```python
# import all libraries and dependencies for dataframe
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#loading data
df = pd.read_csv("D:\Data Visualization(IShant)\
CarPrice_Assignment.csv")

df
```

```
     car_ID  symboling                   CarName fueltype
aspiration  \
0         1          3        alfa-romero giulia      gas         std

1         2          3       alfa-romero stelvio      gas         std

2         3          1  alfa-romero Quadrifoglio      gas         std

3         4          2               audi 100 ls      gas         std

4         5          2                audi 100ls      gas         std

..      ...        ...                       ...      ...         ...

200     201         -1           volvo 145e (sw)      gas         std

201     202         -1               volvo 144ea      gas       turbo

202     203         -1               volvo 244dl      gas         std

203     204         -1                 volvo 246   diesel       turbo

204     205         -1               volvo 264gl      gas       turbo


    doornumber      carbody drivewheel enginelocation  wheelbase  ...
\
0          two  convertible        rwd          front       88.6  ...

1          two  convertible        rwd          front       88.6  ...

2          two    hatchback        rwd          front       94.5  ...

3         four        sedan        fwd          front       99.8  ...

4         four        sedan        4wd          front       99.4  ...

..         ...          ...        ...            ...        ...  ...

200       four        sedan        rwd          front      109.1  ...
```

```
201       four       sedan       rwd          front      109.1  ...
202       four       sedan       rwd          front      109.1  ...
203       four       sedan       rwd          front      109.1  ...
204       four       sedan       rwd          front      109.1  ...

     enginesize  fuelsystem  boreratio  stroke compressionratio
horsepower  \
0           130        mpfi       3.47    2.68              9.0
111
1           130        mpfi       3.47    2.68              9.0
111
2           152        mpfi       2.68    3.47              9.0
154
3           109        mpfi       3.19    3.40             10.0
102
4           136        mpfi       3.19    3.40              8.0
115
..          ...         ...        ...     ...              ...
...
200         141        mpfi       3.78    3.15              9.5
114
201         141        mpfi       3.78    3.15              8.7
160
202         173        mpfi       3.58    2.87              8.8
134
203         145         idi       3.01    3.40             23.0
106
204         141        mpfi       3.78    3.15              9.5
114

     peakrpm citympg  highwaympg    price
0       5000      21          27  13495.0
1       5000      21          27  16500.0
2       5000      19          26  16500.0
3       5500      24          30  13950.0
4       5500      18          22  17450.0
..       ...     ...         ...      ...
200     5400      23          28  16845.0
201     5300      19          25  19045.0
202     5500      18          23  21485.0
203     4800      26          27  22470.0
204     5400      19          25  22625.0

[205 rows x 26 columns]
```

```python
## information of the data
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
```

```python
# unique value of each column
df.nunique()
```

```
car_ID              205
symboling             6
CarName             147
fueltype              2
aspiration            2
doornumber            2
carbody               5
drivewheel            3
enginelocation        2
wheelbase            53
```

```
carlength            75
carwidth             44
carheight            49
curbweight          171
enginetype            7
cylindernumber        7
enginesize           44
fuelsystem            8
boreratio            38
stroke               37
compressionratio     32
horsepower           59
peakrpm              23
citympg              29
highwaympg           30
price               189
dtype: int64
```

```
# description of the data
df.describe()
```

```
           car_ID   symboling   wheelbase   carlength    carwidth
carheight  \
count  205.000000  205.000000  205.000000  205.000000  205.000000
205.000000
mean   103.000000    0.834146   98.756585  174.049268   65.907805
53.724878
std     59.322565    1.245307    6.021776   12.337289    2.145204
2.443522
min      1.000000   -2.000000   86.600000  141.100000   60.300000
47.800000
25%     52.000000    0.000000   94.500000  166.300000   64.100000
52.000000
50%    103.000000    1.000000   97.000000  173.200000   65.500000
54.100000
75%    154.000000    2.000000  102.400000  183.100000   66.900000
55.500000
max    205.000000    3.000000  120.900000  208.100000   72.300000
59.800000

        curbweight  enginesize   boreratio      stroke
compressionratio  \
count   205.000000  205.000000  205.000000  205.000000
205.000000
mean   2555.565854  126.907317    3.329756    3.255415
10.142537
std     520.680204   41.642693    0.270844    0.313597
3.972040
min    1488.000000   61.000000    2.540000    2.070000
7.000000
```

```
25%     2145.000000   97.000000     3.150000     3.110000
8.600000
50%     2414.000000  120.000000     3.310000     3.290000
9.000000
75%     2935.000000  141.000000     3.580000     3.410000
9.400000
max     4066.000000  326.000000     3.940000     4.170000
23.000000

       horsepower      peakrpm     citympg  highwaympg         price
count  205.000000   205.000000  205.000000  205.000000    205.000000
mean   104.117073  5125.121951   25.219512   30.751220  13276.710571
std     39.544167   476.985643    6.542142    6.886443   7988.852332
min     48.000000  4150.000000   13.000000   16.000000   5118.000000
25%     70.000000  4800.000000   19.000000   25.000000   7788.000000
50%     95.000000  5200.000000   24.000000   30.000000  10295.000000
75%    116.000000  5500.000000   30.000000   34.000000  16503.000000
max    288.000000  6600.000000   49.000000   54.000000  45400.000000

# Droping car_ID from the database
df=df.drop(['car_ID'],axis=1)

# Extracting Car Company from the CarName
df['CarName']=df['CarName'].str.split(' ').str[0]

# Unique Car company
df['CarName'].unique()

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)
```

Here, type error for car namming have been found for different brand So;

```
# Renaming the typo errors in Car Company names

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

# Unique Car company
df['CarName'].unique()

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)

# Check for duplicated rows
duplicated_rows_count = df.duplicated().sum()

# Display the count of duplicated rows
print("Duplicated Rows Count:", duplicated_rows_count)

Duplicated Rows Count: 0

# Changing the datatype of symboling as it is categorical variable
df['symboling']=df['symboling'].astype('str')

# Numerical and Categorical Columns
numerical_columns=df.select_dtypes(exclude=['object']).columns
categorical_columns=df.select_dtypes(include=['object']).columns

# Display the lists of numerical and categorical columns
print("Numerical Columns:", numerical_columns)
print("Categorical Columns:", categorical_columns)

Numerical Columns: Index(['wheelbase', 'carlength', 'carwidth',
'carheight', 'curbweight',
       'enginesize', 'boreratio', 'stroke', 'compressionratio',
'horsepower',
       'peakrpm', 'citympg', 'highwaympg', 'price'],
      dtype='object')
Categorical Columns: Index(['symboling', 'CarName', 'fueltype',
'aspiration', 'doornumber',
       'carbody', 'drivewheel', 'enginelocation', 'enginetype',
       'cylindernumber', 'fuelsystem'],
      dtype='object')
```

Showing the data nature of index of columns in file

```
df.shape

(205, 25)

df.describe().T

                  count        mean        std     min
25%  \
wheelbase         205.0   98.756585   6.021776   86.60   94.50

carlength         205.0  174.049268  12.337289  141.10  166.30

carwidth          205.0   65.907805   2.145204   60.30   64.10
```

```
carheight          205.0     53.724878     2.443522    47.80    52.00
curbweight         205.0   2555.565854   520.680204  1488.00  2145.00
enginesize         205.0    126.907317    41.642693    61.00    97.00
boreratio          205.0      3.329756     0.270844     2.54     3.15
stroke             205.0      3.255415     0.313597     2.07     3.11
compressionratio   205.0     10.142537     3.972040     7.00     8.60
horsepower         205.0    104.117073    39.544167    48.00    70.00
peakrpm            205.0   5125.121951   476.985643  4150.00  4800.00
citympg            205.0     25.219512     6.542142    13.00    19.00
highwaympg         205.0     30.751220     6.886443    16.00    25.00
price              205.0  13276.710571  7988.852332  5118.00  7788.00

                       50%       75%       max
wheelbase            97.00    102.40    120.90
carlength           173.20    183.10    208.10
carwidth             65.50     66.90     72.30
carheight            54.10     55.50     59.80
curbweight         2414.00   2935.00   4066.00
enginesize          120.00    141.00    326.00
boreratio             3.31      3.58      3.94
stroke                3.29      3.41      4.17
compressionratio      9.00      9.40     23.00
horsepower           95.00    116.00    288.00
peakrpm            5200.00   5500.00   6600.00
citympg              24.00     30.00     49.00
highwaympg           30.00     34.00     54.00
price             10295.00  16503.00  45400.00
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   symboling         205 non-null    object
 1   CarName           205 non-null    object
 2   fueltype          205 non-null    object
 3   aspiration        205 non-null    object
 4   doornumber        205 non-null    object
```

```
 5   carbody           205 non-null    object
 6   drivewheel        205 non-null    object
 7   enginelocation    205 non-null    object
 8   wheelbase         205 non-null    float64
 9   carlength         205 non-null    float64
 10  carwidth          205 non-null    float64
 11  carheight         205 non-null    float64
 12  curbweight        205 non-null    int64
 13  enginetype        205 non-null    object
 14  cylindernumber    205 non-null    object
 15  enginesize        205 non-null    int64
 16  fuelsystem        205 non-null    object
 17  boreratio         205 non-null    float64
 18  stroke            205 non-null    float64
 19  compressionratio  205 non-null    float64
 20  horsepower        205 non-null    int64
 21  peakrpm           205 non-null    int64
 22  citympg           205 non-null    int64
 23  highwaympg        205 non-null    int64
 24  price             205 non-null    float64
dtypes: float64(8), int64(6), object(11)
memory usage: 40.2+ KB
```

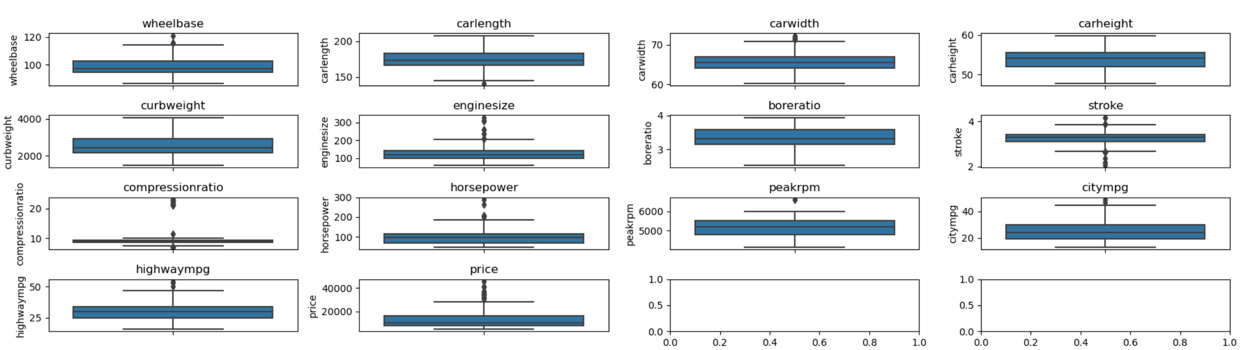
```python
# Check for missing values in each column
missing_values_count = df.isnull().sum()

# Display the count of missing values
print("Missing Values Count:")
print(missing_values_count)
```

```
Missing Values Count:
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
```

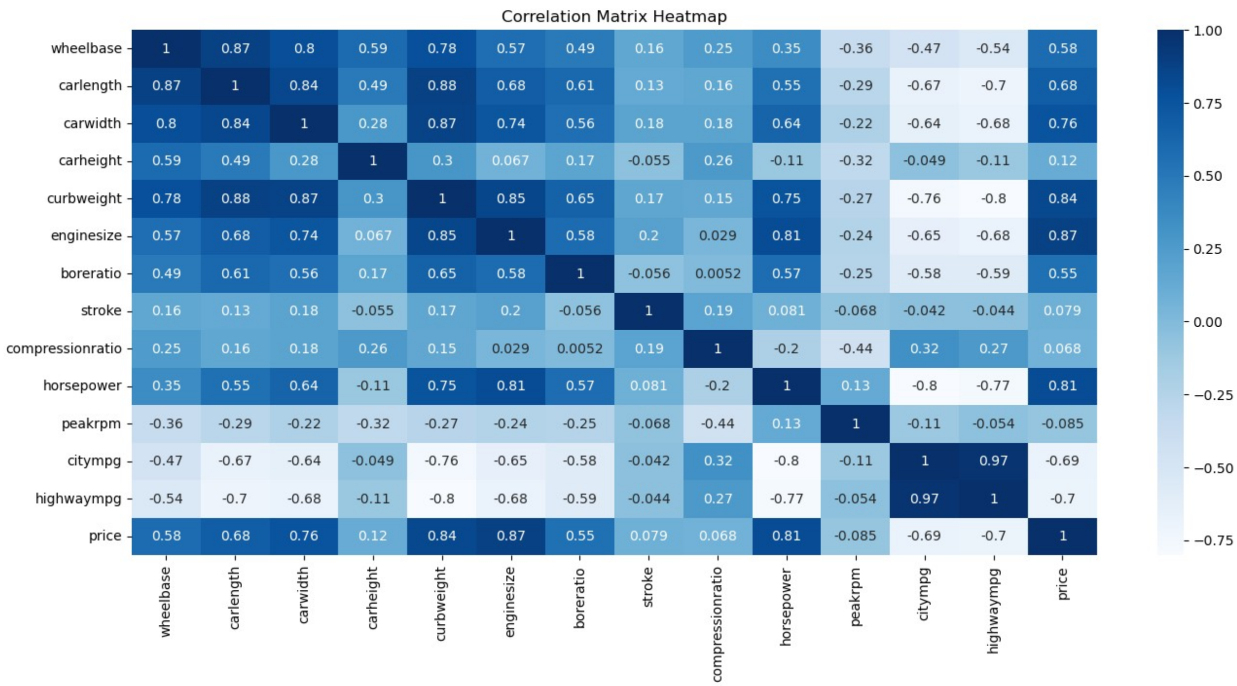
It appears that there are no missing values (NaN) in any of the columns of your DataFrame. Each column has a count of 0 missing values.

```python
# Get numerical columns
nums_cols = df.select_dtypes(include=["float64","int64"]).columns

# Calculate the number of rows and columns needed for the subplots
num_plots = len(nums_cols)
num_rows = (num_plots - 1) // 4 + 1
num_cols = min(num_plots, 4)

# Create subplots
fig, axes = plt.subplots(num_rows, num_cols, figsize=(18, 5))

# Flatten axes if there's only one row or column
if num_rows == 1 or num_cols == 1:
    axes = axes.reshape(-1)

# Plot boxplots
for i, col in enumerate(nums_cols):
    if num_rows > 1 and num_cols > 1:
        row = i // num_cols
        col_index = i % num_cols
        sns.boxplot(y=df[col], ax=axes[row, col_index])
        axes[row, col_index].set_title(col)
    else:
        sns.boxplot(y=df[col], ax=axes[i])
        axes[i].set_title(col)

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



Columns Having Outliers :
Wheelbase,Carlength,Carwidth,enginesize,stroke,compressionratio,horsepower,peakrpm,citympg,highwaympg,price

```python
# Create a heatmap of the correlation matrix
plt.figure(figsize=(16, 7))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='Blues')
plt.title('Correlation Matrix Heatmap')
plt.show()
```




```python
# Drop specified columns
columns_to_drop = ['symboling', 'stroke', 'compressionratio',
'peakrpm', 'citympg', 'highwaympg','carlength', 'curbweight']
df = df.drop(columns=columns_to_drop, axis=1)
```

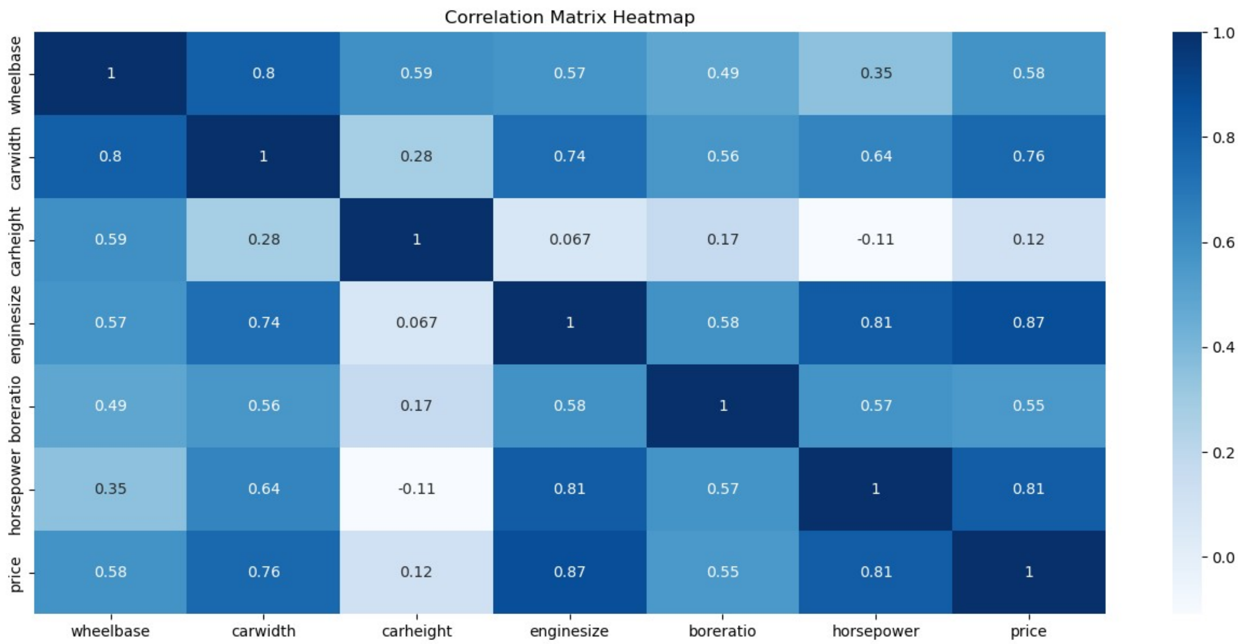
Dropping down the these cells because of having high correlation between each other as they stand by independent variable. So, mustnot have to high correlated with each other as explanatory variables

```
df

          CarName fueltype aspiration doornumber      carbody
drivewheel  \
0     alfa-romero      gas        std        two  convertible
rwd
1     alfa-romero      gas        std        two  convertible
rwd
2     alfa-romero      gas        std        two    hatchback
```

```
rwd
3          audi      gas       std       four       sedan
fwd
4          audi      gas       std       four       sedan
4wd
..          ...      ...       ...        ...         ...        ..
.
200       volvo      gas       std       four       sedan
rwd
201       volvo      gas     turbo       four       sedan
rwd
202       volvo      gas       std       four       sedan
rwd
203       volvo   diesel     turbo       four       sedan
rwd
204       volvo      gas     turbo       four       sedan
rwd

    enginelocation  wheelbase  carwidth  carheight enginetype
cylindernumber  \
0            front       88.6      64.1       48.8       dohc
four
1            front       88.6      64.1       48.8       dohc
four
2            front       94.5      65.5       52.4       ohcv
six
3            front       99.8      66.2       54.3        ohc
four
4            front       99.4      66.4       54.3        ohc
five
..             ...        ...       ...        ...        ...
...
200          front      109.1      68.9       55.5        ohc
four
201          front      109.1      68.8       55.5        ohc
four
202          front      109.1      68.9       55.5       ohcv
six
203          front      109.1      68.9       55.5        ohc
six
204          front      109.1      68.9       55.5        ohc
four

     enginesize fuelsystem  boreratio  horsepower    price
0           130       mpfi       3.47         111  13495.0
1           130       mpfi       3.47         111  16500.0
2           152       mpfi       2.68         154  16500.0
3           109       mpfi       3.19         102  13950.0
4           136       mpfi       3.19         115  17450.0
```

```
..          ...        ...        ...         ...       ...
200         141       mpfi       3.78         114   16845.0
201         141       mpfi       3.78         160   19045.0
202         173       mpfi       3.58         134   21485.0
203         145        idi       3.01         106   22470.0
204         141       mpfi       3.78         114   22625.0

[205 rows x 17 columns]
```

```python
# Create a heatmap of the correlation matrix
plt.figure(figsize=(16, 7))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='Blues')
plt.title('Correlation Matrix Heatmap')
plt.show()
```




```
df.head()

        CarName fueltype aspiration doornumber      carbody drivewheel  \
0   alfa-romero      gas        std        two  convertible        rwd

1   alfa-romero      gas        std        two  convertible        rwd

2   alfa-romero      gas        std        two    hatchback        rwd

3          audi      gas        std       four        sedan        fwd

4          audi      gas        std       four        sedan        4wd
```

```
  enginelocation  wheelbase  carwidth  carheight enginetype
cylindernumber  \
0          front       88.6      64.1       48.8       dohc
four
1          front       88.6      64.1       48.8       dohc
four
2          front       94.5      65.5       52.4       ohcv
six
3          front       99.8      66.2       54.3        ohc
four
4          front       99.4      66.4       54.3        ohc
five

   enginesize fuelsystem  boreratio  horsepower    price
0         130       mpfi       3.47         111  13495.0
1         130       mpfi       3.47         111  16500.0
2         152       mpfi       2.68         154  16500.0
3         109       mpfi       3.19         102  13950.0
4         136       mpfi       3.19         115  17450.0
```

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

```
fueltype
gas       185
diesel     20
Name: count, dtype: int64
```

The output you provided shows that there are 185 occurrences of 'gas' and 20 occurrences of 'diesel'

```
df['fueltype'] = df['fueltype'].replace({'gas':1,'diesel':2})
#transform categorical data into numerical data in the 'fueltype'
column

df['aspiration'].value_counts()

aspiration
std      168
turbo     37
Name: count, dtype: int64

df['aspiration'] = df['aspiration'].replace({'std':1,'turbo':2})

df['doornumber'].value_counts()

doornumber
four    115
two      90
Name: count, dtype: int64

df['doornumber'] = df['doornumber'].replace({'four':4,'two':2})
```

```python
df['carbody'].value_counts()
```

```
carbody
sedan          96
hatchback      70
wagon          25
hardtop         8
convertible     6
Name: count, dtype: int64
```

```python
df['carbody'] =
df['carbody'].replace(['wagon','hardtop','convertible'],'others')
df['carbody'] =
df['carbody'].replace({'sedan':1,'hatchback':2,'others':3})
```

```python
df['drivewheel'].value_counts()
```

```
drivewheel
fwd    120
rwd     76
4wd      9
Name: count, dtype: int64
```

```python
df['drivewheel'] = df['drivewheel'].replace({'fwd':1,'rwd':2,'4wd':3})
```

```python
df['enginelocation'].value_counts()
```

```
enginelocation
front    202
rear       3
Name: count, dtype: int64
```

```python
df['enginelocation'] =
df['enginelocation'].replace({'front':1,'rear':2})
```

```python
df['enginetype'].value_counts()
```

```
enginetype
ohc      148
ohcf      15
ohcv      13
dohc      12
l         12
rotor      4
dohcv      1
Name: count, dtype: int64
```

```python
df['enginetype'] =
df['enginetype'].replace(['ohcf','ohcv','dohc','l','rotor','dohcv'],'o
thers')
df['enginetype'] = df['enginetype'].replace({'ohc':1,'others':2})
```

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

```
cylindernumber
four      159
six        24
five       11
eight       5
two         4
three       1
twelve      1
Name: count, dtype: int64
```

```
df['cylindernumber'] =
df['cylindernumber'].replace(['five','eight','two','three','twelve'],'
others')
df['cylindernumber'] =
df['cylindernumber'].replace({'four':4,'six':6,'others':1})
```

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

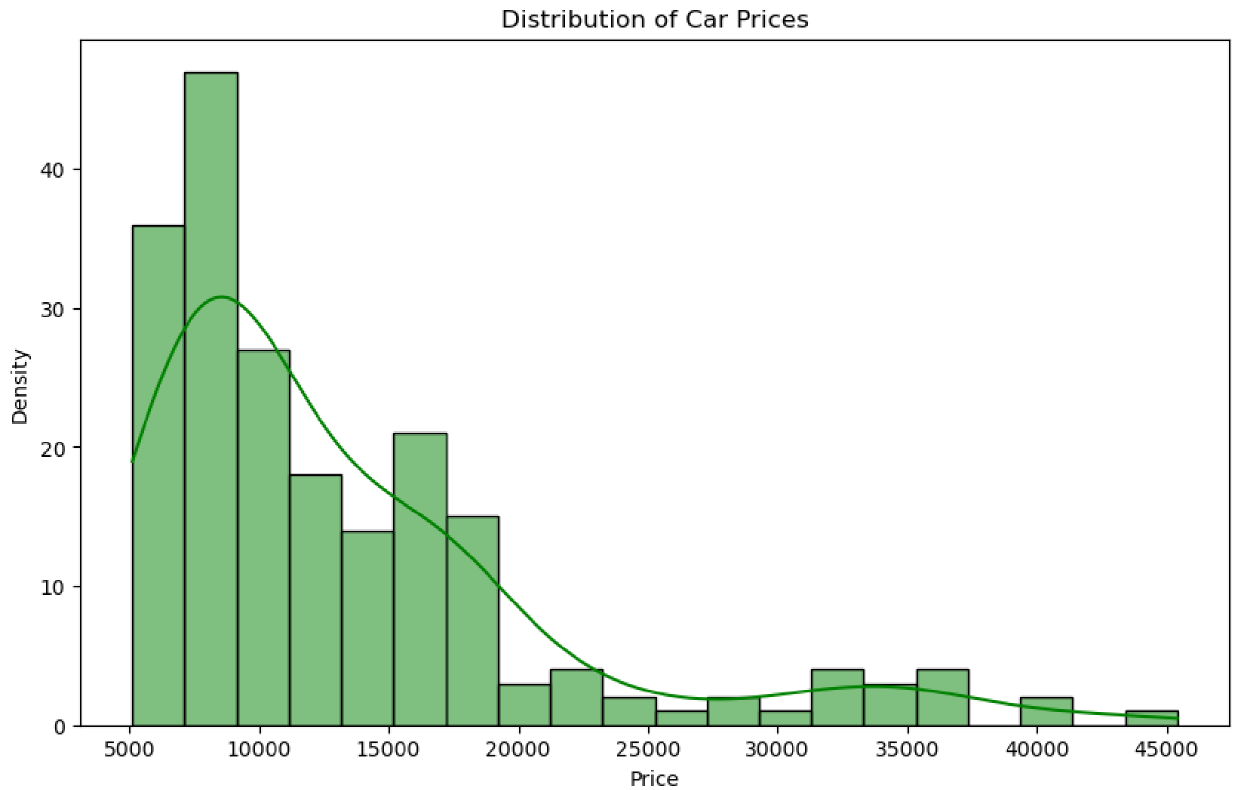
```
fuelsystem
mpfi    94
2bbl    66
idi     20
1bbl    11
spdi     9
4bbl     3
mfi      1
spfi     1
Name: count, dtype: int64
```

```
df['fuelsystem'] =
df['fuelsystem'].replace(['idi','1bbl','spdi','4bbl','mfi','spfi'],'ot
hers')
df['fuelsystem'] =
df['fuelsystem'].replace({'mpfi':1,'2bbl':2,'others':3})
```

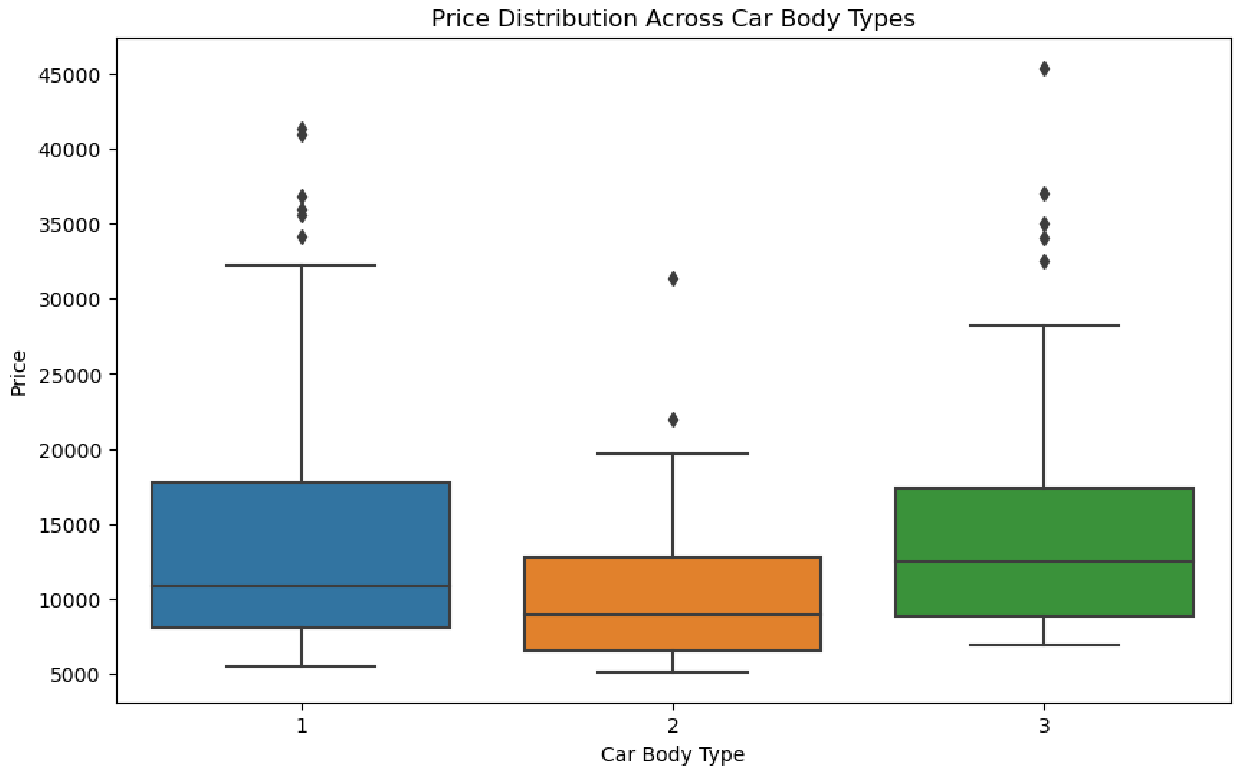
```
df.head()
```

```
       CarName  fueltype  aspiration  doornumber  carbody  drivewheel  \
0  alfa-romero         1           1           2        3           2

1  alfa-romero         1           1           2        3           2

2  alfa-romero         1           1           2        2           2

3         audi         1           1           4        1           1

4         audi         1           1           4        1           3
```

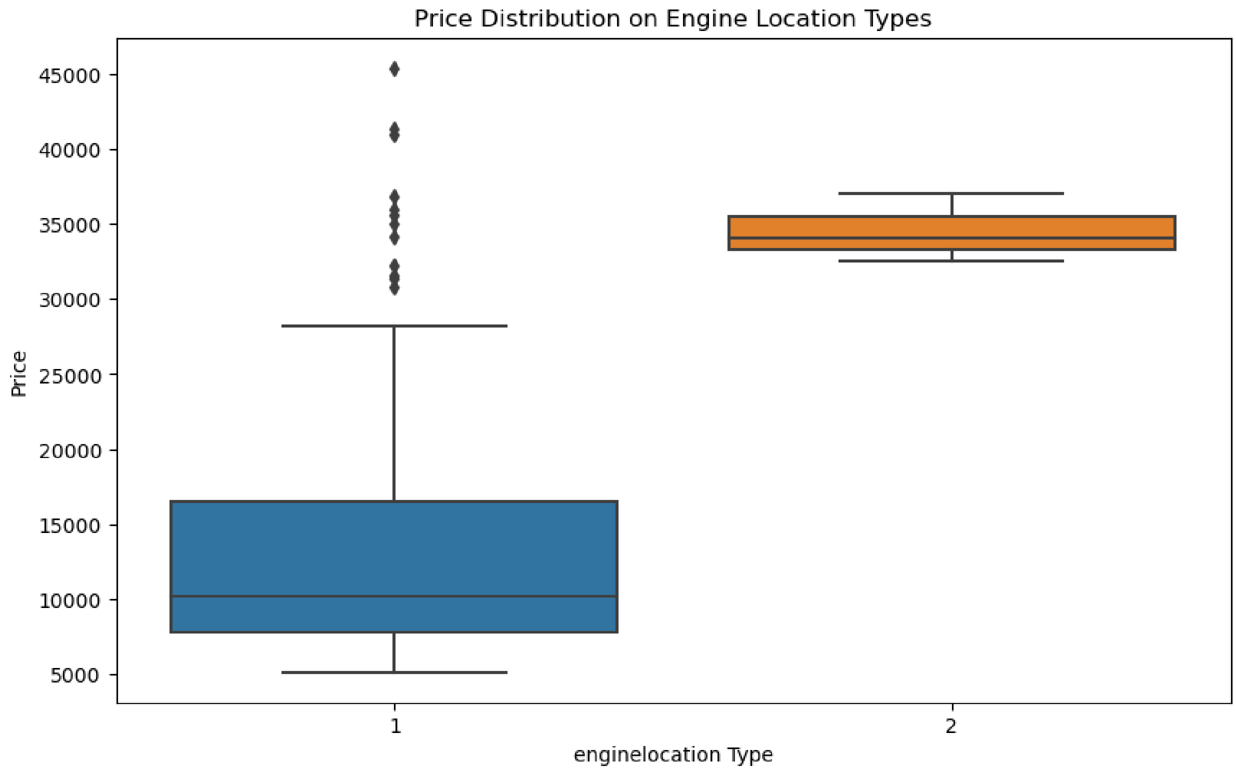
```
   enginelocation  wheelbase  carwidth  carheight  enginetype
cylindernumber  \
0               1       88.6      64.1       48.8           2
4
1               1       88.6      64.1       48.8           2
4
2               1       94.5      65.5       52.4           2
6
3               1       99.8      66.2       54.3           1
4
4               1       99.4      66.4       54.3           1
1

   enginesize  fuelsystem  boreratio  horsepower    price
0         130           1       3.47         111  13495.0
1         130           1       3.47         111  16500.0
2         152           1       2.68         154  16500.0
3         109           1       3.19         102  13950.0
4         136           1       3.19         115  17450.0

plt.figure(figsize=(10, 6))
sns.histplot(df['price'], kde=True, bins=20, color='green')
plt.title('Distribution of Car Prices')
plt.xlabel('Price')
plt.ylabel('Density')
plt.show()
```



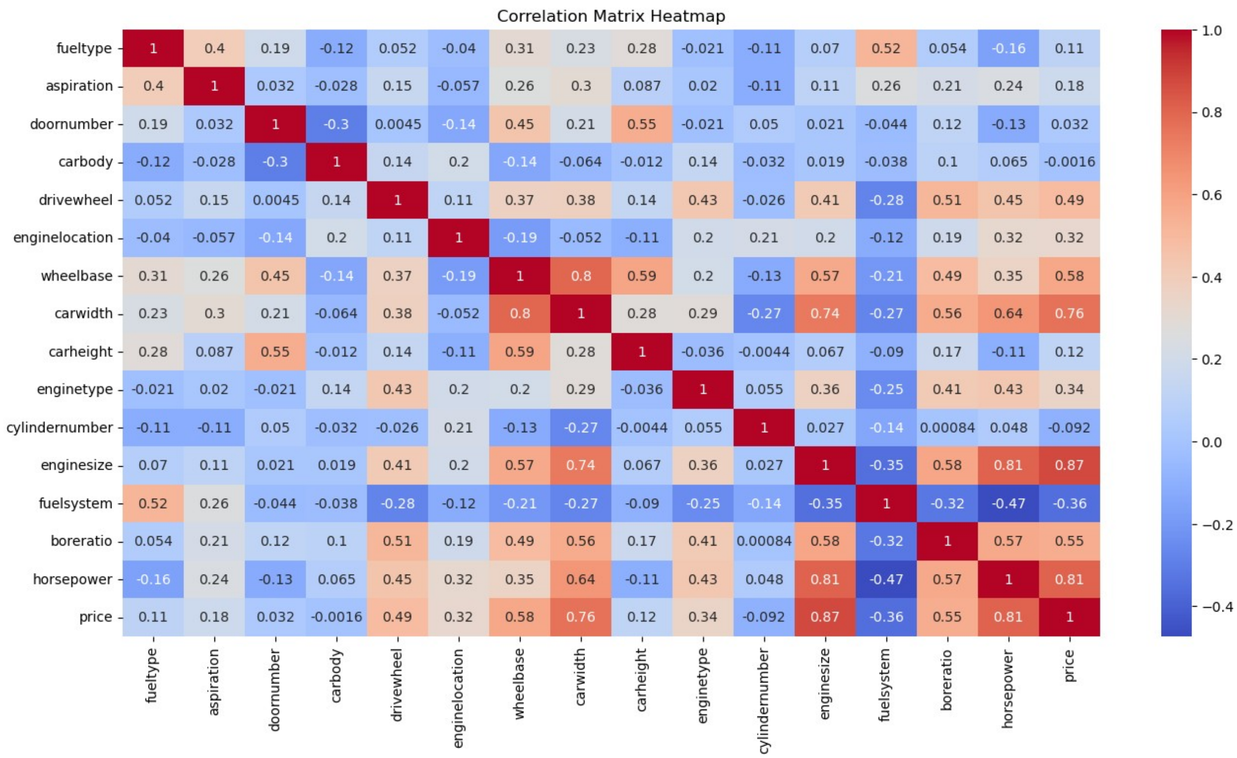

```
# Create a box plot
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='carbody', y='price')
plt.xlabel('Car Body Type')
plt.ylabel('Price')
plt.title('Price Distribution Across Car Body Types')
plt.show()
```



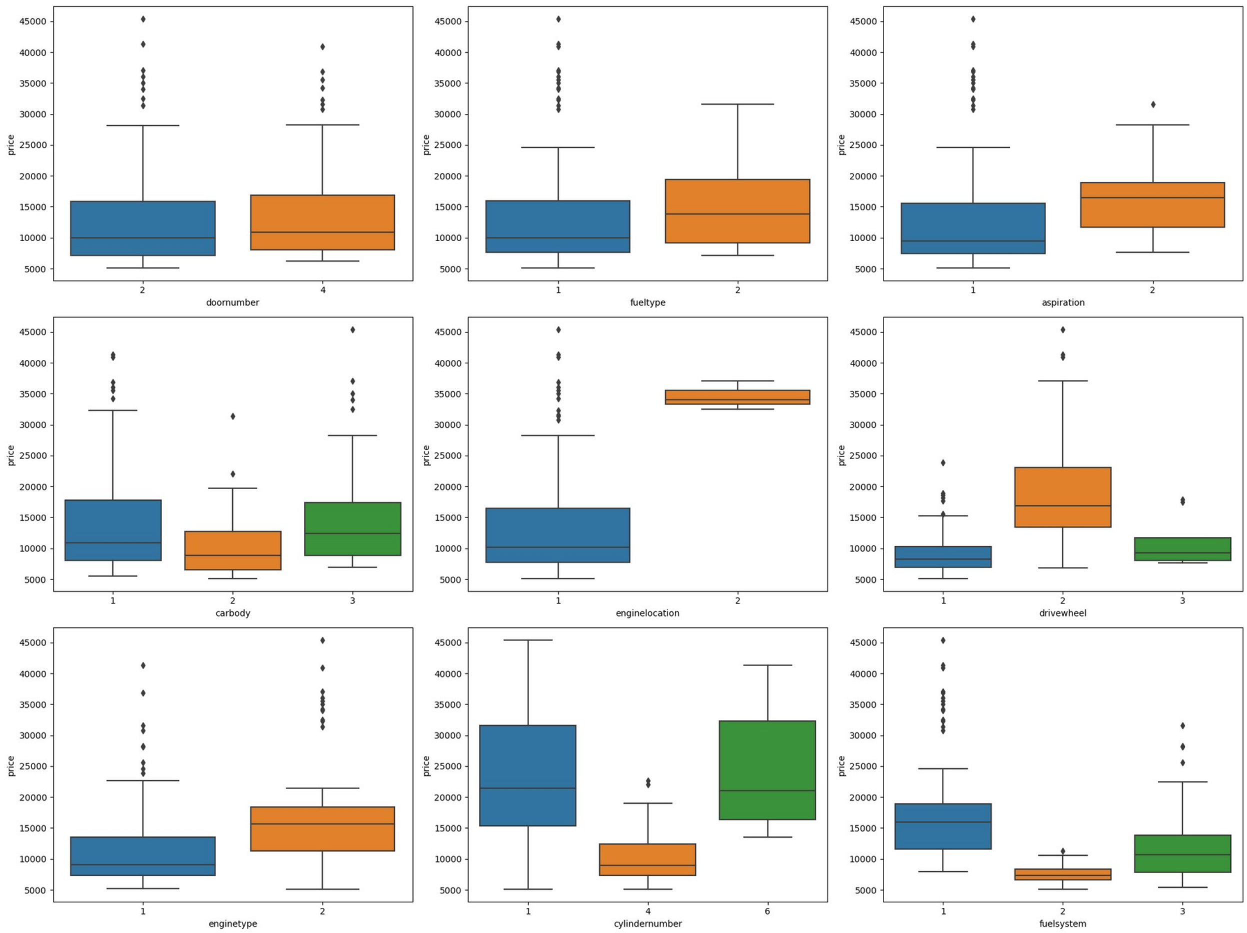

```
# Create a box plot
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='enginelocation', y='price')
plt.xlabel('enginelocation Type')
plt.ylabel('Price')
plt.title('Price Distribution on Engine Location Types')
plt.show()
```




```
# Select only numerical columns
numeric_df = df.select_dtypes(include=['float64', 'int64'])

plt.figure(figsize=(16, 8))
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix Heatmap')
plt.show()
```




```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(20, 15))

# List of categorical columns
cat_columns = ['doornumber', 'fueltype', 'aspiration', 'carbody',
'enginelocation',
               'drivewheel', 'enginetype', 'cylindernumber',
'fuelsystem']

# Loop through each categorical column and create a boxplot
for i, col in enumerate(cat_columns, 1):
    plt.subplot(3, 3, i)
    sns.boxplot(x=col, y='price', data=df)

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



Diesel fueltype cars are comparatively expensive than the cars with fueltype as gas.

All the types of carbody is relatively cheaper as compared to convertible and hardtop carbody.

The cars with rear enginelocation are way expensive than cars with front enginelocation.

rwd drivewheel car price is always very high.

Enginetype ohcv comes into higher price range cars.

The price of car is directly proportional to no. of cylinders in most cases.

```
# Import necessary libraries
from sklearn.model_selection import train_test_split  # Import train_test_split function for splitting data
from sklearn import preprocessing  # Import preprocessing module for data preprocessing tasks
from sklearn.metrics import mean_absolute_error
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
```

```
# Select numeric columns from the DataFrame
data_numeric = df.select_dtypes(include=['float64', 'int64'])

# Define independent variables (X) and dependent variable (y)
X = data_numeric.drop(columns=['price'])  # Independent variables
excluding the dependent variables
y = data_numeric['price']  # Dependent variables

data_numeric = df.select_dtypes(include=['float64', 'int64'])

# Define independent variables (X) and dependent variable (y)
X = data_numeric.drop(columns=['price'])  # Independent variables
y = data_numeric['price']  # Dependent variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test =
train_test_split(X,y,random_state=0)
model = LinearRegression()
model.fit(X_train,y_train)

y_pred = model.predict(X_test)
MSE = mean_squared_error(y_test,y_pred)
print(f"Mean Squared Error: {MSE}")
rmse = np.sqrt(MSE)
print(f"Root Mean Squared Error: {rmse}")
r2 = r2_score(y_test,y_pred) * 100
print(f"R2 Score: {r2}")
```

```
Mean Squared Error: 12240940.842944507
Root Mean Squared Error: 3498.7055953515874
R2 Score: 83.58419628988368
```

Model has an MSE of approximately 11,874,973.25, an RMSE(Root Mean Squared Error) of approximately 3,446.01, and an R^2 score of approximately 84.07%. These values suggest that your model performs reasonably well, with relatively low error and a good fit to the data. However, further analysis and comparison with alternative models may be warranted to assess the model's performance comprehensively.

```
from sklearn.preprocessing import StandardScaler

# Define feature matrix A and target vector b
A = data_numeric.drop("price", axis=1)
b = data_numeric["price"]
# Scale the features using StandardScaler
scaler = StandardScaler()
A = scaler.fit_transform(A)

# Split the scaled data into training and testing sets
X_train_s, X_test_s, y_train_s, y_test_s = train_test_split(A, b,
test_size=0.30, random_state=40)
```

```python
# Print the shapes of the scaled data
print("X_train_scaler:", X_train_s.shape)
print("X_test_scaler:", X_test_s.shape)
print("Y_train_scaler:", y_train_s.shape)
print("Y_test_scaler:", y_test_s.shape)
```

```
X_train_scaler: (143, 15)
X_test_scaler: (62, 15)
Y_train_scaler: (143,)
Y_test_scaler: (62,)
```

```python
model.fit(X_train_s,y_train_s)
y_pred = model.predict(X_test_s)


mse = mean_squared_error(y_test_s,y_pred)
print(f"Mean Squared Error: {mse}")
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error: {rmse}")
r2 = r2_score(y_test_s,y_pred) * 100
print(f"R2 Score: {r2}")
```

```
Mean Squared Error: 5784164.607479063
Root Mean Squared Error: 2405.029024248785
R2 Score: 81.75755006601055
```

Here, it shows that the given mean sum of square show that the difference between our model predictiona and actual value. And R-squared of value 81.75 shows our explainatory variables explain about 81% to predict the model shows good models measure fitness.