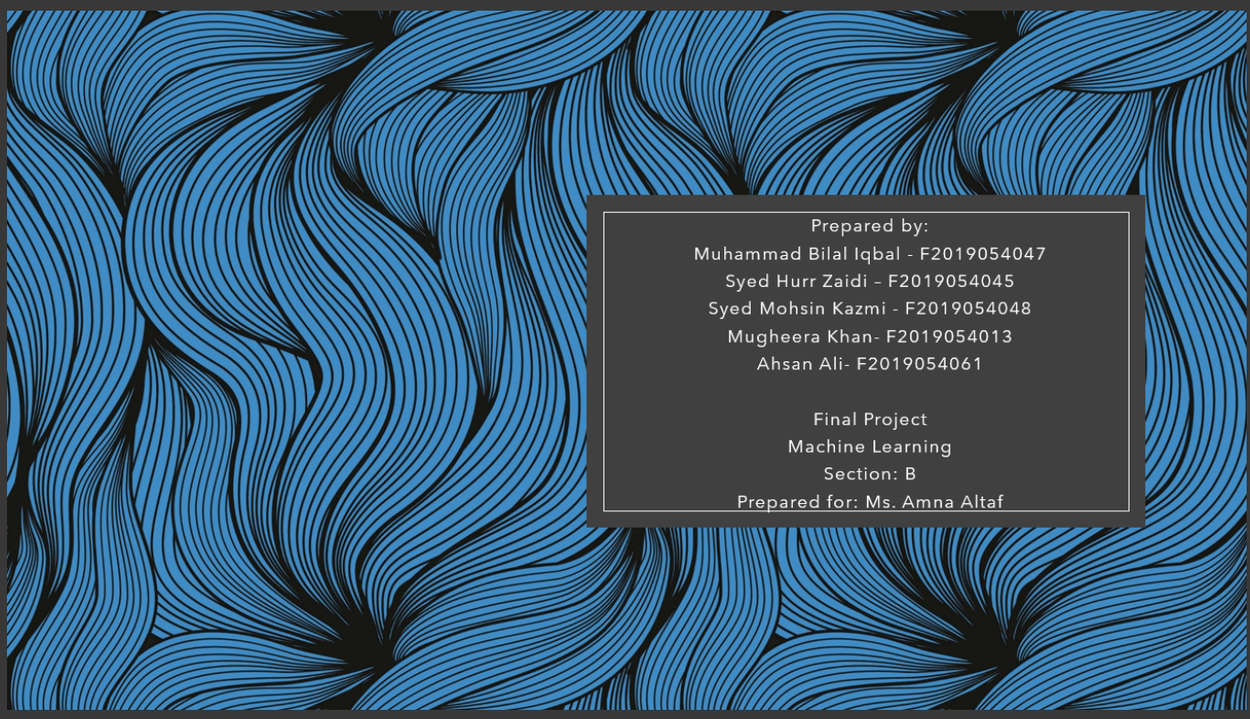



```
# necessary imports

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import StandardScaler
plt.style.use('fivethirtyeight')
%matplotlib inline
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import LinearSVR
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.ensemble import GradientBoostingRegressor
```

```
pd.set_option('display.max_columns', 100)
pd.set_option('display.max_rows', 500)
```

```
df = pd.read_csv('/content/CarPrice_Assignment.csv')
df.head()
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	en
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	

```
df.shape
```

```
(205, 26)
```

```
df.describe()
```

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionr
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.0
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.329756	3.255415	10.14

```
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
```

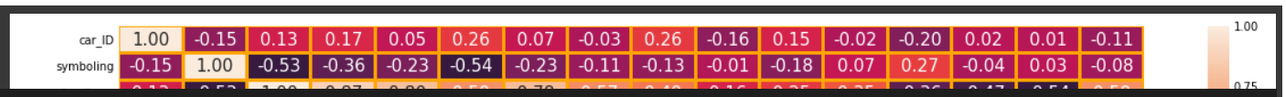
- There are no Null values in the data

```
df.isna().sum()
```

```
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
```

```
# heatmap of the data for checking the correlation between the numerical features and target column.

plt.figure(figsize = (18, 7))
sns.heatmap(df.corr(), annot = True, fmt = '0.2f', annot_kws = {'size' : 15}, linewidth = 2, linecolor = 'orange')
plt.show()
```




```
df.head()
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	168.8	64.1
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	168.8	64.1
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	176.6	66.2
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	176.6	66.4

```
df['CarName'] = df['CarName'].str.split(' ', expand = True)[0]
```

```
# handling duplicate values

df['CarName'] = df['CarName'].replace({'toyouta': 'toyota', 'Nissan': 'nissan', 'maxda': 'mazda', 'vokswagen': 'volkswagen',
                                       'vw': 'volkswagen', 'porcshce': 'porsche'})
```

Data Preprocessing

```
df.head()
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth
0	1	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1
1	2	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1
2	3	1	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5
3	4	2	audi	gas	std	four	sedan	fwd	front	99.8	176.6	66.2
4	5	2	audi	gas	std	four	sedan	4wd	front	99.4	176.6	66.4

```
df.drop(columns = ['car_ID'], axis = 1, inplace = True)
```

```
df.head()
```

	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth	carheight
0	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8
1	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8
2	1	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4
3	2	audi	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3
4	2	audi	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3

```
cat_cols = df.select_dtypes(include = 'object')
cat_cols.head()
```

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

```
df['cylindernumber'].value_counts()
```

```
four      159
six        24
five       11
eight       5
```

```
two         4
three       1
twelve      1
Name: cylindernumber, dtype: int64
```

```
num_cols = df.select_dtypes(exclude = 'object')
cat_cols = df.select_dtypes(include = 'object')
```

```
# checking for outliers

cols = num_cols.columns

plt.figure(figsize = (16, 20))
plotnumber = 1

# plotting the countplot of each categorical column.

for i in range(1, len(cols)):
    if plotnumber <= 16:
        ax = plt.subplot(4, 4, plotnumber)
        sns.boxplot(x = cols[i], data = df, ax = ax, palette='rocket')
        plt.title(f"\n{cols[i]} \n", fontsize = 20)

    plotnumber += 1

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

wheelbase carlength carwidth carheight

```
# encoding ordinal categorical columns

df['doornumber'] = df['doornumber'].map({'two': 2, 'four': 4})
df['cylindernumber'] = df['cylindernumber'].map({'two': 2, 'three': 3, 'four': 4, 'five': 5, 'six': 6, 'eight': 8, 'twelve': 12})
```

```
df.head()
```

	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth	carheight
0	3	alfa-romero	gas	std	2	convertible	rwd	front	88.6	168.8	64.1	48.8
1	3	alfa-romero	gas	std	2	convertible	rwd	front	88.6	168.8	64.1	48.8
2	1	alfa-romero	gas	std	2	hatchback	rwd	front	94.5	171.2	65.5	52.4
3	2	audi	gas	std	4	sedan	fwd	front	99.8	176.6	66.2	54.3
4	2	audi	gas	std	4	sedan	4wd	front	99.4	176.6	66.4	54.3

```
# creating features and label variable

X = df.drop(columns = 'price', axis = 1)
y = df['price']
```

compressionratio horsepower peakrpm citympg

```
X = pd.get_dummies(X, drop_first = True)
X.head()
```

	symboling	doornumber	wheelbase	carlength	carwidth	carheight	curbweight	cylindernumber	enginesize	boreratio	stroke	compress
0	3	2	88.6	168.8	64.1	48.8	2548	4	130	3.47	2.68	
1	3	2	88.6	168.8	64.1	48.8	2548	4	130	3.47	2.68	
2	1	2	94.5	171.2	65.5	52.4	2823	6	152	2.68	3.47	
3	2	4	99.8	176.6	66.2	54.3	2337	4	109	3.19	3.40	
4	2	4	99.4	176.6	66.4	54.3	2824	5	136	3.19	3.40	

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
```

```
# scaling data
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Linear Regression

```
reg = LinearRegression()
reg.fit(X_train, y_train)
y_pred_reg = reg.predict(X_test)
mse_reg = mean_squared_error(y_test, y_pred_reg)
mae_reg = mean_absolute_error(y_test, y_pred_reg)
r2_reg = r2_score(y_test, y_pred_reg)
print("Linear Regression:")
print("Mean Squared Error:", mse_reg)
print("Mean Absolute Error:", mae_reg)
print("R-squared Score:", r2_reg)
```

```
Linear Regression:
Mean Squared Error: 10730664.086408442
Mean Absolute Error: 2053.5954942800777
R-squared Score: 0.8613902238127651
```

```
rf = RandomForestRegressor()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
mae_rf = mean_absolute_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)
print("Linear Regression:")
print("Mean Squared Error:", mse_rf)
print("Mean Absolute Error:", mae_rf)
print("R-squared Score:", r2_rf)
```

```
Linear Regression:
Mean Squared Error: 7465012.40900206
Mean Absolute Error: 1807.9887604065038
R-squared Score: 0.9035731907256982
```

```
svr = LinearSVR()
svr.fit(X_train, y_train)
y_pred_svr = rf.predict(X_test)
mse_svr = mean_squared_error(y_test, y_pred_svr)
mae_svr = mean_absolute_error(y_test, y_pred_svr)
r2_svr = r2_score(y_test, y_pred_svr)
print("Linear SVR:")
print("Mean Squared Error:", mse_svr)
print("Mean Absolute Error:", mae_svr)
print("R-squared Score:", r2_svr)
```

```
Linear SVR:
Mean Squared Error: 7465012.40900206
Mean Absolute Error: 1807.9887604065038
R-squared Score: 0.9035731907256982
```

```
GB = GradientBoostingRegressor()
GB.fit(X_train, y_train)
y_pred_GB = GB.predict(X_test)
mse_GB = mean_squared_error(y_test, y_pred_GB)
mae_GB = mean_absolute_error(y_test, y_pred_GB)
r2_GB = r2_score(y_test, y_pred_GB)
print("Gradient Boosting Regressor:")
print("Mean Squared Error:", mse_GB)
print("Mean Absolute Error:", mae_GB)
print("R-squared Score:", r2_GB)
```

```
Gradient Boosting Regressor:
Mean Squared Error: 8539762.55811899
Mean Absolute Error: 1881.515430675795
R-squared Score: 0.8896904639506626
```


✓ 0s completed at 2:21 AM