ANKIT GAUTAM

TASK-2 CAR PRICE PREDICTION WITH MACHINE LEARNING

price of a car depends on a lot of factors like the goodwill of the brand of the car,features of the car, horsepower and the mileage it gives and many more.

In [1]:

```
import pandas as pd #importing the library
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df = pd.read_csv("CarPrice_Assignment.csv") #importing the dataset
df.head()
```

Out[2]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	engin
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

In [3]:

```
df.shape
```

Out[3]:

```
(205, 26)
```

In [4]:

```
df.isnull().sum() #check their is any null vale
```

Out[4]:

```
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
dtype: int64
```

In [5]:

```
df.info()
```

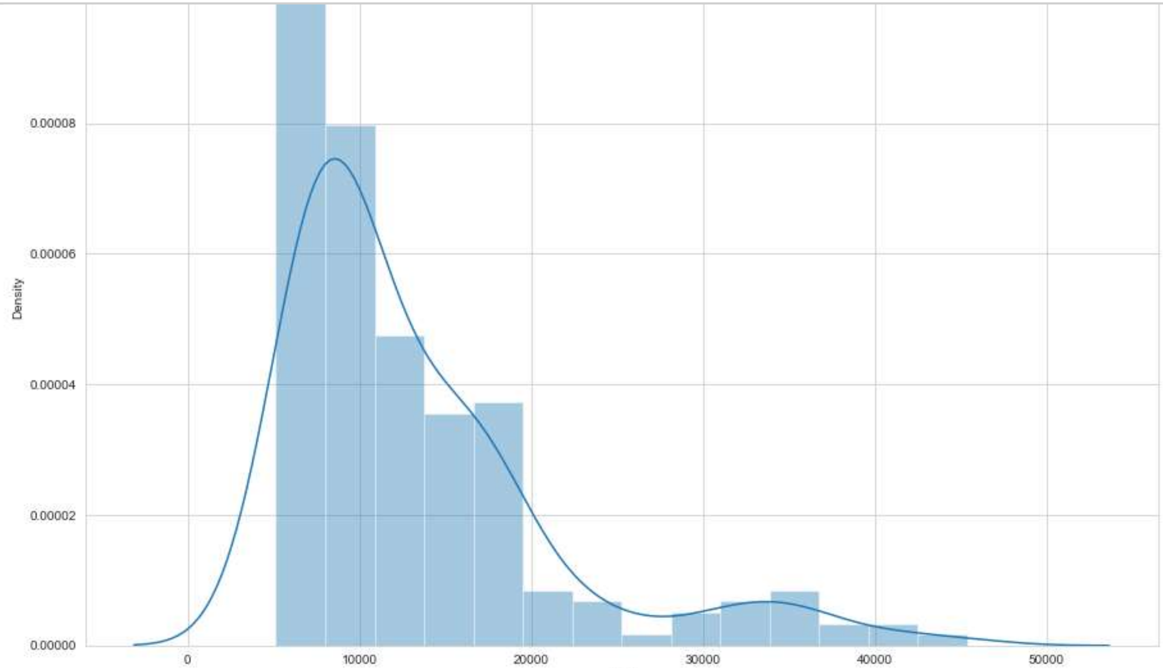
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           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
```

The column“Price” is the target variable and rest of the columns are independent variables.

The price column in this dataset is supposed to be the column whose values we need to predict. So let’s see the distribution of the values of the price column

In [6]:

```python
sns.set_style("whitegrid")
plt.figure(figsize=(15, 10))
sns.distplot(df.price)
plt.show()
```



In [7]:

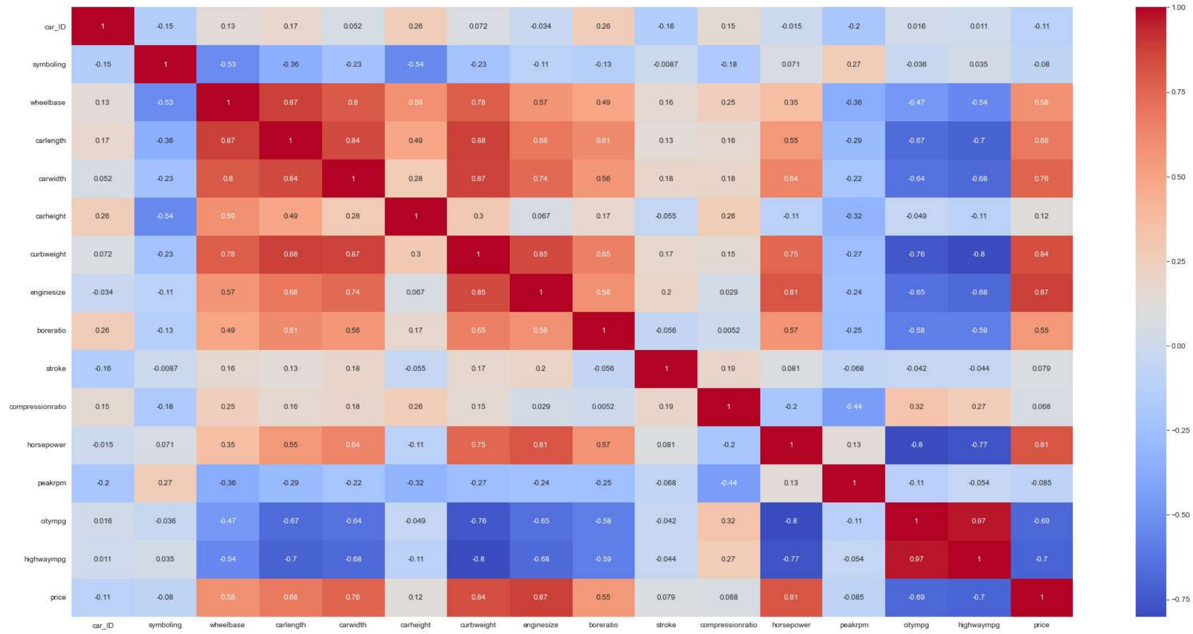
```python
df.corr()
```

Out[7]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	e
car_ID	1.000000	-0.151621	0.129729	0.170636	0.052387	0.255960	0.071962	
symboling	-0.151621	1.000000	-0.531954	-0.357612	-0.232919	-0.541038	-0.227691	
wheelbase	0.129729	-0.531954	1.000000	0.874587	0.795144	0.589435	0.776386	
carlength	0.170636	-0.357612	0.874587	1.000000	0.841118	0.491029	0.877728	
carwidth	0.052387	-0.232919	0.795144	0.841118	1.000000	0.279210	0.867032	
carheight	0.255960	-0.541038	0.589435	0.491029	0.279210	1.000000	0.295572	
curbweight	0.071962	-0.227691	0.776386	0.877728	0.867032	0.295572	1.000000	
enginesize	-0.033930	-0.105790	0.569329	0.683360	0.735433	0.067149	0.850594	
boreratio	0.260064	-0.130051	0.488750	0.606454	0.559150	0.171071	0.648480	
stroke	-0.160824	-0.008735	0.160959	0.129533	0.182942	-0.055307	0.168790	
compressionratio	0.150276	-0.178515	0.249786	0.158414	0.181129	0.261214	0.151362	
horsepower	-0.015006	0.070873	0.353294	0.552623	0.640732	-0.108802	0.750739	
peakrpm	-0.203789	0.273606	-0.360469	-0.287242	-0.220012	-0.320411	-0.266243	
citympg	0.015940	-0.035823	-0.470414	-0.670909	-0.642704	-0.048640	-0.757414	
highwaympg	0.011255	0.034606	-0.544082	-0.704662	-0.677218	-0.107358	-0.797465	
price	-0.109093	-0.079978	0.577816	0.682920	0.759325	0.119336	0.835305	

In [8]:

```
plt.figure(figsize=(30, 15)) #heatmap
correlations = df.corr()
sns.heatmap(correlations, cmap="coolwarm", annot=True)
plt.show()
```



The heatmap shows some useful insights:Correlation of target variable “Price” with independent variables:Price is highly (positively) correlated with wheelbase, carlength, carwidth, curbweight, enginesize, horsepower (notice how all of these variables represent the size/weight/engine power of the car)

Price is negatively correlated to ‘citympg’ and ‘highwaympg’ (-0.70 approximately). (This suggest that cars having high mileage may fall in the ‘economy’ cars category, and are priced lower (think Maruti Alto/Swift type of cars, which are designed to be affordable by the middle class, who value mileage more than horsepower/size of car etc.)Correlation among independent variables)

In [9]:

```
X = df[["symboling", "wheelbase", "carlength", "carwidth", "carheight", "curbweight",  "engi
y = df['price']  # Identify X,y
```

In [10]:

```
from sklearn.model_selection import train_test_split  #spliting the dataset into train & tes
```

In [11]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=10)
```

In [12]:

```
from sklearn.preprocessing import MinMaxScaler #scaling
```

In [13]:

```
scaler = MinMaxScaler()
```

In [14]:

```
scaler.fit(X_train)
```

Out[14]:

```
MinMaxScaler()
```

In [15]:

```
X_train_scaled = scaler.transform(X_train)
```

In [16]:

```
X_train_scaled
```

Out[16]:

```
array([[0.6       , 0.55976676, 0.77014925, ..., 0.14285714, 0.23684211,
        0.34263443],
       [0.6       , 0.20699708, 0.24179104, ..., 0.48571429, 0.57894737,
        0.06183903],
       [1.        , 0.25364431, 0.41641791, ..., 0.08571429, 0.18421053,
        0.14465518],
       ...,
       [0.4       , 0.49271137, 0.71492537, ..., 0.05714286, 0.15789474,
        0.63656224],
       [1.        , 0.2303207 , 0.41492537, ..., 0.14285714, 0.28947368,
        0.41954223],
       [0.4       , 0.37609329, 0.55373134, ..., 0.05714286, 0.15789474,
        0.31629927]])
```

In [17]:

```
from sklearn.tree import DecisionTreeRegressor # train the model
model = DecisionTreeRegressor()
```

In [18]:

```
model.fit(X_train, y_train)
```

Out[18]:

```
DecisionTreeRegressor()
```

In [19]:

```
model.score(X_test,y_test)
```

Out[19]:

```
0.9775227217595573
```

The model gives 99% accuracy on the test set, which is excellent.

In [ ]:

```
```