Car Price Prediction using RANDOM FOREST

In [1]:

```
import pandas as pd
```

Load Dataset from Local Directory

In [2]:

```
from google.colab import files
uploaded = files.upload()
```

Choose File **No file selected**

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Dataset.csv to Dataset.csv
```

Load Dataset

In [5]:

```
dataset = pd.read_csv('Dataset.csv')
dataset = dataset.drop(['car_ID'],axis = 1)
```

Summarize Dataset

In [6]:

```
print(dataset.shape)
print(dataset.head(5))
```

```
(205, 25)
   symboling                   CarName fueltype  ... citympg highwaympg    price
0          3         alfa-romero giulia      gas  ...      21          27  13495.0
1          3        alfa-romero stelvio      gas  ...      21          27  16500.0
2          1  alfa-romero Quadrifoglio      gas  ...      19          26  16500.0
3          2               audi 100 ls      gas  ...      24          30  13950.0
4          2                audi 100ls      gas  ...      18          22  17450.0

[5 rows x 25 columns]
```

Splitting Dataset into X & Y

This X contains Numerical & Text Dataset

In [7]:

```
Xdata = dataset.drop('price', axis = 'columns')
numericalCols = Xdata.select_dtypes(exclude = ['object']).columns
X = Xdata[numericalCols]
X
```

Out[7]:

	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionratio	horse
0	3	88.6	168.8	64.1	48.8	2548	130	3.47	2.68	9.0	
1	3	88.6	168.8	64.1	48.8	2548	130	3.47	2.68	9.0	
2	1	94.5	171.2	65.5	52.4	2823	152	2.68	3.47	9.0	
3	2	99.8	176.6	66.2	54.3	2337	109	3.19	3.40	10.0	

	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionratio	horse
4	2	99.4	176.6	66.4	54.3	2824	136	3.19	3.40	8.0	
...	...	...	...	...	...	...	...	...	...	...	
200	-1	109.1	188.8	68.9	55.5	2952	141	3.78	3.15	9.5	
201	-1	109.1	188.8	68.8	55.5	3049	141	3.78	3.15	8.7	
202	-1	109.1	188.8	68.9	55.5	3012	173	3.58	2.87	8.8	
203	-1	109.1	188.8	68.9	55.5	3217	145	3.01	3.40	23.0	
204	-1	109.1	188.8	68.9	55.5	3062	141	3.78	3.15	9.5	

205 rows × 14 columns

In [8]:

```
Y = dataset['price']
Y
```

Out[8]:

```
0       13495.0
1       16500.0
2       16500.0
3       13950.0
4       17450.0
         ...
200     16845.0
201     19045.0
202     21485.0
203     22470.0
204     22625.0
Name: price, Length: 205, dtype: float64
```

Scaling the Independent Variables (Features)

In [11]:

```
from sklearn.preprocessing import scale
cols = X.columns
X = pd.DataFrame(scale(X))
X.columns = cols
X
```

Out[11]:

	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionratio	ho
0	1.743470	-1.690772	-0.426521	-0.844782	-2.020417	-0.014566	0.074449	0.519071	-1.839377	-0.288349	
1	1.743470	-1.690772	-0.426521	-0.844782	-2.020417	-0.014566	0.074449	0.519071	-1.839377	-0.288349	
2	0.133509	-0.708596	-0.231513	-0.190566	-0.543527	0.514882	0.604046	-2.404880	0.685946	-0.288349	
3	0.938490	0.173698	0.207256	0.136542	0.235942	-0.420797	-0.431076	-0.517266	0.462183	-0.035973	
4	0.938490	0.107110	0.207256	0.230001	0.235942	0.516807	0.218885	-0.517266	0.462183	-0.540725	
...	...	...	...	...	...	...	...	...	...	...	
200	-1.476452	1.721873	1.198549	1.398245	0.728239	0.763241	0.339248	1.666445	-0.336970	-0.162161	
201	-1.476452	1.721873	1.198549	1.351515	0.728239	0.949992	0.339248	1.666445	-0.336970	-0.364062	
202	-1.476452	1.721873	1.198549	1.398245	0.728239	0.878757	1.109571	0.926204	-1.232021	-0.338824	

	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionratio	ho
203	-1.476452	1.721873	1.198549	1.398245	0.728239	1.273437	0.435538	1.183483	0.462183	3.244916	
204	-1.476452	1.721873	1.198549	1.398245	0.728239	0.975021	0.339248	1.666445	-0.336970	-0.162161	

205 rows × 14 columns

Spliting Dataset into Train & Test

In [12]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.20, random_state
= 0)
```

Traing using RANDOM FOREST

In [13]:

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()
model.fit(x_train, y_train)
```

Out[13]:

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

Evaluating Model

In [15]:

```
ypred = model.predict(x_test)

from sklearn.metrics import r2_score
r2score = r2_score(y_test, ypred)
print("R2Score", r2score*100)
```

```
R2Score 90.67976965632458
```