🔹 🪐 Car Price Prediction 🪐 🔹

In [15]:
```python
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, AdaBoostRegressor

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

In [16]:
```python
import os
working_directory = os.getcwd()
print (working_directory)
```

```
/Users/ishu/Desktop
```

In [25]:
```python
path = working_directory + '/car.csv'
df = pd.read_csv (path)
df.head()
```

Out[25]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth	carheight	curbweight	enginetype	cylindernumber	enginesize	fuelsyste
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130	mp
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130	mp
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4	2823	ohcv	six	152	mp
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3	2337	ohc	four	109	mp
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3	2824	ohc	five	136	mp

In [26]:
```python
df.tail()
```

Out[26]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth	carheight	curbweight	enginetype	cylindernumber	enginesize	fuelsystem
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	109.1	188.8	68.9	55.5	2952	ohc	four	141	mpfi
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	109.1	188.8	68.8	55.5	3049	ohc	four	141	mpfi
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3012	ohcv	six	173	mpfi
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3217	ohc	six	145	idi
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	109.1	188.8	68.9	55.5	3062	ohc	four	141	mpfi

In [27]:
```python
#Checking Dimensions of the Data.
df.shape
```

Out[27]:
```
(205, 26)
```

In [28]:
```python
#Checking the basic information of dataset.

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
```

- Observation

1.)From above output we can observe that 10 categorical & 16 numerical Attributes. 2.)All the features are having correct data-types. So we don't have to do any changes.

In [29]:
```python
 #Descriptive Statistics Analysis.

df.describe()
```

Out[29]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	highwaympg	pric
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.00000
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.329756	3.255415	10.142537	104.117073	5125.121951	25.219512	30.751220	13276.71057
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	0.270844	0.313597	3.972040	39.544167	476.985643	6.542142	6.886443	7988.85233
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540000	2.070000	7.000000	48.000000	4150.000000	13.000000	16.000000	5118.00000
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.150000	3.110000	8.600000	70.000000	4800.000000	19.000000	25.000000	7788.00000
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310000	3.290000	9.000000	95.000000	5200.000000	24.000000	30.000000	10295.00000
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.580000	3.410000	9.400000	116.000000	5500.000000	30.000000	34.000000	16503.00000
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.940000	4.170000	23.000000	288.000000	6600.000000	49.000000	54.000000	45400.00000

In [30]:
```python
#Checking NaN(NULL) values in our Dataset.

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

Out[30]:

	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

Observation

We can observe that none of the features is having Missing values.

In [31]:
```python
#Showing Only Categorical Features.

df.select_dtypes(include="object").head()
```

Out[31]:

	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	enginetype	cylindernumber	fuelsystem
0	alfa-romero giulia	gas	std	two	convertible	rwd	front	dohc	four	mpfi
1	alfa-romero stelvio	gas	std	two	convertible	rwd	front	dohc	four	mpfi
2	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	ohcv	six	mpfi
3	audi 100 ls	gas	std	four	sedan	fwd	front	ohc	four	mpfi
4	audi 100ls	gas	std	four	sedan	4wd	front	ohc	five	mpfi

In [32]:
```python
#Showing only the Numerical Features.

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

Out[32]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	highwaympg	price
0	1	3	88.6	168.8	64.1	48.8	2548	130	3.47	2.68	9.0	111	5000	21	27	13495.0
1	2	3	88.6	168.8	64.1	48.8	2548	130	3.47	2.68	9.0	111	5000	21	27	16500.0
2	3	1	94.5	171.2	65.5	52.4	2823	152	2.68	3.47	9.0	154	5000	19	26	16500.0
3	4	2	99.8	176.6	66.2	54.3	2337	109	3.19	3.40	10.0	102	5500	24	30	13950.0
4	5	2	99.4	176.6	66.4	54.3	2824	136	3.19	3.40	8.0	115	5500	18	22	17450.0

🔹 Checking the Unique Car Company Names.

In [42]:
```python
df["CompanyName"].unique()
```

Out[42]:
```
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 [43]:
```python
#Creating a Function to Replace the Values.


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')
df["CompanyName"].unique()
```

Out[43]:
```
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)
```

Observation

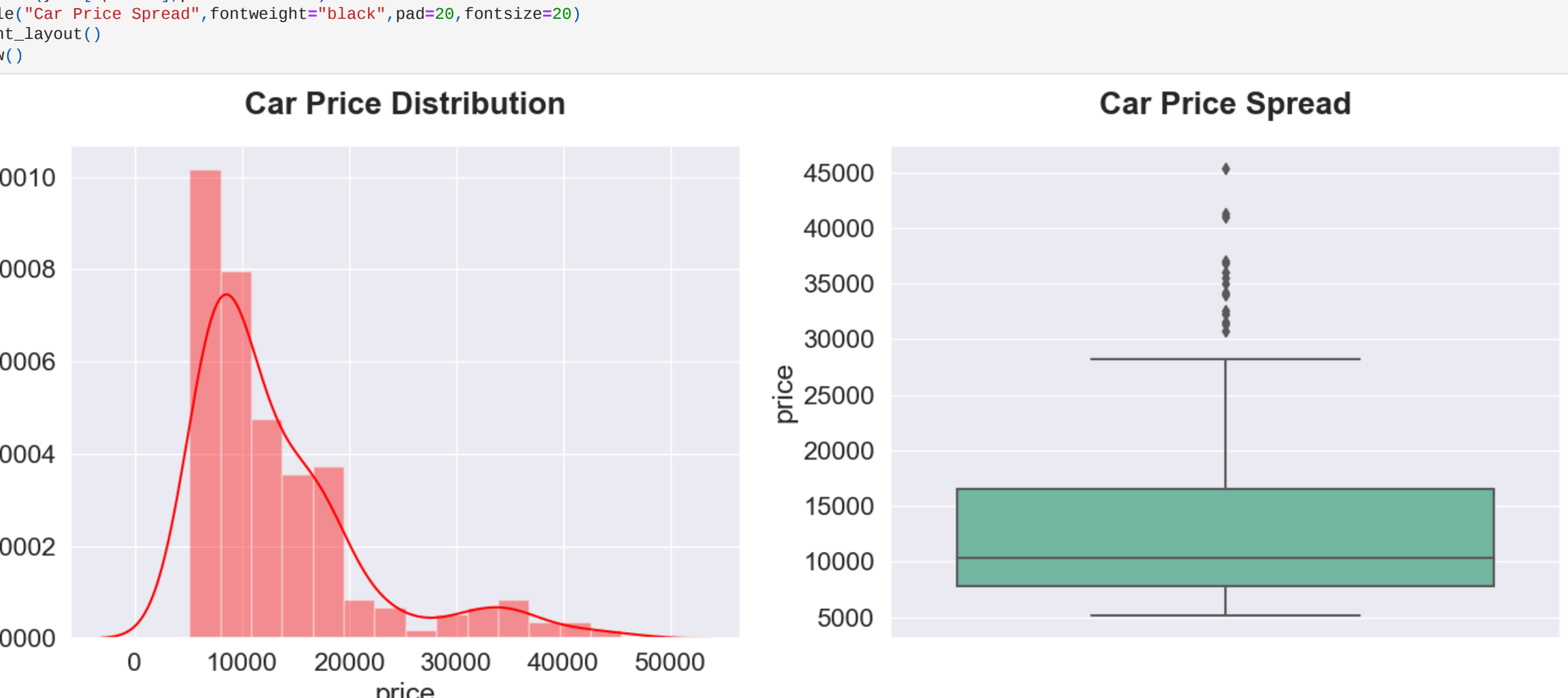
Now all the car company name seems correct.So we don't need to do any more cleaning. Now we can go to next step which is exploratory data analysis.

🔹 Exploratory data Analysis

In [44]:
```python
#Visualizing our Target Feature.

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 [45]:
```python
df["price"].agg(["min","mean","median","max","std","skew"]).to_frame().T
```

Out[45]:

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

Insights

We can clearly observe that our Car Price Feature is Right Skewed. We can clearly observe that there is a significant difference between mean & median value. We can also make an insight that most of the car's price is below 14000. We can also that the skewness of the car price is above 1.5 which means that the data points are highly spread.

In [49]:
```python
df[df["CompanyName"]=="mercury"]
```

Out[49]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth	carheight	curbweight	enginetype	cylindernumber	enginesize	fuels
75	76	1	mercury	gas	turbo	two	hatchback	rwd	front	102.7	178.4	68.0	54.8	2910	ohc	four	140	

In [50]:
```python
df[df["CompanyName"]=="Nissan"]
```

Out[50]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth	carheight	curbweight	enginetype	cylindernumber	enginesize	fuelsy
89	90	1	Nissan	gas	std	two	sedan	fwd	front	94.5	165.3	63.8	54.5	1889	ohc	four	97	

In [51]:
```python
df[df["CompanyName"]=="renault"]
```

Out[51]:

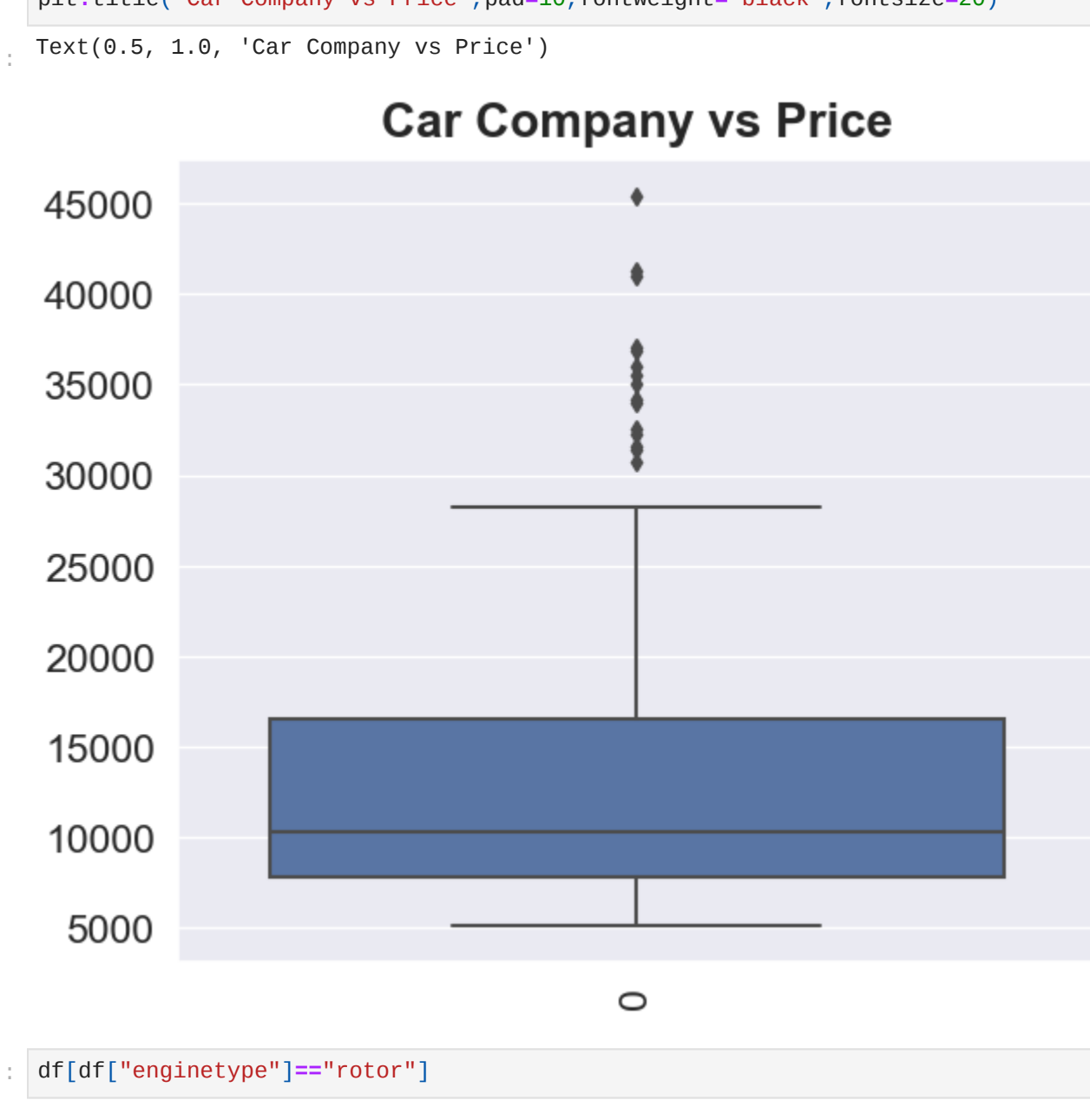
	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth	carheight	curbweight	enginetype	cylindernumber	enginesize	fuel
130	131	0	renault	gas	std	four	wagon	fwd	front	96.1	181.5	66.5	55.2	2579	ohc	four	132	
131	132	2	renault	gas	std	two	hatchback	fwd	front	96.1	176.8	66.6	50.5	2460	ohc	four	132	

Insights

Toyota company has sold the highest number of cars. So we can say that Toyota is kind of customers most favoured company. Nisaan ,Mercury or renault are having very low data-points. So we can't make any inference of least sold car companies.

In [58]:
```python
#Visualizing Car Company w.r.t Price.
plt.figure(figsize=(15,6))
plt.subplot(1,2,1)
sns.boxplot(df["price"])
plt.xticks(rotation=90)
plt.title("Car Company vs Price",pad=10,fontweight="black",fontsize=20)
```

Out[58]:
```
Text(0.5, 1.0, 'Car Company vs Price')
```



In [61]:
```python
df[df["enginetype"]=="rotor"]
```

Out[61]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth	carheight	curbweight	enginetype	cylindernumber	enginesize	fuels
55	56	3	mazda	gas	std	two	hatchback	rwd	front	95.3	169.0	65.7	49.6	2380	rotor	two	70	
56	57	3	mazda	gas	std	two	hatchback	rwd	front	95.3	169.0	65.7	49.6	2380	rotor	two	70	
57	58	3	mazda	gas	std	two	hatchback	rwd	front	95.3	169.0	65.7	49.6	2385	rotor	two	70	
58	59	3	mazda	gas	std	two	hatchback	rwd	front	95.3	169.0	65.7	49.6	2500	rotor	two	80	

Insights

Cars having Overhead Camshaft (OHC) engines are mostly sold. Only one car has been sold having engine type dohcv. There are very few data-points of engine type dohcv & rotor. So we can say that cars having ohcv engine types are mostly expensive. Cars hvaing Overhead Camshaft (OHC) engines are least expensive cars.

🔹 Data Preprocessing

In [75]:
```python
#Deriving New Features From "Company Name" Feature.
#As we made an insight above that we can split the car company name into different price ranges.Like Low Range, Medium Range, High Range cars.

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

Out[75]:

CompanyName	Nissan	alfa-romero	audi	bmw	buick	chevrolet	dodge	honda	isuzu	jaguar	mazda	mercury	mitsubishi	nissan	peugeot	plymouth	porsche	renault	saab	subaru	toyo
mean	5499.0	15498.33	17859.17	26118.75	33647.0	6007.0	7875.44	8184.69	8916.5	34600.0	10652.88	16503.0	9239.77	10704.88	15489.09	7963.43	31400.5	9595.0	15223.33	8541.25	9885

In [76]:
```python
df = df.merge(z.T,how="left",on="CompanyName")
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[76]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth	carheight	curbweight	enginetype	cylindernumber	enginesize	fuelsy
0	1	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130	
1	2	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130	
2	3	1	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4	2823	ohcv	six	152	
3	4	2	audi	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3	2337	ohc	four	109	
4	5	2	audi	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3	2824	ohc	five	136	

In [77]:
```python
#Creating new DataFrame with all the useful Features.

new_df = df[['fueltype','aspiration','doornumber','carbody','drivewheel','enginetype','cylindernumber','fuelsystem'
             ,'wheelbase','carlength','carwidth','curbweight','enginesize','boreratio','horsepower','citympg','highwaympg',
             'price','CarsRange']]
new_df.head()
```

Out[77]:

	fueltype	aspiration	doornumber	carbody	drivewheel	enginetype	cylindernumber	fuelsystem	wheelbase	carlength	carwidth	curbweight	enginesize	boreratio	horsepower	citympg	highwaympg	price
0	gas	std	two	convertible	rwd	dohc	four	mpfi	88.6	168.8	64.1	2548	130	3.47	111	21	27	13495.0
1	gas	std	two	convertible	rwd	dohc	four	mpfi	88.6	168.8	64.1	2548	130	3.47	111	21	27	16500.0
2	gas	std	two	hatchback	rwd	ohcv	six	mpfi	94.5	171.2	65.5	2823	152	2.68	154	19	26	16500.0
3	gas	std	four	sedan	fwd	ohc	four	mpfi	99.8	176.6	66.2	2337	109	3.19	102	24	30	13950.0
4	gas	std	four	sedan	4wd	ohc	five	mpfi	99.4	176.6	66.4	2824	136	3.19	115	18	22	17450.0

In [78]:
```python
#Creating Dummies Variables for all the Categorical Features.

new_df = pd.get_dummies(columns=["fueltype","aspiration","doornumber","carbody","drivewheel","enginetype",
                                "cylindernumber","fuelsystem","CarsRange"],data=new_df)
new_df.head()
```

Out[78]:

	wheelbase	carlength	carwidth	curbweight	enginesize	boreratio	horsepower	citympg	highwaympg	price	fueltype_diesel	fueltype_gas	aspiration_std	aspiration_turbo	doornumber_four	doornumber_two
0	88.6	168.8	64.1	2548	130	3.47	111	21	27	13495.0	0	1	1	0	0	1
1	88.6	168.8	64.1	2548	130	3.47	111	21	27	16500.0	0	1	1	0	0	1
2	94.5	171.2	65.5	2823	152	2.68	154	19	26	16500.0	0	1	1	0	0	1
3	99.8	176.6	66.2	2337	109	3.19	102	24	30	13950.0	0	1	1	0	1	0
4	99.4	176.6	66.4	2824	136	3.19	115	18	22	17450.0	0	1	1	0	1	0

Feature Scaling of Numerical Data.

In [79]:
```python
scaler = StandardScaler()
num_cols = ['wheelbase','carlength','carwidth','curbweight','enginesize','boreratio','horsepower',
            'citympg','highwaympg']

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

Out[79]:

	wheelbase	carlength	carwidth	curbweight	enginesize	boreratio	horsepower	citympg	highwaympg	price	fueltype_diesel	fueltype_gas	aspiration_std	aspiration_turbo	doornumber_four	doornumber_tw
0	-1.690772	-0.426521	-0.844782	-0.014566	0.074449	0.519071	0.174483	-0.646553	-0.546059	13495.0	0	1	1	0	0	
1	-1.690772	-0.426521	-0.844782	-0.014566	0.074449	0.519071	0.174483	-0.646553	-0.546059	16500.0	0	1	1	0	0	
2	-0.708596	-0.231513	-0.190566	0.514882	0.604046	-2.404880	1.264536	-0.953012	-0.691627	16500.0	0	1	1	0	0	
3	0.173698	0.207256	0.136542	-0.420797	-0.431076	-0.517266	-0.053668	-0.186865	-0.109354	13950.0	0	1	1	0	1	
4	0.107110	0.207256	0.230001	0.516807	0.218885	-0.517266	0.275883	-1.106241	-1.273900	17450.0	0	1	1	0	1	

In [82]:
```python
#Selecting Features & Labels for Model Training & Testing.
x = new_df.drop(columns=["price"])
y = new_df["price"]
x.shape
```

Out[82]:
```
(205, 48)
```

In [83]:
```python
y.shape
```

Out[83]:
```
(205,)
```

In [84]:
```python
#Splitting Data for Model Traning & Testing.

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
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,)
```

🔹 Building out the model

In [85]:
```python
training_score = []
testing_score = []
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)
```

🪐Linear regression

In [87]:
```python
model_prediction(LinearRegression())
```

```
r2_Score of LinearRegression() model on Training Data is: 96.0384799952984
r2_Score of LinearRegression() model on Testing Data is: 88.40540860813923
```

🪐DecisionTreeRegressor

In [88]:
```python
model_prediction(DecisionTreeRegressor())
```

```
r2_Score of DecisionTreeRegressor() model on Training Data is: 99.86537119069865
r2_Score of DecisionTreeRegressor() model on Testing Data is: 91.36869031650147
```

🪐RandomForestRegressor

In [89]:
```python
model_prediction(RandomForestRegressor())
```

```
r2_Score of RandomForestRegressor() model on Training Data is: 98.76195779742146
r2_Score of RandomForestRegressor() model on Testing Data is: 95.3310695145149
```

🪐AdaBoostRegressor

In [90]:
```python
model_prediction(AdaBoostRegressor())
```

```
r2_Score of AdaBoostRegressor() model on Training Data is: 96.13232438011892
r2_Score of AdaBoostRegressor() model on Testing Data is: 91.19728053897768
```

🪐GradientBoostingRegressor

In [91]:
```python
model_prediction(GradientBoostingRegressor())
```

```
r2_Score of GradientBoostingRegressor() model on Training Data is: 99.37237309193024
r2_Score of GradientBoostingRegressor() model on Testing Data is: 92.48950516083171
```

🔹 Model Performance Comparision

In [93]:
```python
models = ["Linear Regression","Decision Tree","Random Forest","Ada Boost","Gradient Boost"]
df = pd.DataFrame({"Algorithms":models,
                   "Training Score":training_score,
                   "Testing Score":testing_score})
```

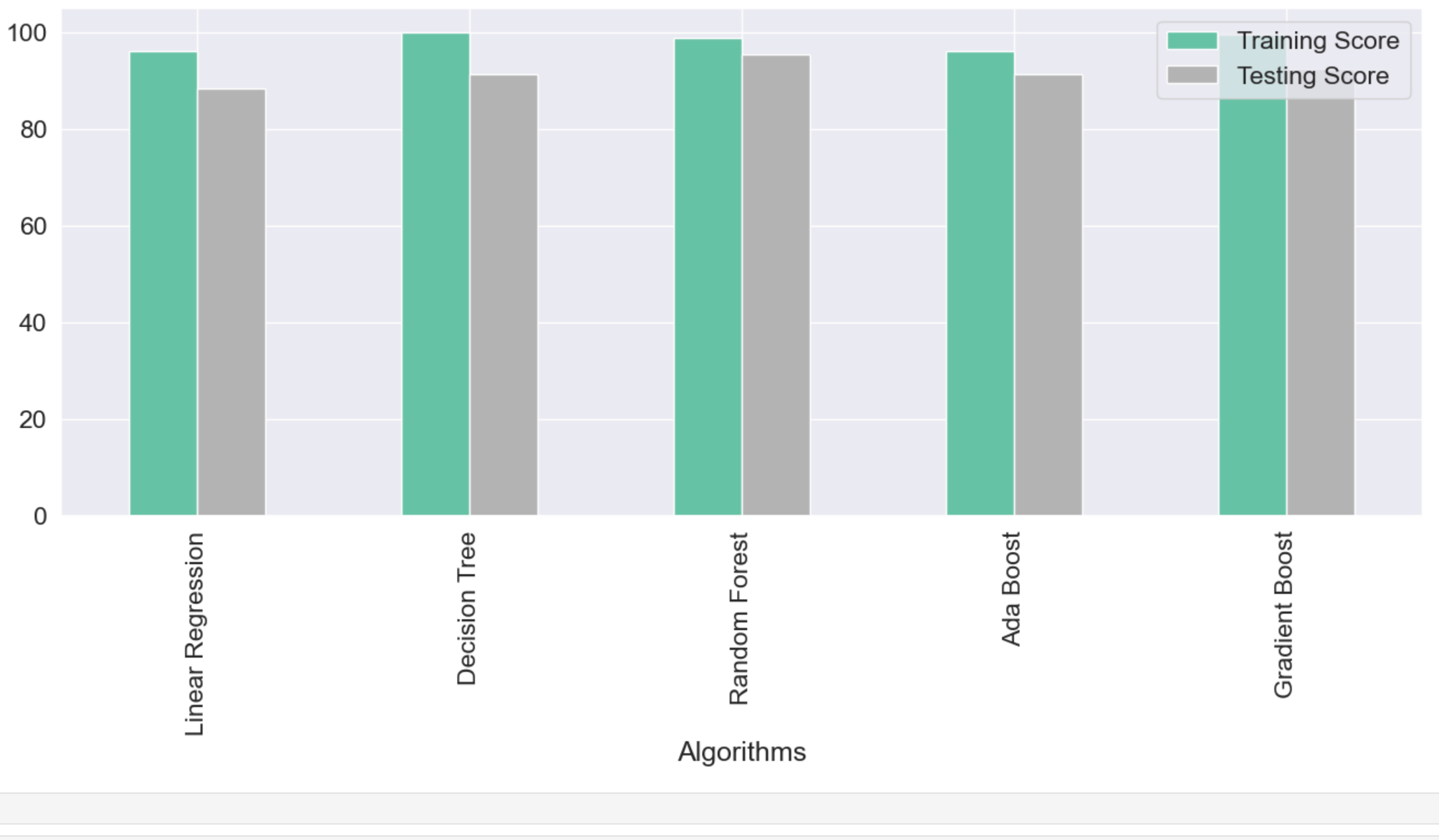
In [94]:
```python
df
```

Out[94]:

	Algorithms	Training Score	Testing Score
0	Linear Regression	96.038480	88.405409
1	Decision Tree	99.865371	91.368690
2	Random Forest	98.761958	95.331070
3	Ada Boost	96.132324	91.197281
4	Gradient Boost	99.372373	92.489505

plotting above chart

In [100…
```python
df.plot(x="Algorithms",y=["Training Score","Testing Score"], figsize=(16,6),kind="bar",
        title="Performance Visualization of Different Models",colormap="Set2")
plt.show()
```



In [ ]:

In [ ]: