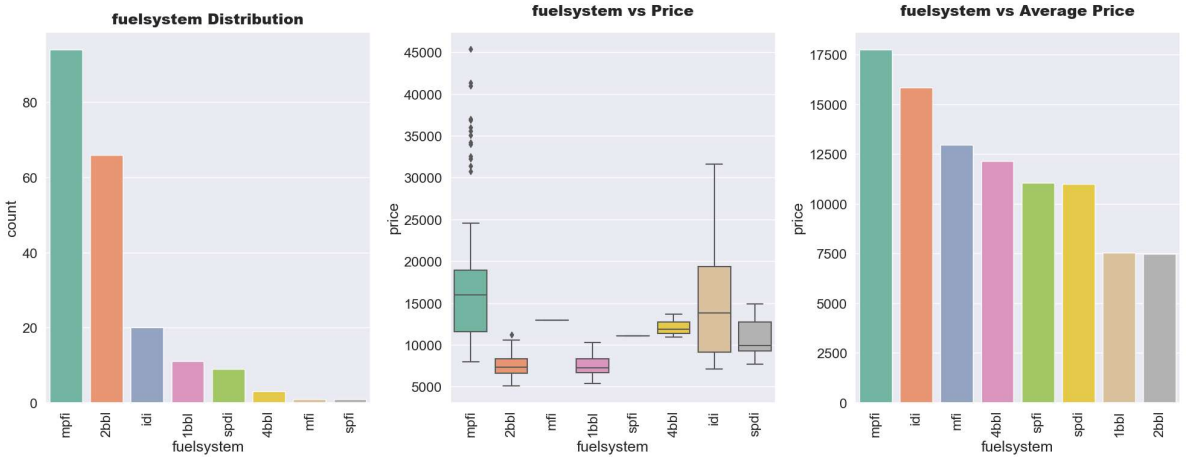
	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel
18	19	2	chevrolet	gas	std	two	hatchback	fwd

In [40]:
```
df[df["cylindernumber"]=="twelve"]
```

Out[40]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	e
49	50	0	jaguar	gas	std	two	sedan	rwd	

In [41]:
```
categorical_visualization("fuelsystem")
```



In [42]:
```
df[df["fuelsystem"]=="mfi"]
```

Out[42]:

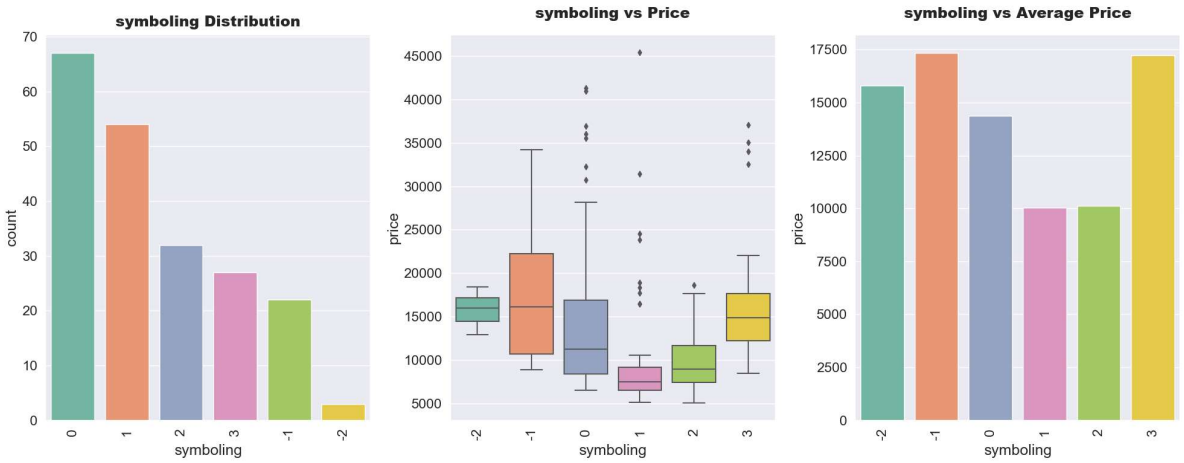
	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel
29	30	3	dodge	gas	turbo	two	hatchback	fwd

In [43]:
```
df[df["fuelsystem"]=="spfi"]
```

Out[43]:

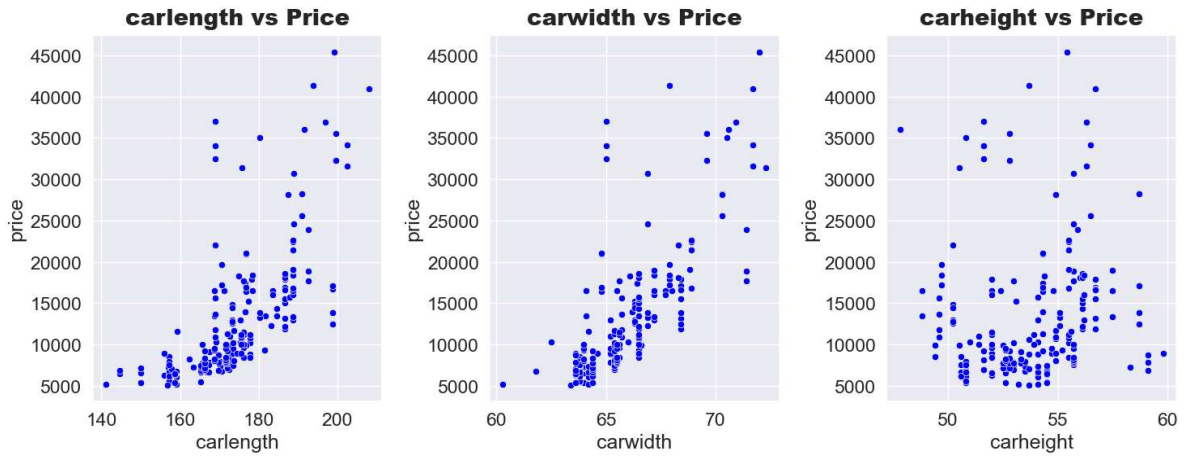
	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel
46	47	2	isuzu	gas	std	two	hatchback	rwd

In [44]:
```
categorical_visualization("symboling")
```

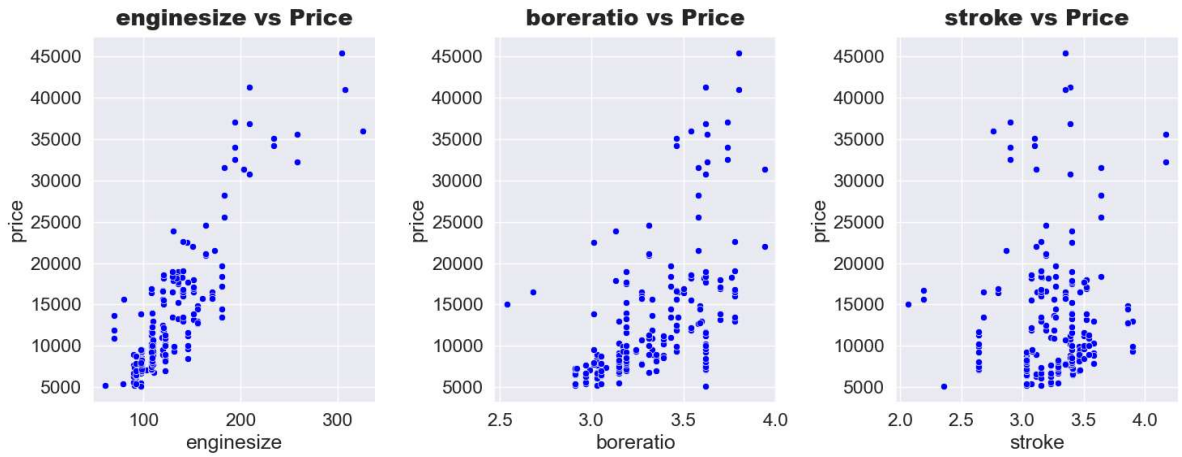


In [45]:
```
def scatter_plot(cols):
    x=1
    plt.figure(figsize=(15,6))
    for col in cols:
        plt.subplot(1,3,x)
        sns.scatterplot(x=col,y="price",data=df,color="blue")
        plt.title(f"{col} vs Price",fontweight="black",fontsize=20,pad=10)
        plt.tight_layout()
        x+=1
```

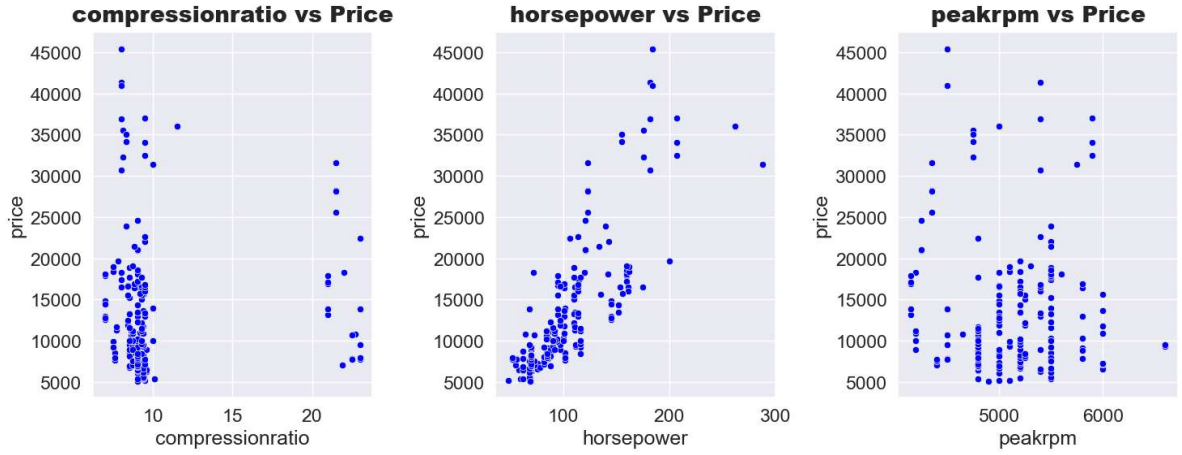
In [46]:
```
scatter_plot(["carlength","carwidth","carheight"])
```



In [47]:
```
scatter_plot(["enginesize","boreratio","stroke"])
```

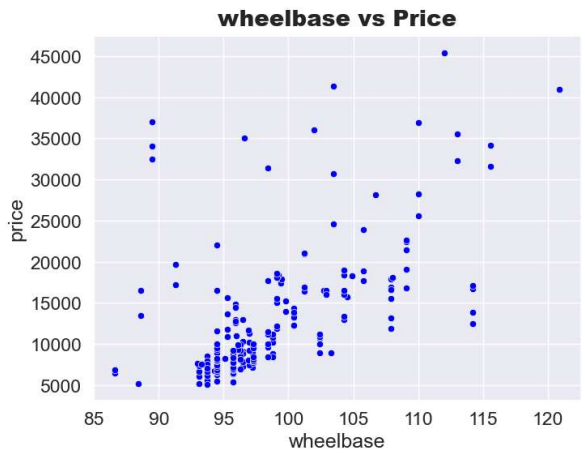


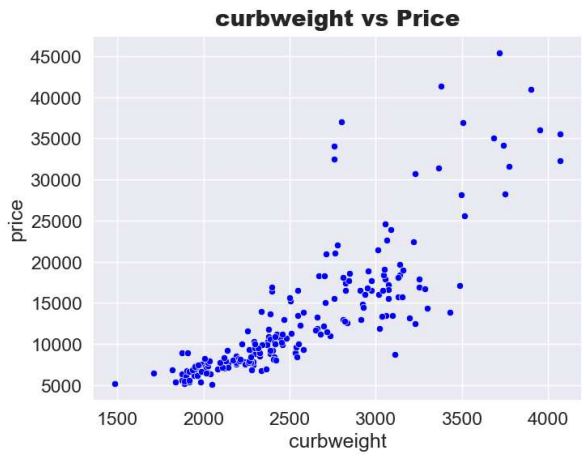
In [48]:
```
scatter_plot(["compressionratio","horsepower","peakrpm"])
```



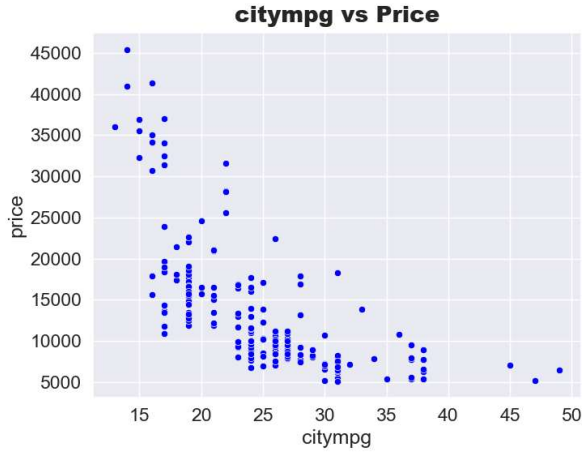
In [49]:
```
def scatter_plot(cols):
    x=1
    plt.figure(figsize=(15,6))
    for col in cols:
        plt.subplot(1,2,x)
        sns.scatterplot(x=col,y="price",data=df,color="blue")
        plt.title(f"{col} vs Price",fontweight="black",fontsize=20,pad=10)
        plt.tight_layout()
        x+=1
```

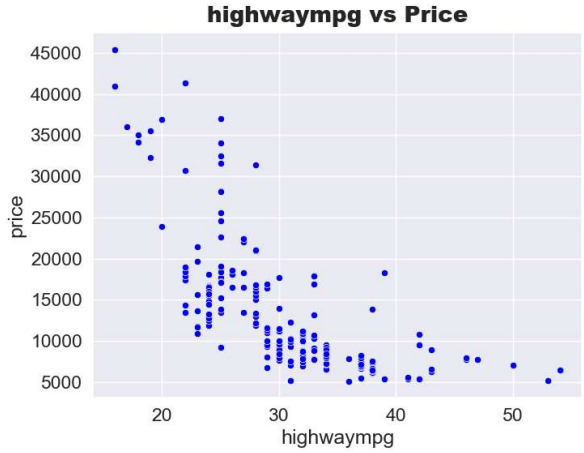
In [50]:
```
scatter_plot(["wheelbase","curbweight"])
```





In [51]:
```
scatter_plot(["citympg","highwaympg"])
```





Feature Engineering

In [52]:
```
z = round(df.groupby(["CompanyName"])["price"].agg(["mean"]),2).T
z
```

Out[52]:

CompanyName	Nissan	alfa-romero	audi	bmw	buick	chevrolet	dodge	honda	isu
mean	5499.0	15498.33	17859.17	26118.75	33647.0	6007.0	7875.44	8184.69	8916

In [53]:
```
df = df.merge(z.T,how="left",on="CompanyName")
```

In [54]:
```
bins = [0,10000,20000,40000]
cars_bin=['Budget','Medium','Highend']
df['CarsRange'] = pd.cut(df['mean'],bins,right=False,labels=cars_bin)
df.head()
```

Out[54]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel
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

Data Preprocessing

In [55]:
```
new_df = df[['fueltype','aspiration','doornumber','carbody','drivewheel','engi
             ,'wheelbase','carlength','carwidth','curbweight','enginesize','bo
             'price','CarsRange']]
```

In [56]:
```
new_df.head()
```

Out[56]:

	fueltype	aspiration	doornumber	carbody	drivewheel	enginetype	cylindernumber	fuelsys
0	gas	std	two	convertible	rwd	dohc	four	
1	gas	std	two	convertible	rwd	dohc	four	
2	gas	std	two	hatchback	rwd	ohcv	six	
3	gas	std	four	sedan	fwd	ohc	four	
4	gas	std	four	sedan	4wd	ohc	five	

In [57]:
```
new_df = pd.get_dummies(columns=["fueltype","aspiration","doornumber","carbody
                                 "cylindernumber","fuelsystem","CarsRange"],dat
```

In [58]:
```
new_df.head()
```

Out[58]:

	wheelbase	carlength	carwidth	curbweight	enginesize	boreratio	horsepower	citympg	high
0	88.6	168.8	64.1	2548	130	3.47	111	21	
1	88.6	168.8	64.1	2548	130	3.47	111	21	
2	94.5	171.2	65.5	2823	152	2.68	154	19	
3	99.8	176.6	66.2	2337	109	3.19	102	24	
4	99.4	176.6	66.4	2824	136	3.19	115	18	

In [59]:
```
scaler = StandardScaler()
```

In [60]:
```
num_cols = ['wheelbase','carlength','carwidth','curbweight','enginesize','bore
            'citympg','highwaympg']

new_df[num_cols] = scaler.fit_transform(new_df[num_cols])
```

In [61]:
```
new_df.head()
```

Out[61]:

	wheelbase	carlength	carwidth	curbweight	enginesize	boreratio	horsepower	citympg	h
0	-1.690772	-0.426521	-0.844782	-0.014566	0.074449	0.519071	0.174483	-0.646553	
1	-1.690772	-0.426521	-0.844782	-0.014566	0.074449	0.519071	0.174483	-0.646553	
2	-0.708596	-0.231513	-0.190566	0.514882	0.604046	-2.404880	1.264536	-0.953012	
3	0.173698	0.207256	0.136542	-0.420797	-0.431076	-0.517266	-0.053668	-0.186865	
4	0.107110	0.207256	0.230001	0.516807	0.218885	-0.517266	0.275883	-1.106241	

In [62]:
```
x = new_df.drop(columns=["price"])
y = new_df["price"]
```

In [63]:
```
x.shape
```

Out[63]:
```
(205, 48)
```

In [64]:
```
y.shape
```

Out[64]:
```
(205,)
```

In [65]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=
```

In [66]:
```
print("x_train - >  ",x_train.shape)
print("x_test - >  ",x_test.shape)
print("y_train - >  ",y_train.shape)
print("y_test - >  ",y_test.shape)
```

```
x_train - >   (164, 48)
x_test - >   (41, 48)
y_train - >   (164,)
y_test - >   (41,)
```

Model Building

In [67]:
```
training_score = []
testing_score = []
```

In [68]:
```
def model_prediction(model):
    model.fit(x_train,y_train)
    x_train_pred = model.predict(x_train)
    x_test_pred = model.predict(x_test)
    a = r2_score(y_train,x_train_pred)*100
    b = r2_score(y_test,x_test_pred)*100
    training_score.append(a)
    testing_score.append(b)

    print(f"r2_Score of {model} model on Training Data is:",a)
    print(f"r2_Score of {model} model on Testing Data is:",b)
```

1.Linear-Regression Model

In [69]:
```
model_prediction(LinearRegression())
```

```
r2_Score of LinearRegression() model on Training Data is: 96.0384799952984
r2_Score of LinearRegression() model on Testing Data is: 88.40540860813918
```

2.Decision-Tree-Regressor Model

In [70]:
```
model_prediction(RandomForestRegressor())
```

```
r2_Score of RandomForestRegressor() model on Training Data is: 98.78953311167
368
r2_Score of RandomForestRegressor() model on Testing Data is: 95.737753499172
61
```

3.Random-Forest-Regressor Model

In [71]:
```
model_prediction(DecisionTreeRegressor())
```

```
r2_Score of DecisionTreeRegressor() model on Training Data is: 99.86537119069
865
r2_Score of DecisionTreeRegressor() model on Testing Data is: 89.233911843990
91
```

4.Ada-Boost-Regressor Model

In [72]:
```
model_prediction(AdaBoostRegressor())
```

```
r2_Score of AdaBoostRegressor() model on Training Data is: 96.08634222061683
r2_Score of AdaBoostRegressor() model on Testing Data is: 91.31432552629506
```

5.Gradient-Boosting-Regressor Model

In [73]:
```
model_prediction(GradientBoostingRegressor())
```

```
r2_Score of GradientBoostingRegressor() model on Training Data is: 99.3723730
9193024
r2_Score of GradientBoostingRegressor() model on Testing Data is: 92.37821321
297834
```

6. LGMB Regressor Model

In [74]:
```
model_prediction(LGBMRegressor())
```

```
r2_Score of LGBMRegressor() model on Training Data is: 96.22424829025965
r2_Score of LGBMRegressor() model on Testing Data is: 88.73966577207719
```

7.XGBRegressor Model

In [75]:
```
model_prediction(XGBRegressor())
```

```
r2_Score of XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...) model on Training Data i
s: 99.86524431746504
r2_Score of XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...) model on Testing Data i
s: 93.01118352713648
```

8.Cat-Boost-Regressor Model

In [76]:
```
model_prediction(CatBoostRegressor(verbose=False))
```

```
r2_Score of <catboost.core.CatBoostRegressor object at 0x000001D4AEAE11E0> mo
del on Training Data is: 99.7086556035143
r2_Score of <catboost.core.CatBoostRegressor object at 0x000001D4AEAE11E0> mo
del on Testing Data is: 94.01252203857477
```

All Model Performance Comparison

In [77]:
```
models = ["Linear Regression","Decision Tree","Random Forest","Ada Boost","Gra
```

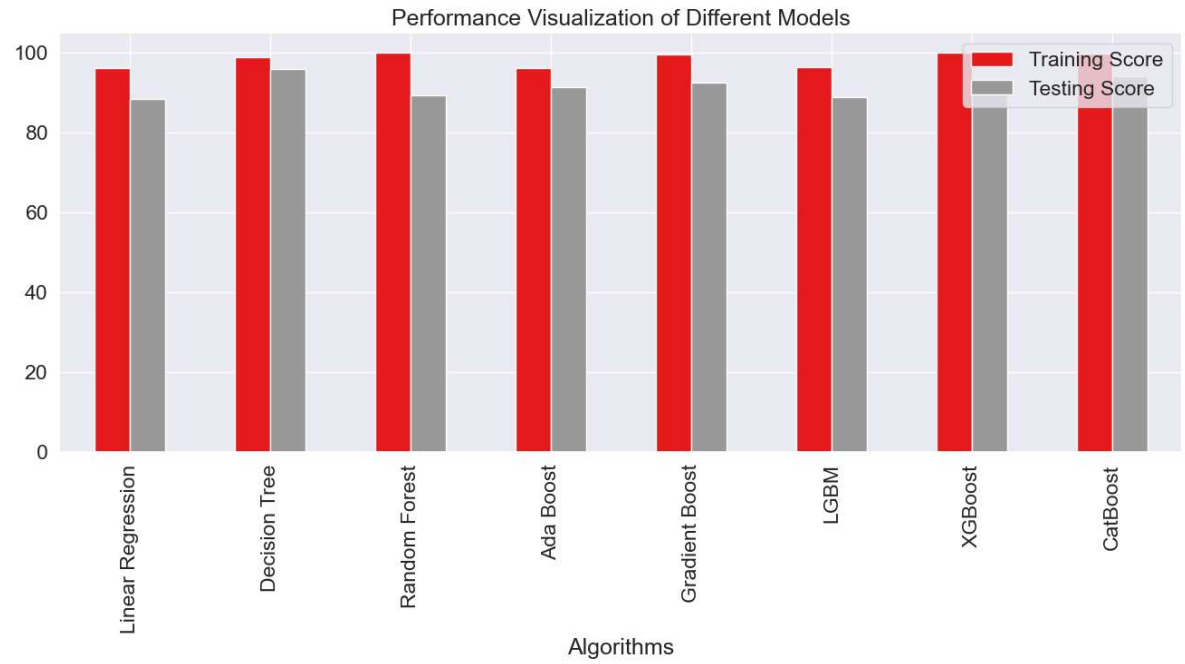
In [78]:
```
df = pd.DataFrame({"Algorithms":models,
                   "Training Score":training_score,
                   "Testing Score":testing_score})
```

In [79]:
```
df
```

Out[79]:

	Algorithms	Training Score	Testing Score
0	Linear Regression	96.038480	88.405409
1	Decision Tree	98.789533	95.737753
2	Random Forest	99.865371	89.233912
3	Ada Boost	96.086342	91.314326
4	Gradient Boost	99.372373	92.378213
5	LGBM	96.224248	88.739666
6	XGBoost	99.865244	93.011184
7	CatBoost	99.708656	94.012522

In [80]:
```
df.plot(x="Algorithms",y=["Training Score","Testing Score"], figsize=(16,6),ki
        title="Performance Visualization of Different Models",colormap="Set1")
plt.show()
```



In [ ]:
```

```