

# CAR PRICE ANALYSIS

2024



Prepared By:  
Hoang Trung - Data Team

## A. Giới thiệu về công ty Geely Auto

Geely Auto, viết tắt từ Zhejiang Geely Holding Group Co., Ltd, là một tập đoàn ô tô Trung Quốc có trụ sở chính tại Hàng Châu, Chiết Giang. Được thành lập vào năm 1986 bởi Li Shufu, Geely ban đầu bắt đầu như một công ty sản xuất xe máy và gia nhập ngành công nghiệp ô tô vào cuối thập niên 1990.

Công ty đã nhanh chóng phát triển và mở rộng danh mục sản phẩm, bao gồm các dòng xe du lịch, xe SUV, xe điện, và các dòng xe thương mại. Geely đã nỗ lực trong việc nâng cao chất lượng và công nghệ của sản phẩm của mình thông qua các quan hệ đối tác với các công ty quốc tế và mua lại các thương hiệu nổi tiếng như Volvo và Lotus.

Geely Auto là một trong những nhà sản xuất ô tô hàng đầu tại Trung Quốc và có sự hiện diện trên toàn cầu. Công ty chú trọng vào việc nghiên cứu và phát triển công nghệ mới, đặc biệt là trong lĩnh vực xe điện và xe tự hành, nhằm định hình tương lai của ngành công nghiệp ô tô.



## B. Tổng quan về dữ liệu

Dữ liệu mà Team Data sử dụng để phân tích và tạo lập mô hình dự đoán giá xe gồm có các cột như sau:

- **car\_ID**: ID duy nhất của mỗi quan sát.
- **symboling**: Được gán cho mức độ rủi ro bảo hiểm của xe. Một giá trị +3 chỉ ra rằng xe có rủi ro cao, -3 chỉ ra rằng xe khá an toàn.
- **CarName**: Tên loại xe (Phân loại).
- **fueltype**: Loại nhiên liệu của xe, ví dụ như xăng hoặc diesel.
- **aspiration**: Loại tăng áp sử dụng trong xe.
- **doornumber**: Số lượng cửa của xe.
- **carbody**: Thân xe.
- **drivewheel**: Bánh lái.
- **enginelocation**: Vị trí động cơ của xe.
- **wheelbase**: Khoảng cách trục của xe.
- **carlength**: Chiều dài của xe.
- **carwidth**: Chiều rộng của xe.
- **carheight**: Chiều cao của xe.
- **curbweight**: Trọng lượng của xe không bao gồm hành khách hoặc hành lý.
- **enginetype**: Loại động cơ.
- **cylindernumber**: Số lượng xi lanh trong xe.
- **enginesize**: Kích thước của động cơ.
- **fuelsystem**: Hệ thống nhiên liệu của xe.
- **boreratio**: Tỷ lệ khoan của xe.
- **stroke**: Hành trình piston hoặc dung tích bên trong động cơ.
- **compressionratio**: Tỷ số nén của xe.

- **horsepower**: Mã lực.
- **peakrpm**: Vòng quay tối đa của xe.
- **citympg**: Mức tiêu thụ nhiên liệu trong thành phố.
- **highwaympg**: Mức tiêu thụ nhiên liệu trên đường cao tốc.
- **price**: Giá của xe. (Biến cần dự đoán)

Link truy cập file ipynb được sử dụng để phân tích dữ liệu: [Link](#)

## C. Load Data

### 1. Khai báo thông số của Server để lấy dữ liệu

```
# Các thông tin kết nối đến server
server = '45.117.83.230'
database = 'DA_FINALTEST'
username = 'Student_DA_Q1'
password = '@MindXDream2023'
port = '1433'
driver = '{ODBC Driver 17 for SQL Server}'
```

### 2. Tạo Connection String

```
conn_str = f'DRIVER={driver};SERVER={server},{port};DATABASE={database};UID={username};PWD={password}'
```

### 3. Thực hiện kết nối tới server bằng pyodbc

```
# Thực hiện kết nối
try:
    conn = pyodbc.connect(conn_str)
    print("Kết nối thành công!")
except Exception as e:
    print("Không thể kết nối đến SQL Server:")
    print(e)
```

### 4. Tạo cursor

```
cursor = conn.cursor()
```

### 5. Load dữ liệu vào Data Frame

```
df = pd.read_sql("SELECT * FROM CarPrice_Assignment", conn)
```

df												
205 rows × 26 columns												
car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginelocation	...
0	1	3 alfa-romero giulia	gas	std	two	convertible	rwd	front	886.0	...	...	...
1	2	3 alfa-romero stelvio	gas	std	two	convertible	rwd	front	886.0	...	...	...
2	3	1 alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	945.0	...	...	...
3	4	2 audi 100 ls	gas	std	four	sedan	fwd	front	998.0	...	...	...
4	5	2 audi 100ls	gas	std	four	sedan	4wd	front	994.0	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...
200	201	-1 volvo 145e (sw)	gas	std	four	sedan	rwd	front	1091.0	...	...	...
201	202	-1 volvo 144ea	gas	turbo	four	sedan	rwd	front	1091.0	...	...	...
202	203	-1 volvo 244dl	gas	std	four	sedan	rwd	front	1091.0	...	...	...
203	204	-1 volvo 246	diesel	turbo	four	sedan	rwd	front	1091.0	...	...	...
204	205	-1 volvo 264gl	gas	turbo	four	sedan	rwd	front	1091.0	...	...	...

## C. EDA dữ liệu

### 1. Kiểm tra sơ bộ dữ liệu đã import

```
1 df.info()
Executed at 2024.04.12 11:21:44 in 76ms

<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   fuelytype        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
```

Trong các cột trên thì cột car\_ID sẽ không ảnh hưởng đến việc xây dựng mô hình nên ta sẽ tạo ra 1 DataFrame mới không chứa cột này.

```
df_new = df.drop(columns=['car_ID'])
```

Tiếp tục kiểm tra xem dữ liệu df\_new có null hay không?

```
df_new.isnull().sum()
```

Kết quả trả về như sau:

```

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

```

Điều đó có nghĩa là dữ liệu của chúng ta không hề có dữ liệu null.

## 2. Phân loại dữ liệu

Ta sẽ phân các feature dữ liệu thành 2 loại dữ liệu là dữ liệu dạng object và dữ liệu dạng số để phân loại đâu là biến liên tục, đâu là biến phân loại.

- Danh sách các cột đang có dữ liệu dạng chữ (object)

```

# Phân ra dữ liệu dạng chữ và dạng số và tiến hành phân tích
object_cols = df_new.select_dtypes(include='object').columns.tolist()

object_cols

```

Executed at 2024.04.08 20:45:00 in 221ms

```

['CarName',
 'fueltype',
 'aspiration',
 'doornumber',
 'carbody',
 'drivewheel',
 'enginelocation',
 'enginetype',
 'cylindernumber',
 'fuelsystem']

```

- Danh sách các cột đang có dữ liệu dạng số:

```
numerical_cols = df_new.select_dtypes(exclude='object').columns.tolist()

numerical_cols
Executed at 2024.04.08 20:45:00 in 366ms

['symboling',
 'wheelbase',
 'carlength',
 'carwidth',
 'carheight',
 'curbweight',
 'enginesize',
 'boreratio',
 'stroke',
 'compressionratio',
 'horsepower',
 'peakrpm',
 'citympg',
 'highwaympg',
 'price']
```

Ta sẽ xem xem các cột dạng chữ và dạng số có bao nhiêu dữ liệu không trùng lặp. Đây là cơ sở để xác định xem các giá trị là giá trị liên tục hay giá trị phân loại.

```
print_info_all_columns(df_new, object_cols)
Executed at 2024.04.08 20:45:00 in 118ms

1/CarName  147:
2/fueltype  2: ['gas' 'diesel']
3/aspiration  2: ['std' 'turbo']
4/doornumber  2: ['two' 'four']
5/carbody    5: ['convertible' 'hatchback' 'sedan' 'wagon' 'hardtop']
6/drivewheel  3: ['rwd' 'fwd' '4wd']
7/enginelocation  2: ['front' 'rear']
8/enginetype   7: ['dohc' 'ohcv' 'ohc' 'l' 'rotor' 'ohcf' 'dohcv']
9/cylindernumber  7: ['four' 'six' 'five' 'three' 'twelve' 'two' 'eight']
10/fuelsystem   8: ['mpfi' '2bbl' 'mfi' '1bbl' 'spfi' '4bbl' 'idi' 'spdi']
```

Nhìn vào đây ta có thể rút ra các cột kiểu chữ hầu như đều là biến phân loại. Tuy nhiên với cột **cylindernumber** thì là dữ liệu ordinal nên ta sẽ mapping lại khi encode dữ liệu.

Các cột còn lại có phải ordinal hay không thì ta sẽ tiến hành phân tích và xác định sau.

Dưới đây là thông tin về các giá trị trùng lặp dạng số (Lưu ý là các cột nào có quá nhiều giá trị không trùng có nghĩa là cột đó chắc chắn là 1 giá trị liên tục, vì thế kết quả sẽ không được hiển thị ra mà chỉ hiển thị số lượng giá trị không trùng lặp thôi. Ở đây mình đang đặt ngưỡng cho nó là 150)

```
print_info_all_columns_numbers(df_new, numerical_cols)
Executed at 2024.04.08 20:45:00 in 218ms

1/symboling 6: [ 3  1  2  0 -1 -2]
2/wheelbase 53: [ 886.  945.  998.  994.  1058.  995.  1012.  1035.  110.  884.  937.  1033.
  959.  866.  965.  943.  96.  113.  102.  931.  953.  988.  1049.  1067.
  1156.  966.  1209.  112.  1027.  93.  963.  951.  972.  1004.  913.  992.
  1079.  1142.  108.  895.  984.  961.  991.  933.  97.  969.  957.  1024.
  1029.  1045.  973.  1043.  1091.]
3/carlength 75: [1688.  1712.  1766.  1773.  1927.  1782.  1768.  189.  1938.  197.  1411.  1559.
  1588.  1573.  1746.  1732.  1446.  150.  1634.  1571.  1675.  1754.  1691.  1707.
  1726.  1996.  1917.  1591.  1668.  169.  1778.  175.  1909.  1875.  2026.  1803.
  2081.  1992.  1784.  173.  1724.  1653.  1702.  1656.  1624.  1734.  1817.  1846.
  1785.  1867.  1989.  1673.  1689.  1757.  1815.  1866.  1569.  1579.  172.  1735.
  1736.  1587.  1697.  1663.  1687.  1762.  1756.  1835.  1878.  1717.  1593.  1657.
  1802.  1831.  1888.]
4/carwidth 44: [641.  655.  662.  664.  663.  714.  679.  648.  669.  709.  603.  636.  638.  646.
  639.  64.  652.  625.  66.  618.  696.  706.  642.  657.  665.  661.  703.  717.
  705.  72.  68.  644.  654.  684.  683.  65.  723.  666.  634.  656.  677.  672.
  689.  688.]
5/carheight 49: [488.  524.  543.  531.  557.  559.  52.  537.  563.  532.  508.  506.  598.  502.
  526.  545.  583.  533.  541.  51.  535.  514.  528.  478.  496.  555.  544.  565.
  587.  549.  567.  554.  548.  494.  516.  547.  551.  561.  497.  56.  505.  552.
  525.  53.  591.  539.  556.  562.  575.]
6/curbweight 171:
7/engine-size 44: [130  152  109  136  131  108  164  209  61  90  98  122  156  92  79  110  111  119
  258  326  91  70  80  140  134  183  234  308  304  97  103  120  181  151  194  203
  132  121  146  171  161  141  173  145]
8/boreratio 38: [347.  268.  319.  313.  35.  331.  362.  291.  303.  297.  334.  36.  292.  315.
  343.  363.  354.  308.  333.  339.  376.  358.  346.  38.  378.  317.  335.  359.
  299.  37.  361.  394.  374.  254.  305.  327.  324.  301.]
9/stroke 37: [ 268.  347.  34.  28.  319.  339.  303.  311.  323.  346.  39.  341.
  307.  358.  417.  276.  315.  3255.  316.  364.  31.  335.  312.  386.
  329.  327.  352.  219.  321.  29.  207.  236.  264.  308.  35.  354.
  287.]
10/compressionratio 32: [ 9.  10.  8.  85.  83.  7.  88.  95.  96.  941.  94.  76.  92.  101.
  91.  81.  115.  86.  227.  22.  215.  75.  219.  78.  84.  21.  87.  931.
  93.  77.  225.  23.]
```

```

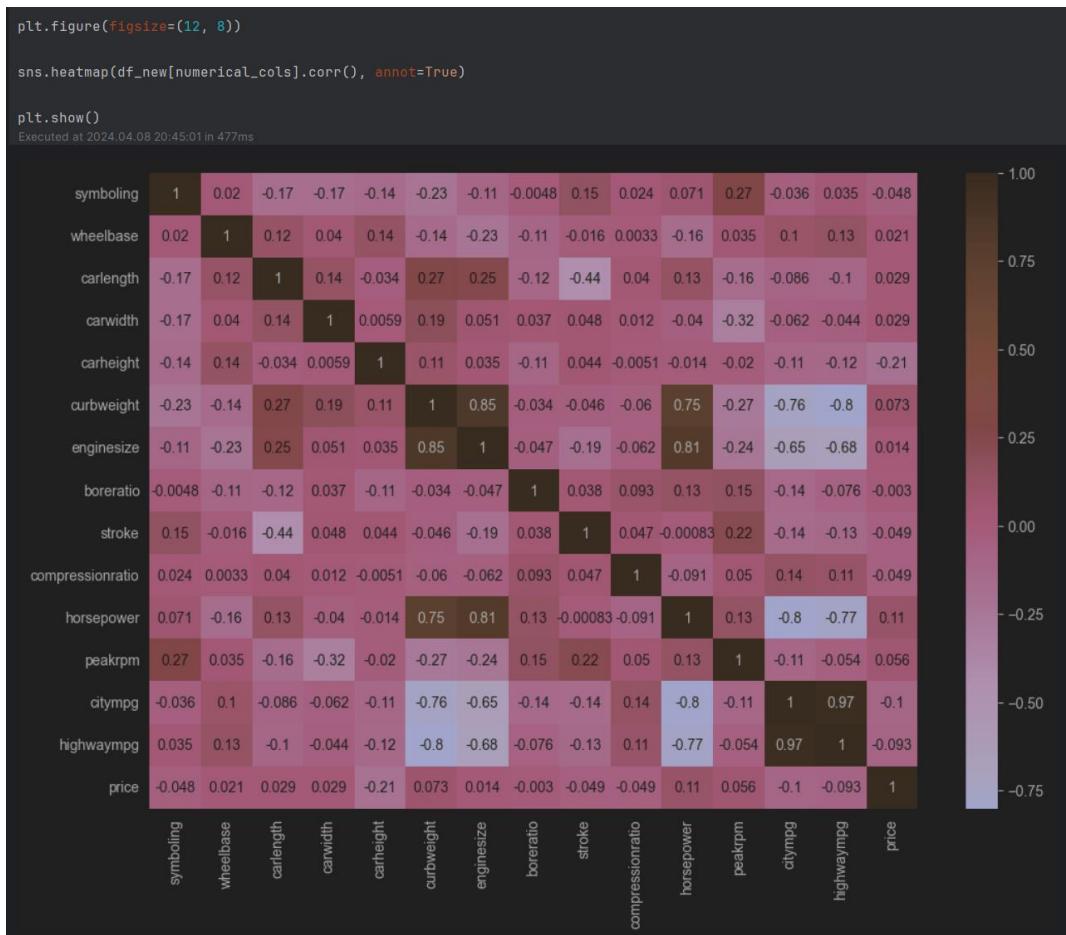
11/horsepower 59: [111 154 102 115 110 140 160 101 121 182 48 70 68 88 145 58 76 60
86 100 78 90 176 262 135 84 64 120 72 123 155 184 175 116 69 55
97 152 200 95 142 143 207 288 73 82 94 62 56 112 92 161 156 52
85 114 162 134 106]
12/peakrpm 23: [5000 5500 5800 4250 5400 5100 4800 6000 4750 4650 4200 4350 4500 5200
4150 5600 5900 5750 5250 4900 4400 6600 5300]
13/citympg 29: [21 19 24 18 17 16 23 20 15 47 38 37 31 49 30 27 25 13 26 36 22 14 45 28
32 35 34 29 33]
14/highwaympg 30: [27 26 30 22 25 20 29 28 53 43 41 38 24 54 42 34 33 31 19 17 23 32 39 18
16 37 50 36 47 46]
15/price 189:

```

Nhìn vào đây ta có thể xác định cột symboling mặc dù là dữ liệu kiểu số nhưng vẫn là 1 biến phân loại.

Còn lại tất cả các cột khác đều là dữ liệu kiểu liên tục.

### 3. Kiểm tra sự tương quan giữa các feature dạng số



Ta sẽ quan tâm tới trường sự tương quan của các trường dạng số với trường price vì price là biến mà chúng ta cần phải dự đoán. Bạn có thể thấy sự tương quan của price với các trường khác là rất thấp. Gần như đều dưới 10%.

Nếu sự tương quan thấp như thế này thì chúng ta không thể nào dự đoán bài toán trên được. Đây là 1 điểm khá vô lý. Vì thế ta sẽ đi xem xét dữ liệu về price trong bảng dữ liệu mà chúng ta đang có.

```
print(f'Min Car Price = {df_new["price"].min()}')
```

Executed at 2024.04.12 11:35:00 in 198ms

Min Car Price = 5118.0

```
print(f'Max Car Price = {df_new["price"].max()}')
```

Executed at 2024.04.12 11:35:00 in 408ms

Max Car Price = 17859167.0

```
print(f'Mean Car Price = {df_new["price"].mean()}')
```

Executed at 2024.04.12 11:35:01 in 172ms

Mean Car Price = 102468.9512195122

```
print(f'Median Car Price = {df_new["price"].median()}')
```

Executed at 2024.04.12 11:35:01 in 242ms

Median Car Price = 10595.0

Khi xem xét qua trường price ta có thể thấy 1 sự chênh lệch khá lớn giữa mean price và median price => khả năng chúng ta sẽ có nhiều outlier trong dữ liệu của mình.

```
all_infomation_continuous_variable(df_new, 'price')
```

Executed at 2024.04.12 11:35:01 in 139ms

Các thông số thống kê cơ bản của price

```
count      2.050000e+02
mean      1.024690e+05
std       1.246484e+06
min       5.118000e+03
25%       7.788000e+03
50%       1.059500e+04
75%       1.655800e+04
max       1.785917e+07
Name: price, dtype: float64
price Median: 10595.0
```

```
price Q1: 7788.0
price Q3: 16558.0
price IQR: 8770.0
price Range: 17854049.0
price Var: 1553723335783.5476
price Std: 1246484.3905093828
price Skew: 14.310115796985
price Kurtosis: 204.85084648590842
```

Phân phối lệch phải

-----

Kiểm tra số lượng Outlier!

Số lượng Outlier bên trên là: 18

Số lượng Outlier bên dưới là: 0

Phần trăm Outlier của cột price là 8.780487804878048

Ta có thể thấy số lượng outlier là không nhiều (8.78%) và tất cả đều là Outlier bên trên (18 giá trị)

```
# Khá khó nhìn vì có những outlier quá lớn => ta sẽ loại bỏ các outlier này
sns.boxplot(df_new['price'])

plt.show()
```

Executed at 2024.04.12 11:35:01 in 238ms



Với biểu đồ boxplot vì tồn tại các giá trị outlier quá lớn nên ta không thể nhìn thấy box.

Ta sẽ lấy ra các giá trị outlier này và kiểm tra vấn đề.

```
df_new.sort_values(by='price', ascending=False).head(18)[['CarName', 'price']]
```

Executed at 2024.04.12 11:44:15 in 261ms

	CarName	price
9	audi 5000s (diesel)	17859167.0
129	porsche cayenne	314005.0
44	isuzu D-Max	89165.0
45	isuzu D-Max V-Cross	89165.0
74	buick regal sport coupe (turbo)	45400.0
16	bmw x5	41315.0
73	buick century special	40960.0
128	porsche boxter	37028.0
17	bmw x3	36880.0
49	jaguar xk	36000.0
48	jaguar xf	35550.0
72	buick skylark	35056.0
71	buick opel isuzu deluxe	34184.0
127	porsche cayenne	34028.0
126	porcschce panamera	32528.0
47	jaguar xj	32250.0
70	buick skyhawk	31600.0
15	bmw x4	30760.0

Giờ việc ta cần làm là đi kiểm tra giá từng loại xe xem có vấn đề nhầm lẫn gì khi thu thập dữ liệu hay không trước khi quyết định bỏ đi các dữ liệu này trước khi xây dựng model.



Dealership

**1987 Audi 5000**

1987 Audi 5000S, 5cyl, 2.3L, 5 Speed, all power, low miles  
58700k - Excellent condition. Run and dri ...

**\$12,995**



Dealership

**1981 Audi 5000**

1981 Audi 5000s 1981 diesel stick shift with only original 52K !!  
Car is in original factory conditi ...

**\$19,995**

Ví dụ với xe audi 5000s chạy dầu Diesel thì giá trung bình luôn dưới 20.000 USD. Trong khi giá của Data chúng ta có là 17.859.167 USD => quá vô lý nên ta sẽ loại bỏ giá trị này khi xây dựng model.

The price of the 2024 Porsche Cayenne starts at \$80,850 and goes up to \$100,750 depending on the trim and options.

Cayenne	\$80,850
Cayenne E-Hybrid	\$93,350
Cayenne S	\$97,350
Cayenne S E-Hybrid	\$100,750
0      \$25k      \$50k      \$75k      \$100k      \$125k      \$150k      \$175k	

Tiếp theo là giá của Porsche Cayenne chỉ nằm trong khoảng 80.850 USD đến 100.750 USD trong khi giá mà ta đang có trong Data là 314.005 USD => quá vô lý nên ta sẽ loại bỏ giá trị này khi xây dựng model.

Price in \$: 27,620



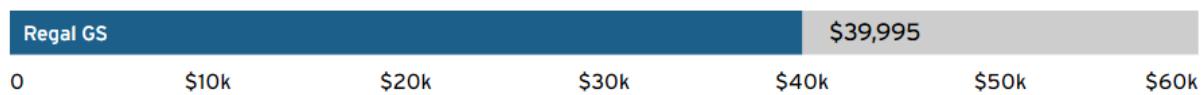
Total hits: 10795

[Full Specs](#) [Compare](#)

Với isuzu D-Max và isuzu D-Max V-Cross thì giá cao nhất cũng chỉ tầm 27.6k USD. Trong khi giá trong Data là 89.165 USD => cao hơn quá nhiều nên ta sẽ loại bỏ giá trị này khi xây dựng model.

## Pricing and Which One to Buy

The price of the 2020 Buick Regal GS starts at \$39,995.



Giá của 1 chiếc xe Buick Regal bắt đầu từ khoảng 39.995 USD nên giá trong Data mà chúng ta có là 45.400 USD là hợp lý nên Outlier này sẽ được giữ lại.

Tương tự ta sẽ kiểm tra với các mã còn lại và có thể thấy những mức giá trên tuy là outlier nhưng vẫn khớp với giá trị của chiếc xe đó trong thực tế.

Vì thế ở đây ta sẽ chỉ loại bỏ đi 4 giá trị là các Outlier không hợp lý đó là:

Car Name	Price
audi 5000s (diesel)	17859167.0
porsche cayenne	314005.0
isuzu D-Max	89165.0
isuzu D-Max V-Cross	89165.0

```
df_new.drop([129, 45, 44, 9], axis=0, inplace=True)
```

Và dưới đây là kết quả sau khi remove đi các outlier này.

```
df_new.sort_values(by='price', ascending=False)[['CarName', 'price']]
```

	CarName	price
74	buick regal sport coupe (turbo)	45400.0
16	bmw x5	41315.0
73	buick century special	40960.0
128	porsche boxter	37028.0
17	bmw x3	36880.0
49	jaguar xk	36000.0
48	jaguar xf	35550.0
72	buick skylark	35056.0
71	buick opel isuzu deluxe	34184.0
127	porsche cayenne	34028.0

Các outlier đều đã bị loại bỏ.

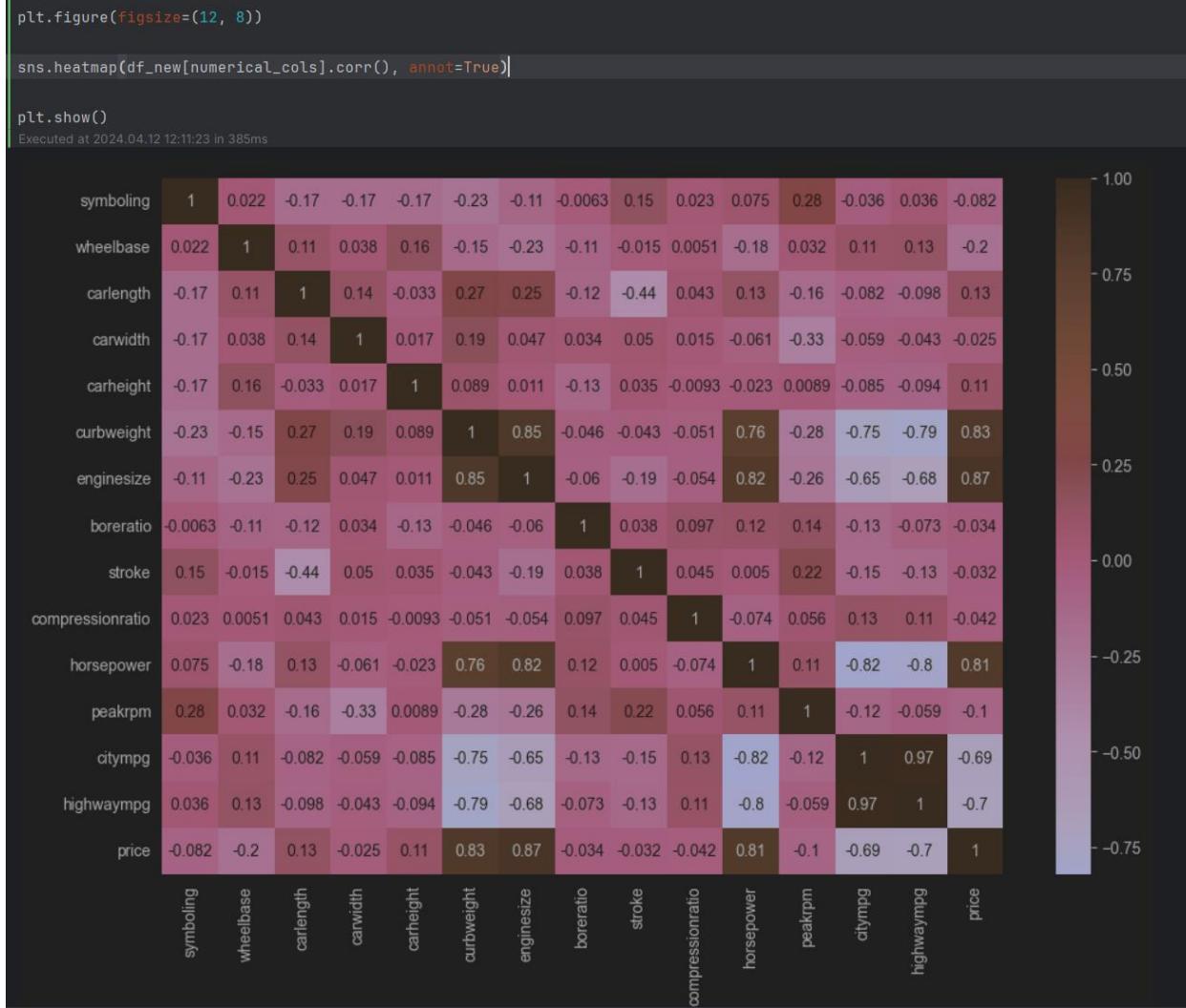
Ta sẽ vẽ lại biểu đồ boxplot và vẽ lại heatmap thể hiện sự tương quan của các trường khác tới trường price.

```
# nhìn vào đây có thể thấy các xe có giá nằm trong khoảng từ 80k đến 170k
sns.boxplot(df_new['price'])
```

```
plt.show()
```

```
Executed at 2024.04.12 12:10:02 in 144ms
```





Sau khi loại bỏ các outlier bất hợp lý thì ta có thể thấy có khá nhiều trường có sự tương quan lớn tới price đó là.

- **curbweight** (độ nặng của xe – tương quan thuận)
- **enginesize** (kích thước của động cơ – tương quan thuận)
- **horsepower** (mã lực của xe – tương quan thuận)
- **citympg** (mức tiết kiệm nhiên liệu khi đi trong thành phố - tương quan nghịch)
- **highwaympg** (mức tiết kiệm nhiên liệu khi đi trên cao tốc – tương quan nghịch)

Đây là 5 feature dạng số ta sẽ sử dụng khi xây dựng mô hình dự đoán giá xe.

### 3.1. Phân tích sự liên quan giữa độ nặng của xe (curbweight) và giá xe

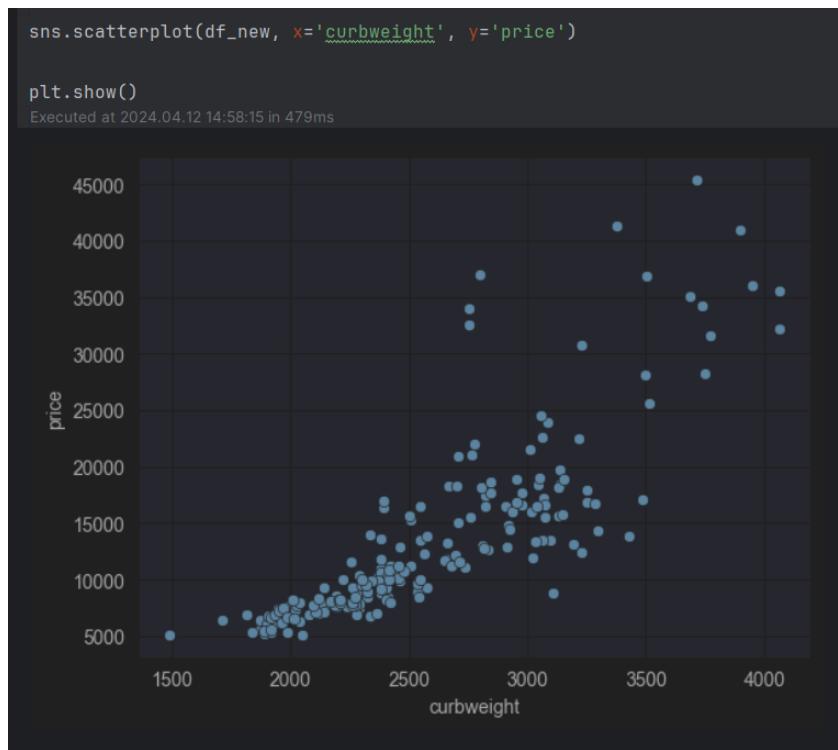
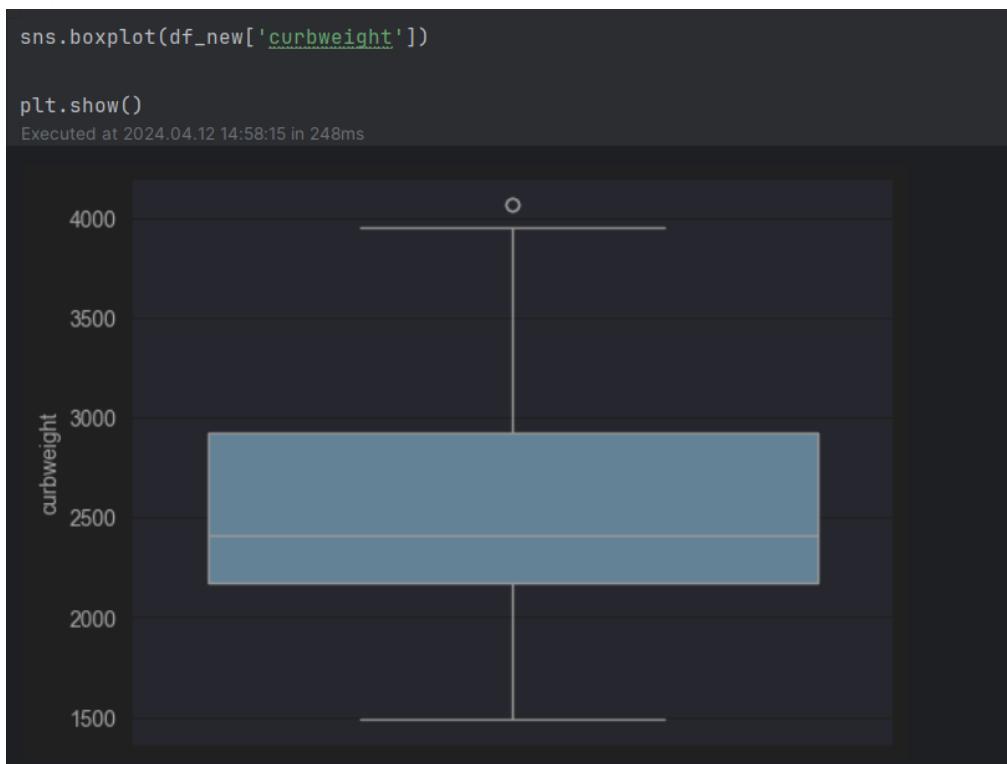
```
all_infomation_continuous_variable(df_new, 'curbweight')
Executed at 2024.04.12 12:16:03 in 262ms

Các thông số thống kê cơ bản của curbweight
count      201.000000
mean      2555.666667
std       517.296727
min      1488.000000
25%      2169.000000
50%      2414.000000
75%      2926.000000
max      4066.000000
Name: curbweight, dtype: float64
curbweight Median: 2414.0
curbweight Mode: 0    2385
Name: curbweight, dtype: int64
curbweight Q1: 2169.0
curbweight Q3: 2926.0
curbweight IQR: 757.0
curbweight Range: 2578
curbweight Var: 267595.90333333367
curbweight Std: 517.2967265828518
curbweight Skew: 0.7058035875297635
curbweight Kurtosis: 0.034915576048686336
Phân phối lệch phải
-----
Kiểm tra số lượng Outlier!
Số lượng Outlier bên trên là: 2
Số lượng Outlier bên dưới là: 0
Phần trăm Outlier của cột curbweight là 0.9950248756218906
```

Số lượng Outlier không đáng kể (Dưới 1%) nên ta sẽ không cần phải loại bỏ các outlier.



Cân nặng xe nằm trong khoảng từ 1750 đến khoảng 3250.



Ngoài ra khi nhìn vào biểu đồ scatter plot ta có thể thấy xe càng nặng thì giá sẽ càng cao.

### 3.2. Phân tích sự liên quan giữa Kích thước động cơ của xe (enginesize) và giá xe

```
all_infomation_continuous_variable(df_new, 'enginesize')
Executed at 2024.04.12 14:58:19 in 271ms

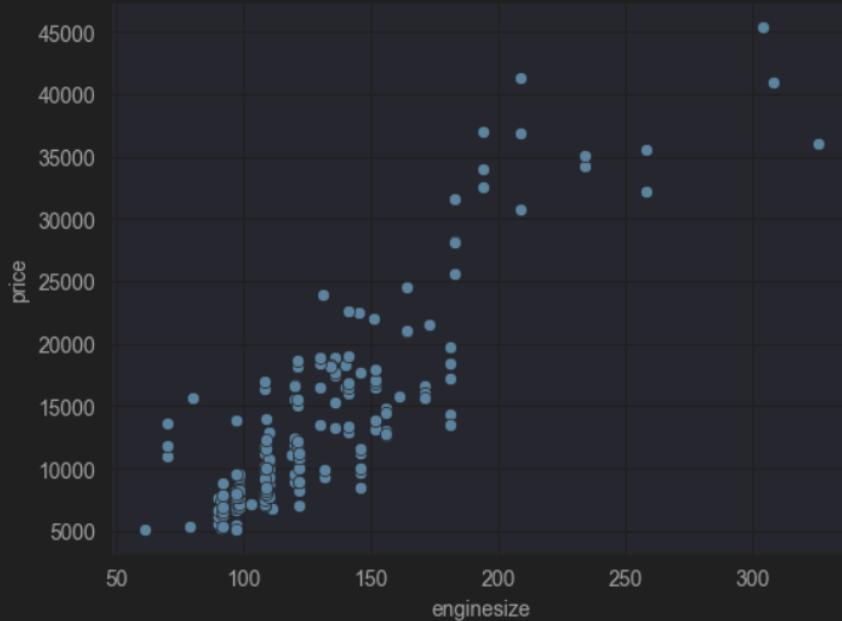
Các thông số thống kê cơ bản của enginesize
count      201.000000
mean      126.875622
std       41.546834
min       61.000000
25%      98.000000
50%     120.000000
75%     141.000000
max      326.000000
Name: enginesize, dtype: float64
enginesize Median: 120.0
enginesize Mode: 0      92
1     122
Name: enginesize, dtype: int64
enginesize Q1: 98.0
enginesize Q3: 141.0
enginesize IQR: 43.0
enginesize Range: 265
enginesize Var: 1726.1394527363163
enginesize Std: 41.54683444904457
enginesize Skew: 1.9791441966363523
enginesize Kurtosis: 5.497490766643349
Phân phối lệch phải
-----
Kiểm tra số lượng Outlier!
Số lượng Outlier bên trên là: 10
Số lượng Outlier bên dưới là: 0
Phần trăm Outlier của cột enginesize là 4.975124378109453
```



```
sns.scatterplot(df_new, x='enginesize', y='price')

plt.show()
```

Executed at 2024.04.12 14:58:19 in 288ms



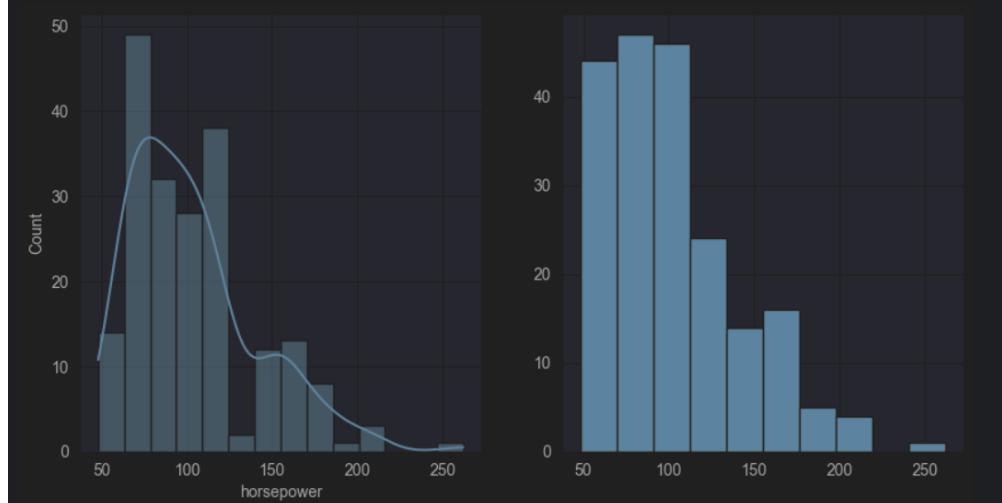
Ta có thể thấy là động cơ xe càng lớn thì giá thành sẽ càng cao.

### 3.3. Phân tích sự liên quan giữa mã lực của xe (horse power) và giá xe

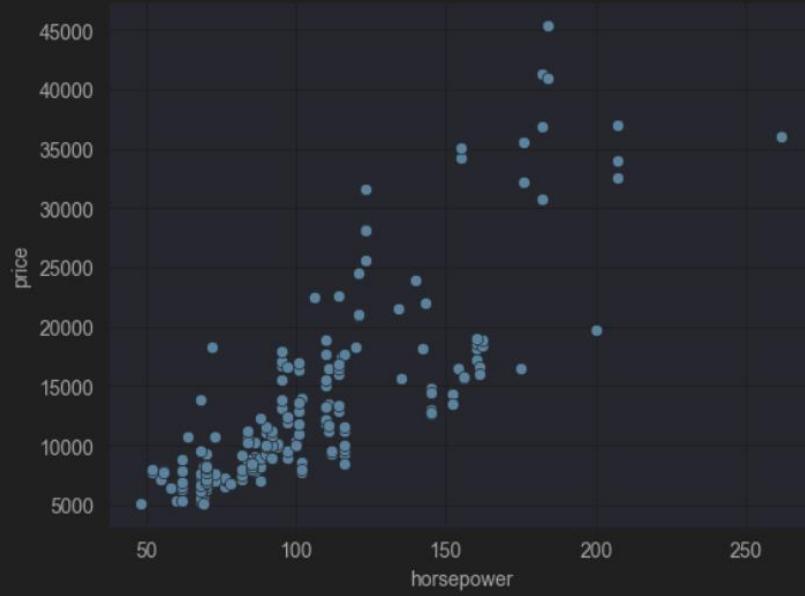
```
all_infomation_continuous_variable(df_new, 'horsepower')
Executed at 2024.04.12 14:58:22 in 275ms

Các thông số thống kê cơ bản của horsepower
count      201.000000
mean      103.263682
std       37.389372
min       48.000000
25%      70.000000
50%      95.000000
75%     116.000000
max      262.000000
Name: horsepower, dtype: float64
horsepower Median: 95.0
horsepower Mode: 0    68
Name: horsepower, dtype: int64
horsepower Q1: 70.0
horsepower Q3: 116.0
horsepower IQR: 46.0
horsepower Range: 214
horsepower Var: 1397.9651243781086
horsepower Std: 37.38937181042373
horsepower Skew: 1.1553715777806066
horsepower Kurtosis: 1.327302314200054
Phân phối lệch phải
-----
Kiểm tra số lượng Outlier!
Số lượng Outlier bên trên là: 5
Số lượng Outlier bên dưới là: 0
Phần trăm Outlier của cột horsepower là 2.4875621890547266
```

```
hist_plot_continuous_variable(df_new, 'horsepower')
Executed at 2024.04.12 14:58:22 in 317ms
```



```
# mã lực càng cao thì xe có giá càng cao  
sns.scatterplot(df_new, x='horsepower', y='price')  
  
plt.show()  
Executed at 2024.04.12 14:58:23 in 336ms
```



Ta có thể thấy xe có mã lực càng lớn thì sẽ có giá càng cao.

### 3.4. Phân tích sự liên quan giữa hiệu suất nhiên liệu của 1 xe hơi trong điều kiện lái xe trong thành phố hoặc đô thị (citympg) và giá xe

```
all_infomation_continuous_variable(df_new, 'citympg')
Executed at 2024.04.12 14:58:23 in 64ms

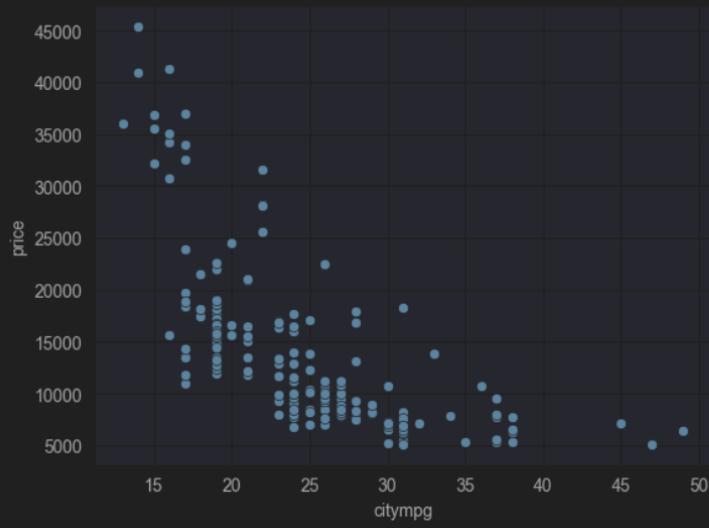
Các thông số thống kê cơ bản của citympg
count      201.000000
mean      25.179104
std       6.423220
min      13.000000
25%      19.000000
50%      24.000000
75%      30.000000
max      49.000000
Name: citympg, dtype: float64
citympg Median: 24.0
citympg Mode: 0      31
Name: citympg, dtype: int64
citympg Q1: 19.0
citympg Q3: 30.0
citympg IQR: 11.0
citympg Range: 36
citympg Var: 41.25776119402983
citympg Std: 6.4232204690505394
citympg Skew: 0.6804334707346078
citympg Kurtosis: 0.7539680878039432
Phân phối lệch phải
-----
Kiểm tra số lượng Outlier!
Số lượng Outlier bên trên là: 2
Số lượng Outlier bên dưới là: 0
Phần trăm Outlier của cột citympg là 0.9950248756218906
```

```
hist_plot_continuous_variable(df_new, 'citympg')
Executed at 2024.04.12 14:58:24 in 408ms
```



```
sns.scatterplot(df_new, x='citympg', y='price')

plt.show()
Executed at 2024.04.12 14:58:24 in 319ms
```



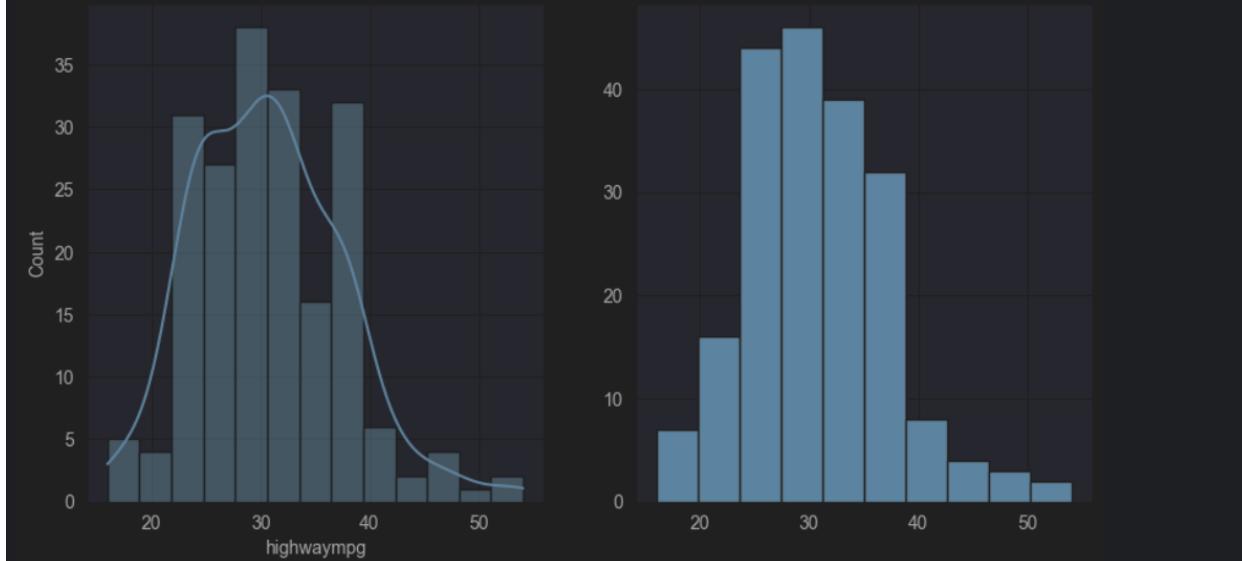
Ta có thể thấy là hiệu suất sử dụng nhiên liệu trong thành phố và đô thị càng thấp thì giá xe càng cao.

### 3.5. Phân tích sự liên quan giữa hiệu suất nhiên liệu của 1 xe hơi trong điều kiện lái xe trên đường cao tốc hoặc xa lộ và giá xe

```
all_infomation_continuous_variable(df_new, 'highwaympg')
Executed at 2024.04.12 14:58:24 in 253ms

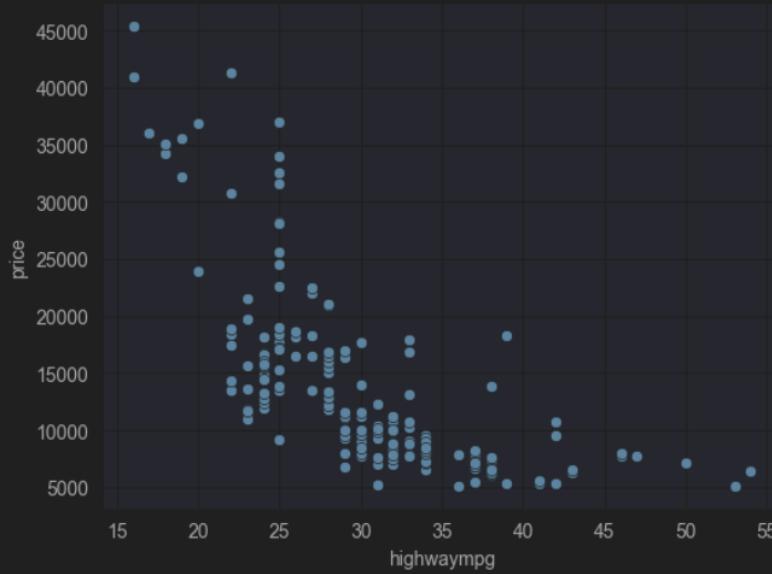
Các thông số thống kê cơ bản của highwaympg
count      201.000000
mean       30.686567
std        6.815150
min        16.000000
25%        25.000000
50%        30.000000
75%        34.000000
max        54.000000
Name: highwaympg, dtype: float64
highwaympg Median: 30.0
highwaympg Mode: 0    25
Name: highwaympg, dtype: int64
highwaympg Q1: 25.0
highwaympg Q3: 34.0
highwaympg IQR: 9.0
highwaympg Range: 38
highwaympg Var: 46.44626865671642
highwaympg Std: 6.815149936480959
highwaympg Skew: 0.5495071459179158
highwaympg Kurtosis: 0.5611711398011869
Phân phối lệch phải
-----
Kiểm tra số lượng Outlier!
Số lượng Outlier bên trên là: 3
Số lượng Outlier bên dưới là: 0
Phần trăm Outlier của cột highwaympg là 1.4925373134328357
```

```
hist_plot_continuous_variable(df_new, 'highwaympg')
Executed at 2024.04.12 14:58:24 in 354ms
```



```
sns.scatterplot(df_new, x='highwaympg', y='price')

plt.show()
Executed at 2024.04.12 14:58:25 in 441ms
```



Ta có thể thấy là hiệu suất sử dụng nhiên liệu trong cao tốc càng thấp thì giá xe càng cao.

## 4. Kiểm tra các Feature dạng chữ

### 4.1. Xử lý và tạo ra cột Car Company từ Car Name

Đầu tiên ta có thể khẳng định 1 điều là giá xe không phụ thuộc vào tên của xe đó mà thường sẽ phụ thuộc vào tên thương hiệu hoặc tên công ty sản xuất ra chiếc xe đó.

Để chứng minh Car Name không ảnh hưởng đến giá xe ta sẽ làm 1 số thử nghiệm nhỏ như sau.

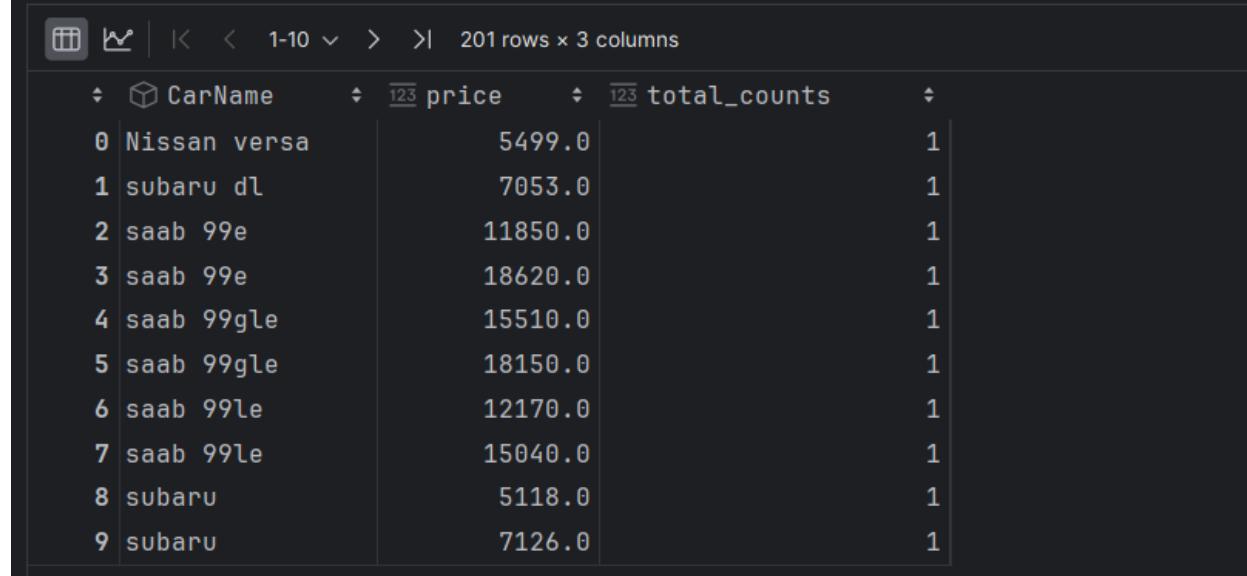
```
round(df_new['CarName'].nunique() / len(df_new) * 100, 2)
Executed at 2024.04.12 14:58:03 in 131ms
```

72.14

Ta có thể thấy nếu kết quả trên < 100% có nghĩa là có khá nhiều xe có cùng tên nhưng có giá khác nhau.

Tiếp theo ta tiến hành Group By dữ liệu theo CarName và price.

```
df_new.groupby(['CarName', 'price']).agg(
    total_counts = ('CarCompany', 'count')
).sort_values(by='total_counts', ascending=False).reset_index()
Executed at 2024.04.12 14:58:03 in 421ms
```



	CarName	price	total_counts
0	Nissan versa	5499.0	1
1	subaru dl	7053.0	1
2	saab 99e	11850.0	1
3	saab 99e	18620.0	1
4	saab 99gle	15510.0	1
5	saab 99gle	18150.0	1
6	saab 99le	12170.0	1
7	saab 99le	15040.0	1
8	subaru	5118.0	1
9	subaru	7126.0	1

Ta có thể nhận thấy các cặp giá trị CarName và price là duy nhất không có trùng lặp.

Tiếp theo ta sẽ đếm xem các CarName có trùng nhau nhiều hay là không?

```
df_car_name_groupby = df_new.groupby(['CarName']).agg(
    total_counts = ('price', 'count')
).sort_values(by='total_counts', ascending=False).reset_index()
```

df\_car\_name\_groupby

Executed at 2024.04.12 14:58:03 in 127ms

CarName	total_counts
peugeot 504	6
toyota corona	6
toyota corolla	6
subaru dl	4
toyota mark ii	3
mazda 626	3
mitsubishi mirage g4	3
mitsubishi g4	3
mitsubishi outlander	3
honda civic	3

Rất nhiều xe có tên trùng nhau. Ta sẽ thử lấy ra 1 tên xe bất kỳ và kiểm tra giá của nó.

```
df_new[df_new['CarName'] == 'toyota corolla'][['CarName', 'curbweight', 'enginesize', 'horsepower', 'citympg', 'highwaympg', 'price']]
```

Executed at 2024.04.12 15:22:25 in 304ms

CarName	curbweight	enginesize	horsepower	citympg	highwaympg	price
toyota corolla	2275	110	56	38	47	7788.0
toyota corolla	2122	98	70	28	34	8358.0
toyota corolla	2536	146	116	24	30	9639.0
toyota corolla	2714	146	116	24	30	11549.0
toyota corolla	2326	122	92	29	34	8948.0
toyota corolla	2414	122	92	27	32	10898.0

Bạn có thể thấy giá của 1 chiếc xe không được quyết định bởi tên của nó mà là do sự khác biệt giữa các thông số của nó mới quyết định đến giá của chiếc xe đó.

Vì thế ta sẽ loại bỏ Feature CarName khi xây dựng mô hình.

Tuy nhiên có khả năng tên công ty sản xuất xe sẽ có ảnh hưởng tới giá xe. Vì thế ta sẽ tách ra tên công ty sản xuất xe từ CarName.

Ta sẽ viết 1 hàm Split\_Car\_Name như sau:

```
def Split_Car_Name(car_name):
    return car_name.split(" ")[0]
df_new['CarCompany'] = df_new['CarName'].apply(Split_Car_Name)
```

Ta tạo ra 1 cột CarCompany bằng cách tách tên xe (CarName) theo dấu phân tách là dấu “ “ và lấy ra giá trị phân tách đầu tiên.

df\_new['CarCompany'].value\_counts()  
Executed at 2024.04.12 14:57:57 in 355ms

CarCompany	count
toyota	31
nissan	17
mazda	15
honda	13
mitsubishi	13
subaru	12
peugeot	11
volvo	11
volkswagen	9
dodge	9
buick	8
bmw	8
audi	7
plymouth	7
saab	6
isuzu	4
porsche	4
alfa-romero	3
chevrolet	3
jaguar	3
vw	2
maxda	2
renault	2
toyouta	1
vokswagen	1
Nissan	1
mercury	1
porcshce	1

Nhìn vào bảng trên ta có thể thấy có nhiều tên xe bị viết sai trong quá trình thu thập dữ liệu. Ví dụ như **volkswagen**, **vw** và **vokswagen** sẽ cần chuẩn hoá về 1 tên duy nhất là **volkswagen**

Ngoài ra còn có các giá trị cần chuyển đổi ở dưới đây nữa như sau.

- toyota, toyouta => **toyota**
- porsche, porcshce => **porsche**
- mazda, maxda => **mazda**
- nissan, Nissan => **nissan**

```
df_new['CarCompany'] = df_new['CarCompany'].replace('vw', 'volkswagen').replace('vokswagen', 'volkswagen')
df_new['CarCompany'] = df_new['CarCompany'].replace('toyouta', 'toyota')
df_new['CarCompany'] = df_new['CarCompany'].replace('porcshce', 'porsche')
df_new['CarCompany'] = df_new['CarCompany'].replace('maxda', 'mazda')
df_new['CarCompany'] = df_new['CarCompany'].replace('Nissan', 'nissan')
df_new['CarCompany'].value_counts()
```

Executed at 2024.04.12 14:57:57 in 388ms

CarCompany	count
toyota	32
nissan	18
mazda	17
mitsubishi	13
honda	13
volkswagen	12
subaru	12
peugeot	11
volvo	11
dodge	9
buick	8
bmw	8
audi	7
plymouth	7
saab	6
porsche	5
isuzu	4
jaguar	3
chevrolet	3
alfa-romero	3

Đến đây dữ liệu của Car Company đã chỉnh sửa xong. Ta sẽ đi phân tích xem Car Company có ảnh hưởng đến giá xe hay không?

```
df_new.groupby(['CarCompany']).agg(
    mean_price = ('price', 'mean'),
    median_price = ('price', np.median),
    total_count = ('price', 'count')
).sort_values(by='mean_price', ascending=False).reset_index()
Executed at 2024.04.12 14:58:25 in 357ms
```

22 rows × 4 columns

CarCompany	mean_price	median_price	total_count
0 jaguar	34600.000000	35550.0	3
1 buick	33647.000000	32892.0	8
2 porsche	31400.500000	33278.0	4
3 bmw	26118.750000	22835.0	8
4 volvo	18063.181818	18420.0	11
5 audi	17859.166667	17580.0	6
6 mercury	16503.000000	16503.0	1
7 alfa-romero	15498.333333	16500.0	3
8 peugeot	15489.090909	16630.0	11
9 saab	15223.333333	15275.0	6

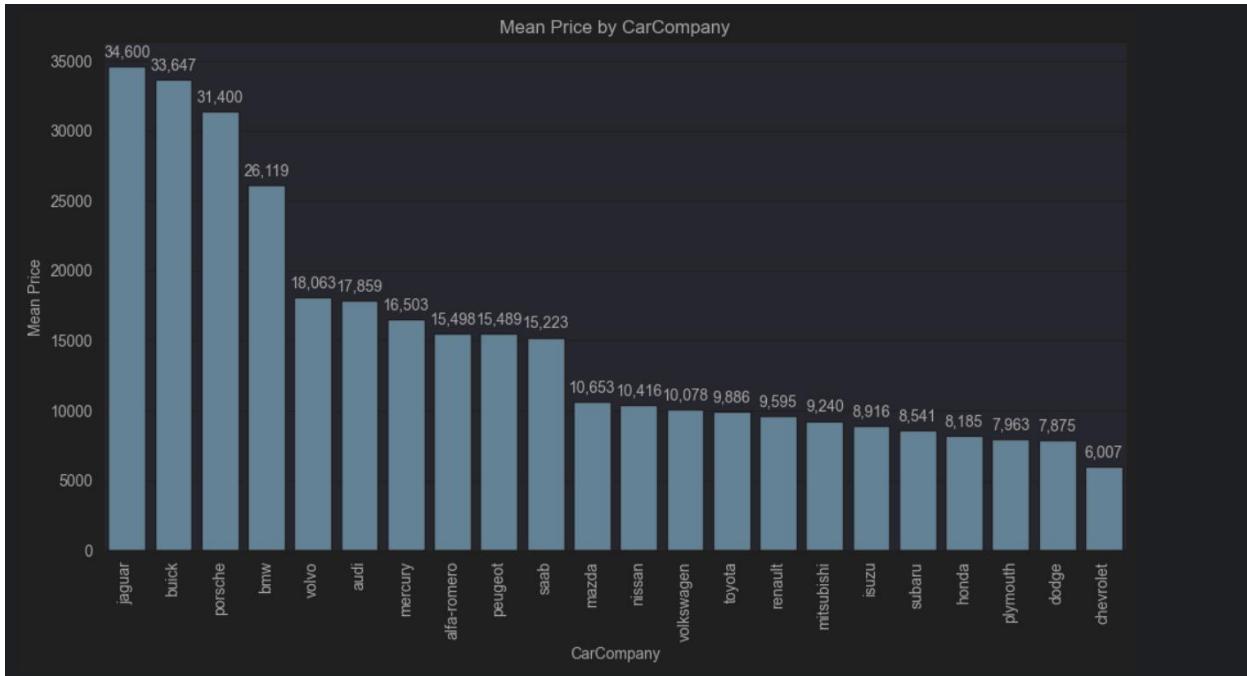
```
plt.figure(figsize=(12,6))

ax = sns.barplot(df_new.groupby(['CarCompany']).agg(
    mean_price = ('price', 'mean'),
    median_price = ('price', np.median),
    total_count = ('price', 'count')
).sort_values(by='mean_price', ascending=False).reset_index(), x='CarCompany', y='mean_price')

for p in ax.patches:
    ax.annotate(format(p.get_height(), ',.0f'),
                (p.get_x() + p.get_width() / 2, p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 9),
                textcoords = 'offset points')

plt.title('Mean Price by CarCompany')
plt.xlabel('CarCompany')
plt.ylabel('Mean Price')
plt.xticks(rotation=90)

plt.show()
Executed at 2024.04.12 14:58:25 in 440ms
```



```
plt.figure(figsize=(12,6))

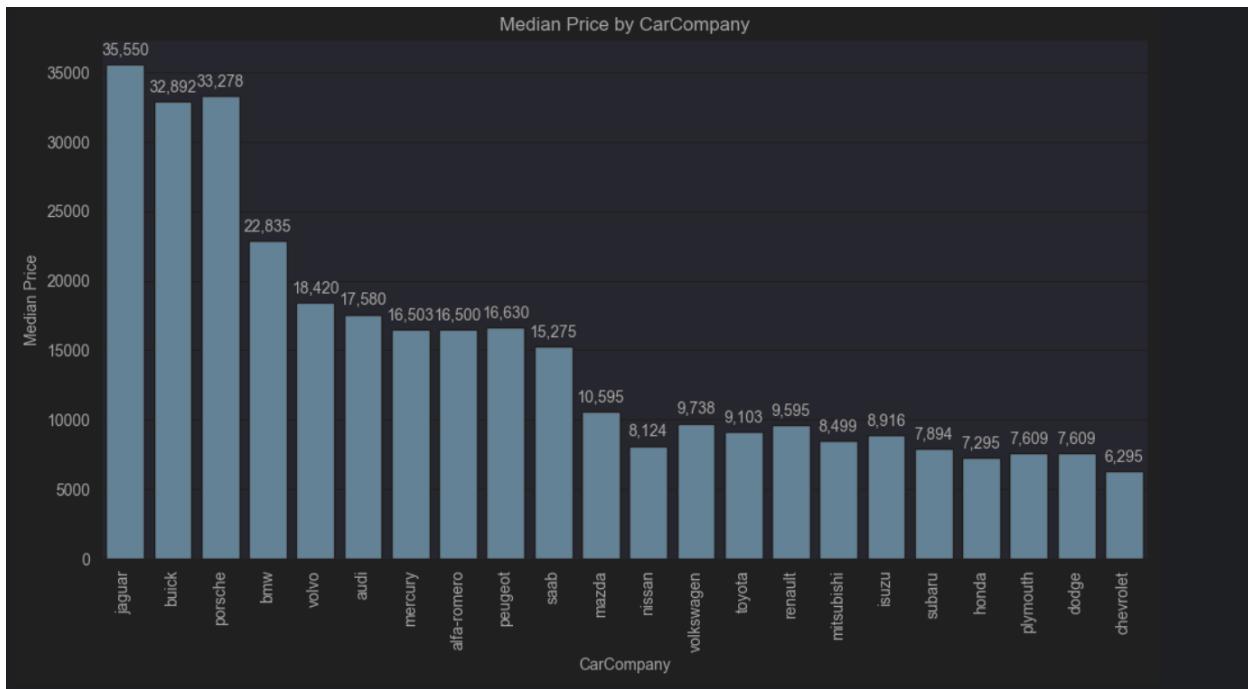
ax = sns.barplot(df_new.groupby(['CarCompany']).agg(
    mean_price = ('price', 'mean'),
    median_price = ('price', np.median),
    total_count = ('price', 'count')
).sort_values(by='mean_price', ascending=False).reset_index(), x='CarCompany', y='median_price')

for p in ax.patches:
    ax.annotate(format(p.get_height(), ',.0f'),
                (p.get_x() + p.get_width() / 2, p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 9),
                textcoords = 'offset points')

plt.title('Median Price by CarCompany')
plt.xlabel('CarCompany')
plt.ylabel('Median Price')
plt.xticks(rotation=90)

plt.show()
```

Executed at 2024.04.12 14:58:26 in 576ms



Như các bạn thấy thì các nhãn xe hạng sang như jaguar, buick, porsche, bmw luôn có giá cao hơn các xe khác.

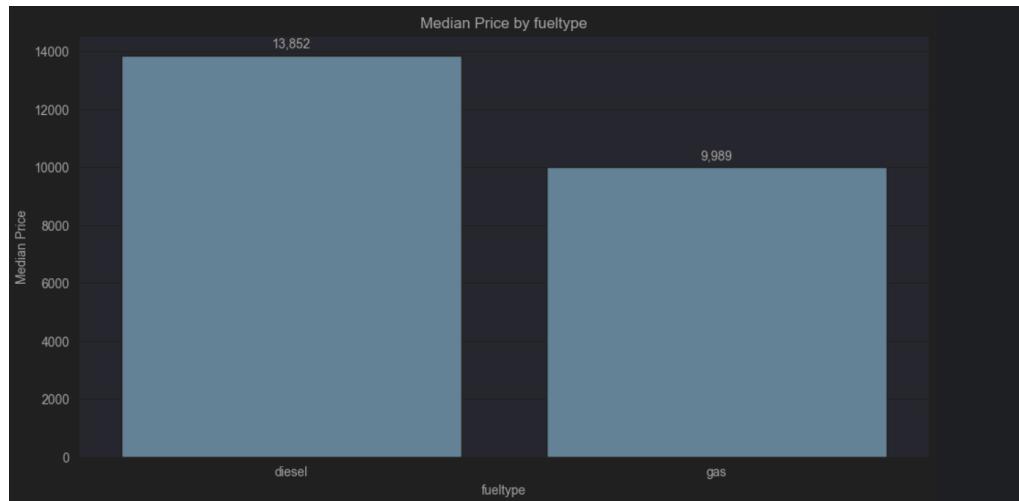
Điều này chứng tỏ thương hiệu của công ty sản xuất xe (CarCompany) có ảnh hưởng đến giá bán của xe.

## 4.2. Phân tích sự liên quan giữa loại nhiên liệu mà xe sử dụng (fueltype) và giá bán xe (price)

```
df_new.groupby(['fueltype']).agg(
    mean_price = ('price', 'mean'),
    median_price = ('price', np.median),
    total_count = ('price', 'count')
).sort_values(by='fueltype', ascending=True).reset_index()
```

# Chỉ có 2 loại là sử dụng dầu diesel và gas  
Executed at 2024.04.12 14:58:04 in 475ms

fueltype	mean_price	median_price	total_count
0 diesel	15838.15000	13852.5	20
1 gas	12916.40884	9989.0	181



Nhìn vào đây ta có thể thấy giá của xe chạy dầu diesel luôn có giá bán cao hơn so với xe chạy bằng xăng.

Trong thực tế thì các xe chạy dầu diesel luôn có giá cao hơn xăng vì Do các chi tiết của hệ thống nhiên liệu như kim phun, bơm cao áp có thiết kế rất tinh vi và đòi hỏi độ xác thực rất cao nên việc tu tạo phải được thực hiện bởi những máy móc tiên tiến, đắt tiền và tiến hành bởi thợ tay nghề cao nên giá thành sửa chữa cao => Đây là lý do chính khiến xe chạy dầu Diesel đắt hơn xe xăng

Vì vậy khi tiến hành xây dựng mô hình ta sẽ map dữ liệu với diesel = 1, gas = 0.

### 4.3. Phân tích sự liên quan giữa loại tăng áp sử dụng trong xe (aspiration) và giá bán của xe (price)

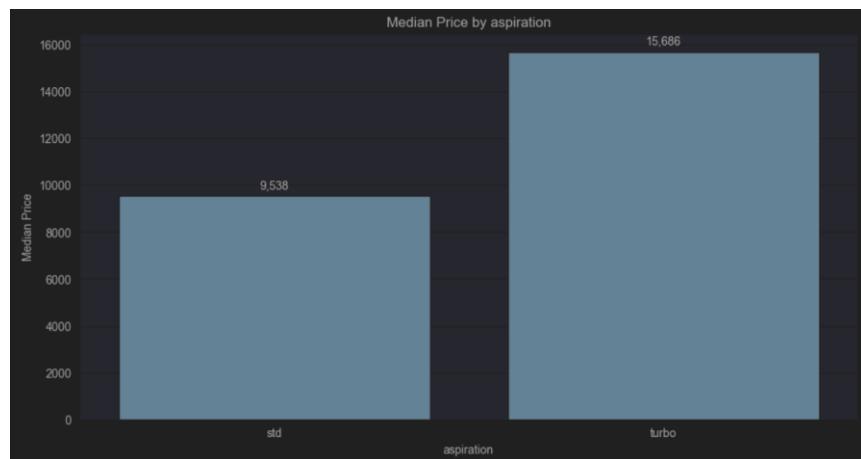
- **STD (naturally aspirated / động cơ thông thường):** Đây là loại động cơ không có hệ thống tăng áp (turbocharger hoặc supercharger). Thay vào đó, động cơ này hoạt động dựa trên nguyên lý hút khí tự nhiên thông qua quá trình hút không khí vào và đốt cháy nó trong xi lanh. Động cơ STD thường có kích thước nhỏ hơn và không có hiệu suất tăng cường so với các động cơ tăng áp.

- **Turbo (turbocharged / động cơ tăng áp):** Đây là loại động cơ có hệ thống tăng áp, trong đó khí nạp vào động cơ được nén trước khi đốt cháy. Hệ thống tăng áp giúp tăng hiệu suất của động cơ bằng cách cung cấp lượng khí nạp nhiều hơn vào xi lanh so với động cơ thông thường. Kết quả là, động cơ turbo thường có hiệu suất cao hơn và cung cấp một lượng lớn công suất tăng khi cần thiết.

Thường thì các xe ô tô được trang bị động cơ turbo sẽ có giá cao hơn so với các xe có động cơ thông thường (STD). Lý do chính là các động cơ turbo thường có hiệu suất cao hơn, cung cấp công suất tốt hơn và tiết kiệm nhiên liệu hơn so với các động cơ thông thường. => chỉ tiêu aspiration có ảnh hưởng đến giá xe

aspiration	mean_price	median_price	total_count
0 std	12542.181818	9538.0	165
1 turbo	16254.805556	15686.0	36

Giá bán trung bình của các xe sử dụng động cơ tăng áp (**turbo**) luôn cao hơn với động cơ thông thường (**std**).



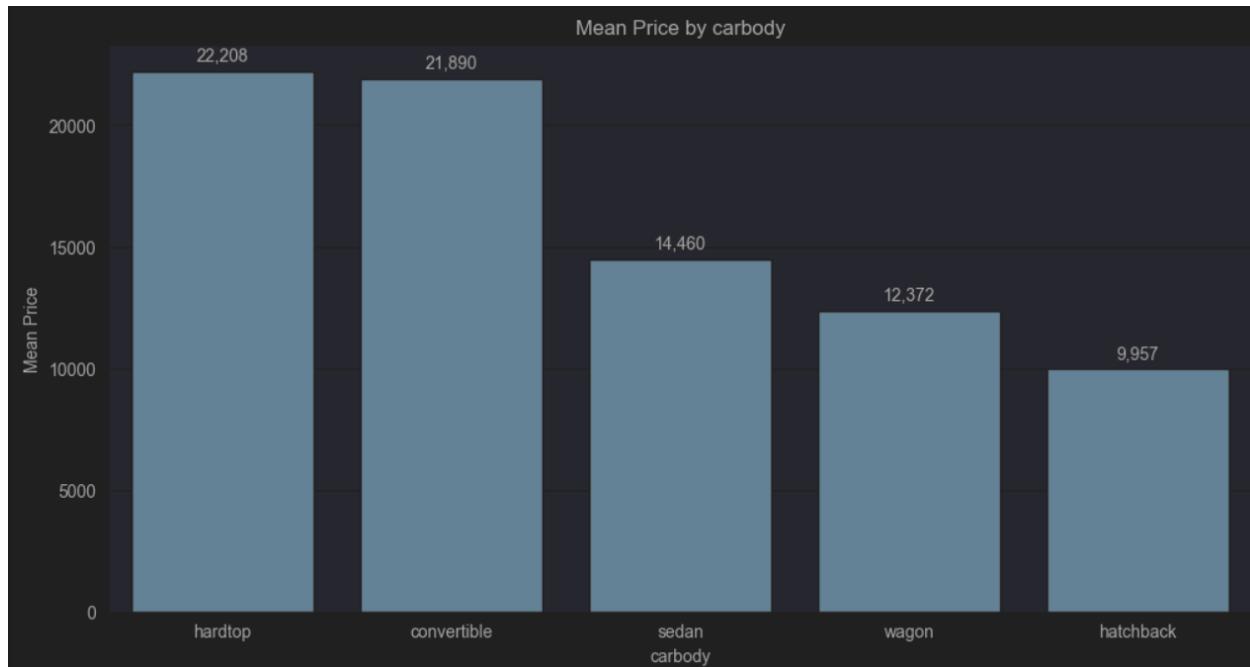
Vì vậy khi tiến hành xây dựng model ta sẽ map dữ liệu turbo với số 1, std với số 0.

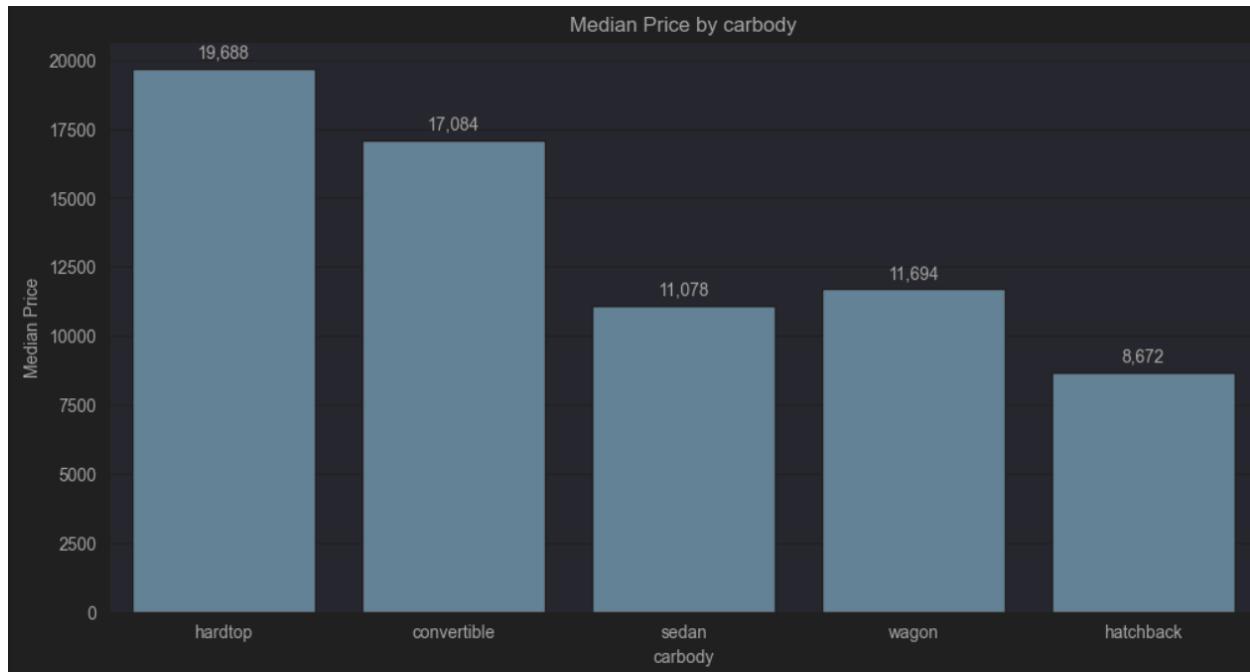
#### 4.4. Phân tích sự liên quan giữa thân xe (carbody) và giá bán của xe (price)

carbody	count
sedan	94
hatchback	68
wagon	25
hardtop	8
convertible	6

Trong dữ liệu của chúng ta có 5 loại thân xe là sedan, hatchback, wagon, hardtop, convertible.

carbody	mean_price	median_price	total_count
hardtop	22208.500000	19687.5	8
convertible	21890.500000	17084.5	6
sedan	14459.755319	11078.5	94
wagon	12371.960000	11694.0	25
hatchback	9957.441176	8672.0	68





Theo kết quả chúng ta sau khi sử dụng GroupBy theo thân xe thì ta có kết quả về mean và median như bảng trên.

Có thể thấy 1 điều đó là ta luôn có 2 loại thân xe là hardtop và convertible có giá cao nhất và thấp nhất sẽ là hatchback.

Tuy nhiên với 2 loại là sedan và wagon thì sedan có mean price cao hơn so với wagon nhưng median price của wagon lại lớn hơn của sedan.

Vì thế ta có thể thấy dữ liệu về thân xe có liên quan đến giá của xe. Tuy nhiên không thể sắp xếp nó theo 1 thứ tự cụ thể nào cả.

Vì thế khi xây dựng mô hình với feature carbody ta sẽ tiến hành sử dụng One Hot Encoding.

#### 4.5. Phân tích sự liên quan giữa vị trí bánh lái (drivewheel) và giá xe

drivewheel	count
fwd	118
rwd	75
4wd	8

Trong Data của chúng ta có 3 loại vị trí bánh lái là fwd, rwd và 4wd.

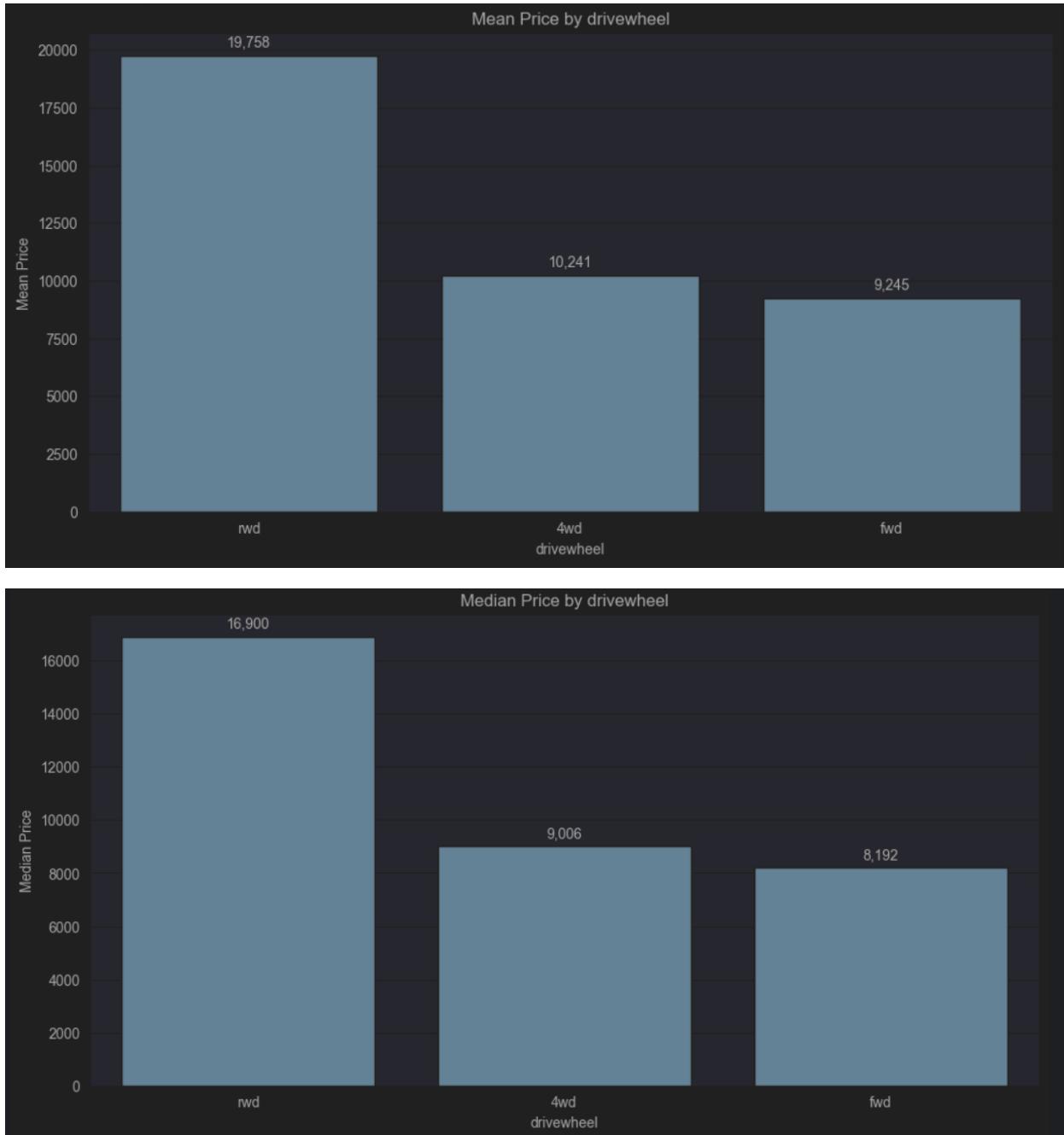
- **RWD (Rear-Wheel Drive):** Trong hệ thống này, bánh lái được động bởi bánh sau của xe. Điều này có nghĩa là động cơ truyền động chỉ đẩy lực đến bánh sau của xe. RWD thường được sử dụng trong các xe thể thao và xe hạng sang, vì nó cung cấp trải nghiệm lái xe truyền cảm giác và tốt cho việc vận hành với tốc độ cao.

- **FWD (Front-Wheel Drive):** Trong hệ thống này, bánh lái được động bởi bánh trước của xe. Điều này có nghĩa là động cơ truyền động tạo lực kéo từ phía trước của xe. FWD thường được sử dụng trong các xe hạng trung và xe du lịch, vì nó cung cấp sự tiết kiệm nhiên liệu và dễ điều khiển hơn trong điều kiện đường trơn trượt.

- **4WD (Four-Wheel Drive):** Trong hệ thống này, cả bánh trước và bánh sau đều được động. Thường thì xe 4WD có thể chuyển đổi giữa chế độ lái 4 bánh hoặc chỉ 2 bánh. 4WD thường được sử dụng trong các xe off-road và các xe đa dụng có khả năng vận hành tốt trên mọi loại địa hình.

Đọc qua các thông tin này thì có thể thấy giá cả của xe có liên quan tới bánh lái của xe được sử dụng là loại nào. Ta sẽ chứng minh nó bằng dữ liệu.

drivewheel	mean_price	median_price	total_count
rwd	19757.613333	16900.0	75
4wd	10241.000000	9005.5	8
fwd	9244.779661	8192.0	118



Bạn có thể thấy rõ sự phân hoá của mức giá theo bánh lái. Các xe sử dụng RWD có giá cao nhất, sau đó mới tới 4WD và FWD. Điều này cũng dễ hiểu vì RWD được sử dụng trên các mẫu xe thể thao hoặc hạng sang nên có giá trị rất lớn.

Vì thế ta sẽ tiến hành map lại dữ liệu cột drivewheel với giá trị lần lượt như sau:

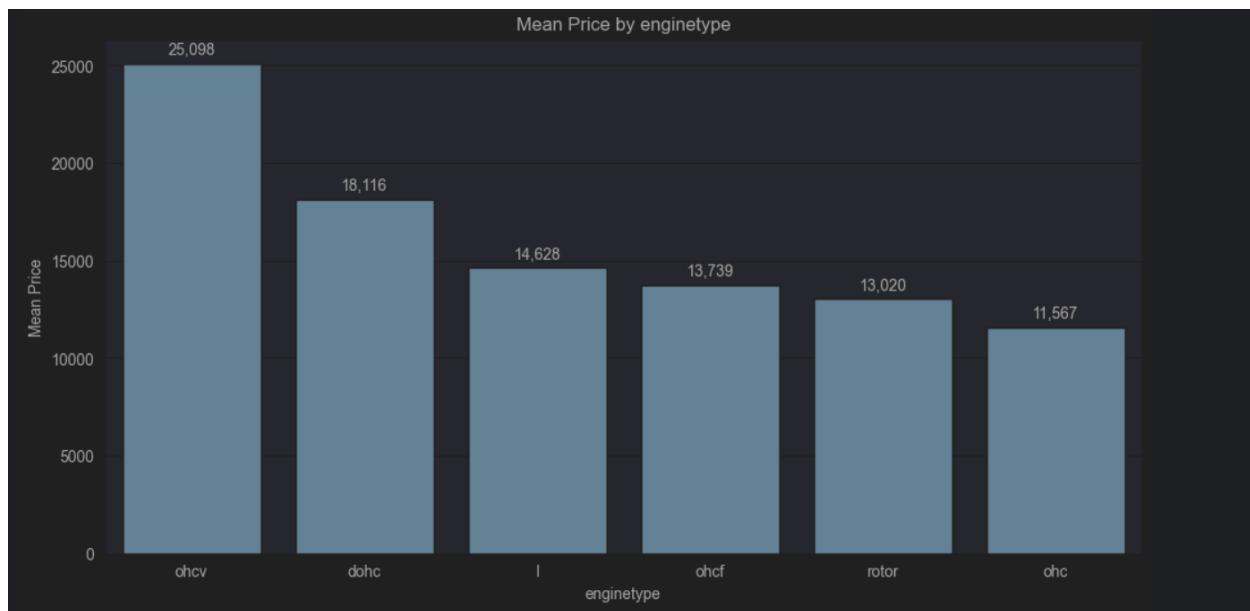
- rwd: 2, 4wd: 1 và fwd: 0

#### 4.6. Phân tích sự liên quan giữa loại động cơ xe (enginetype) và giá xe

enginetype	mean_price	median_price	total_count
ohcv	25098.384615	19699.0	13
dohc	18116.416667	16249.0	12
l	14627.583333	16105.0	12
ohcf	13738.600000	9233.0	15
rotor	13020.000000	12745.0	4
ohc	11567.358621	9095.0	145

Trong dữ liệu của chúng ta có tất cả 6 loại động cơ xe lần lượt là ohcv, dohc, l, ohcf, rotor, ohc.

- **DOHC**: Double Overhead Camshaft - Hai trực cam trên đầu.
- **OHCV**: OverHead Camshaft Valve - Van trên đầu được điều khiển bởi trực cam.
- **OHC**: OverHead Camshaft - Trực cam trên đầu.
- **L**: Là một động cơ trực đơn được sử dụng bởi Mazda, thường được sử dụng cho động cơ rotor của họ.
- **Rotor**: Đây là loại động cơ đặc biệt, thường được sử dụng trong các xe máy bay và trong các mẫu xe đặc biệt như Mazda RX-7.
- **OHCF**: OverHead Camshaft Valve, Front-wheel drive - Điều khiển van trên đầu, dẫn động cầu trước.





Thực tế bạn có thể nhận ra là các xe sử dụng động cơ OHCV, DOHC, L thường có giá bán cao hơn các loại xe sử dụng các loại động cơ còn lại.

Vì thế ta có thể thấy việc sử dụng động cơ nào có liên quan tới giá xe. Tuy nhiên do không thể sắp xếp các giá trị trong feature này theo 1 thứ tự nào cả nên ta sẽ tiến hành One Hot Encoding với feature này.

## 4.7. Phân tích sự liên quan giữa số lượng xỉ lanh trong động cơ và giá bán của xe

```
def words_to_numbers(text):
    word_to_num = {
        'zero': 0,
        'one': 1,
        'two': 2,
        'three': 3,
        'four': 4,
        'five': 5,
        'six': 6,
        'seven': 7,
        'eight': 8,
        'nine': 9,
        'ten': 10,
        'eleven': 11,
        'twelve': 12,
        'thirteen': 13,
        'fourteen': 14,
        'fifteen': 15,
        'sixteen': 16,
        'seventeen': 17,
        'eighteen': 18,
        'nineteen': 19,
        'twenty': 20,
        'thirty': 30,
        'forty': 40,
        'fifty': 50,
        'sixty': 60,
        'seventy': 70,
        'eighty': 80,
        'ninety': 90,
        'hundred': 100,
        'thousand': 1000,
        'million': 1000000,
        'billion': 1000000000,
    }

    # Split the text into words
    words = text.lower().split()

    result = 0
    temp_result = 0
    for word in words:
        if word == 'and':
            continue
        if word in word_to_num:
            if word == 'hundred':
                temp_result *= word_to_num[word]
            else:
                temp_result += word_to_num[word]
        elif '-' in word:
```

```

parts = word.split('-')
temp = 0
for part in parts:
    temp += word_to_num[part]
temp_result += temp
else:
    result += temp_result * word_to_num[word]
    temp_result = 0

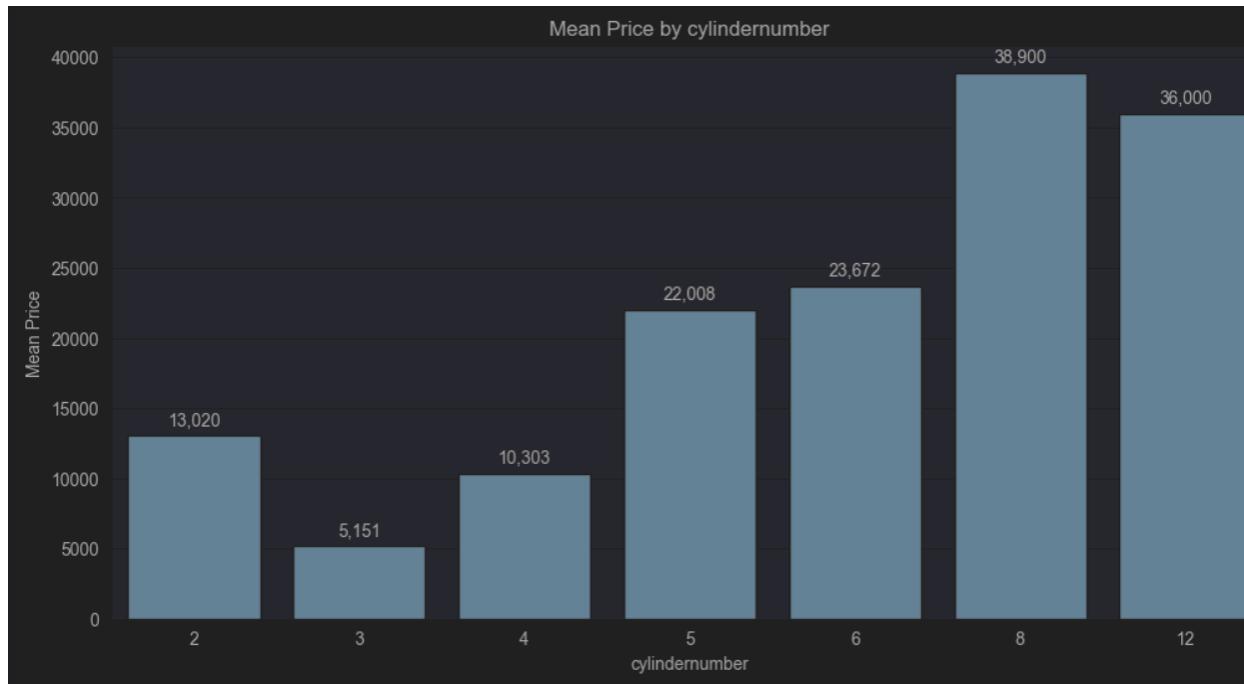
return result + temp_result

```

Ta sẽ viết 1 Function để chuyển đổi giá trị chữ thành số

```
df_new['cylindernumber'] = df_new['cylindernumber'].apply(words_to_numbers)
```

	cylindernumber	mean_price	median_price	total_count
0	8	38900.000000	38008.0	4
1	12	36000.000000	36000.0	1
2	6	23671.833333	21037.5	24
3	5	22007.600000	21397.5	10
4	2	13020.000000	12745.0	4
5	4	10303.197452	8949.0	157
6	3	5151.000000	5151.0	1



Nhìn vào đây ta có thể thấy các xe có nhiều xi lanh trong động cơ thường sẽ có giá cao hơn các loại xe có ít xi lanh trong động cơ.

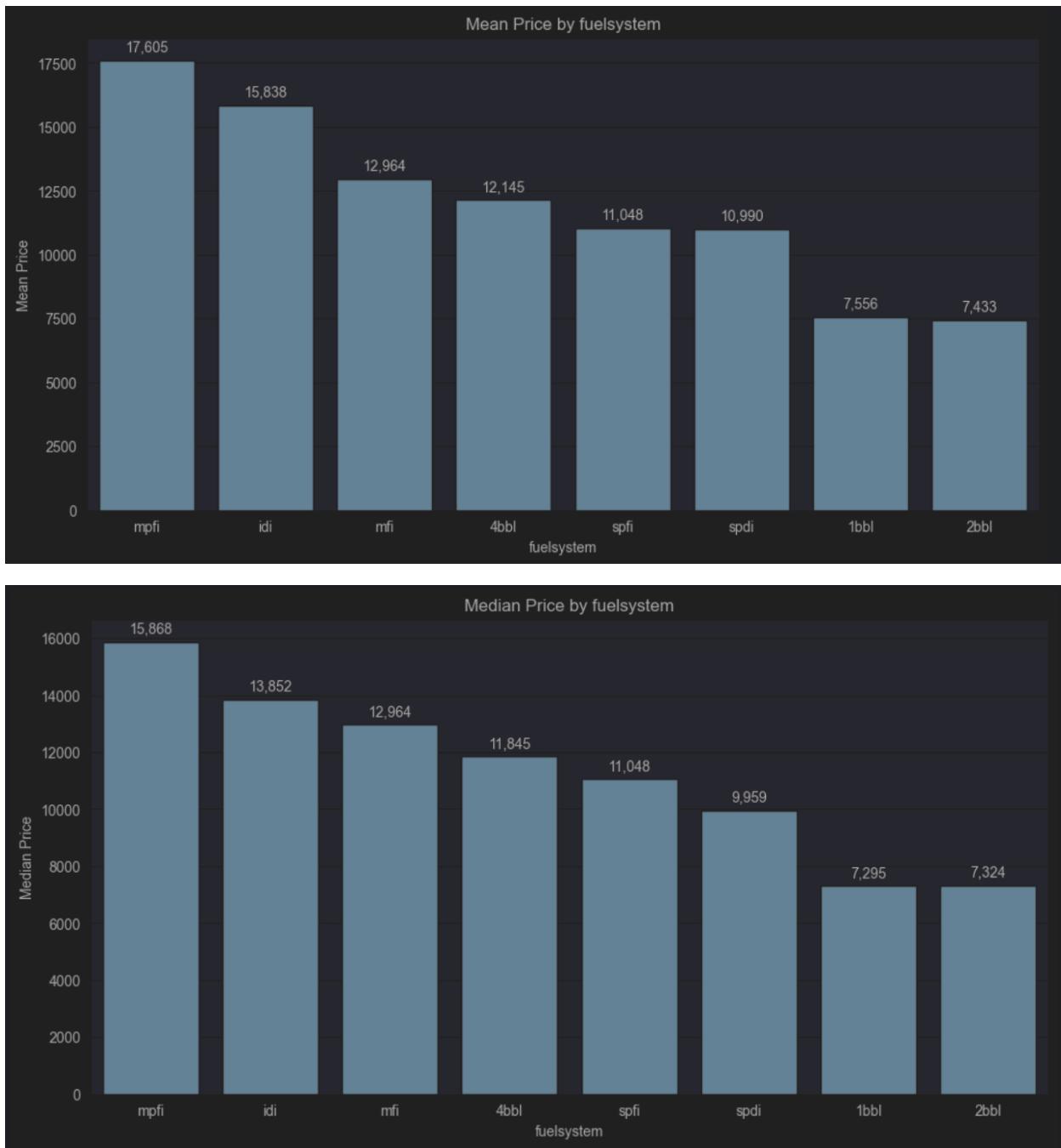
Vì thế đây cũng là 1 yếu tố có ảnh hưởng tới giá xe.

#### 4.7. Phân tích sự liên quan giữa hệ thống truyền dẫn nhiên liệu (fuelsystem) và giá xe (price)

Trong Data của chúng ta có tất cả 7 hệ thống cung cấp nhiên liệu là:

- **MPFI**: Multi-Point Fuel Injection - Hệ thống phun nhiên liệu đa điểm. Hệ thống này cung cấp nhiên liệu trực tiếp vào mỗi xi-lanh của động cơ, cung cấp điều khiển chính xác và hiệu quả cao hơn so với các hệ thống truyền thống.
- **2BBL**: Two-Barrel Carburetor - Carburetor hai ống hút. Carburetor này có hai ống hút nhiên liệu, được sử dụng để phun nhiên liệu vào buồng đốt của động cơ.
- **MFI**: Multi-Port Fuel Injection - Phun nhiên liệu từng điểm. Tương tự như MPFI, MFI là hệ thống phun nhiên liệu đa điểm.
- **1BBL**: One-Barrel Carburetor - Carburetor một ống hút. Carburetor này có một ống hút nhiên liệu, được sử dụng để phun nhiên liệu vào buồng đốt của động cơ.
- **SPFI**: Single-Point Fuel Injection - Hệ thống phun nhiên liệu đơn điểm. Hệ thống này cung cấp nhiên liệu vào một điểm duy nhất, thường là trên bộ phân phối hoặc ống hút của động cơ.
- **4BBL**: Four-Barrel Carburetor - Carburetor bốn ống hút. Carburetor này có bốn ống hút nhiên liệu, được sử dụng để phun nhiên liệu vào buồng đốt của động cơ.
- **IDI**: Indirect Injection - Phun nhiên liệu gián tiếp. Hệ thống này phun nhiên liệu vào một không gian phụ trợ trước khi nó được đưa vào buồng đốt của động cơ.
- **SPDI**: Single-Point Direct Injection - Phun nhiên liệu đơn điểm trực tiếp. Hệ thống này phun nhiên liệu trực tiếp vào buồng đốt của động cơ.

fuelsystem	mean_price	median_price	total_count
mpfi	17605.141304	15867.5	92
idi	15838.150000	13852.5	20
mfi	12964.000000	12964.0	1
4bbl	12145.000000	11845.0	3
spfi	11048.000000	11048.0	1
spdi	10990.444444	9959.0	9
1bbl	7555.545455	7295.0	11
2bbl	7433.203125	7324.0	64



Nhìn vào các kết quả tính toán và biểu đồ ta có thể rút ra được 1 thứ tự phụ thuộc giữa giá xe và hệ thống truyền dẫn nhiên liệu được sử dụng.

- **MPFI** (Multi-Point Fuel Injection): Hệ thống phun nhiên liệu đa điểm cung cấp nhiên liệu một cách chính xác cho mỗi xi lanh. Đây là một hệ thống phun nhiên liệu tiên tiến,

giúp tăng hiệu quả đốt cháy và giảm lượng khí thải, thường được tìm thấy trên các xe hiện đại.

- **SPDI** (Single-Point Direct Injection): Dù chỉ phun nhiên liệu vào một điểm trực tiếp trong xi lanh, nhưng do là phun trực tiếp nên cũng khá hiệu quả và tiên tiến, thường xuất hiện trên các xe có giá trị kinh tế cao.
- **MFI** (Multi-Port Fuel Injection): Tương tự như MPFI nhưng thường được xem là một phiên bản ít phổ biến hơn. Nó vẫn mang lại hiệu suất đốt cháy tốt và hiệu quả nhiên liệu.
- **SPFI** (Single-Point Fuel Injection): Phun nhiên liệu đơn điểm không chính xác bằng các hệ thống phun đa điểm nhưng vẫn tốt hơn so với các hệ thống carburetor.
- **IDI** (Indirect Injection): Phun nhiên liệu gián tiếp không hiệu quả bằng phun trực tiếp nhưng vẫn được sử dụng rộng rãi trên các động cơ diesel.
- **4BBL** (Four-Barrel Carburetor): Carburetor bốn ống hút thường được tìm thấy trên các xe cổ hoặc xe hiệu suất cao từ thời kỳ trước khi hệ thống phun nhiên liệu trở nên phổ biến.
- **2BBL** (Two-Barrel Carburetor): Carburetor hai ống hút là một lựa chọn kinh tế hơn so với carburetor bốn ống hút, vẫn cung cấp hiệu suất đủ dùng nhưng không cao.
- **1BBL** (One-Barrel Carburetor): Carburetor một ống hút là loại đơn giản nhất và thường chỉ được sử dụng trên các loại xe cũ hoặc xe có giá rẻ.

Dựa theo các kết quả phân tích ta sẽ tiến hành map lại dữ liệu cột fuel system như sau:

**mpfi: 7, spdi: 6, mfi: 5, spfi: 4, idi: 3, 4bbl: 2, 2bbl: 1, 1bbl: 0**

## 5. Kết luận

Sau khi tiến hành EDA dữ liệu thì ta rút ra được 12 Feature có ảnh hưởng tới giá xe như sau.

- Độ nặng của xe (curbweight)
- Kích thước động cơ của xe (enginesize)
- Mã lực của xe (horsepower)
- Hiệu suất nhiên liệu của 1 xe hơi đi trong thành phố hoặc đô thị (citympg)
- Hiệu suất nhiên liệu của 1 xe hơi đi trong cao tốc (highwaympg)
- Tên công ty sản xuất xe (CarCompany)
- Loại nhiên liệu mà xe sử dụng (fueltype)
- Loại tăng áp sử dụng trong xe (aspiration)
- Thân xe được sử dụng (carbody)
- Vị trí bánh lái (drivewheel)
- Động cơ xe (enginetype)
- Hệ thống truyền dẫn nhiên liệu (fuelsystem)

```
feature_choose = ['fueltype', 'aspiration', 'carbody', 'drivewheel', 'curbweight', 'enginetype',
  'cylindernumber', 'enginesize', 'fuelsystem', 'horsepower', 'citympg', 'highwaympg', 'CarCompany', 'price']
df_final = df_new[feature_choose]

df_final
```

	fueltype	aspiration	carbody	drivewheel	curbweight	enginetype	cylindernumber	enginesize	fuelsystem	horsepower
0	0	0	convertible	2	2548	dohc	4	130	7	
1	0	0	convertible	2	2548	dohc	4	130	7	
2	0	0	hatchback	2	2823	ohcv	6	152	7	
3	0	0	sedan	0	2337	ohc	4	109	7	
4	0	0	sedan	1	2824	ohc	5	136	7	
5	0	0	sedan	0	2507	ohc	5	136	7	
6	0	0	sedan	0	2844	ohc	5	136	7	
7	0	0	wagon	0	2954	ohc	5	136	7	
8	0	1	sedan	0	3086	ohc	5	131	7	
10	0	0	sedan	2	2395	ohc	4	108	7	

## D. Xây dựng mô hình dự báo giá xe

### 1. Tiến hành Mapping và Encode dữ liệu

Tiến hành map lại dữ liệu cột fueltype

```
df_final['fueltype'] = df_final['fueltype'].map(
{
    'diesel': 1,
    'gas': 0
})
```

Tiến hành map lại dữ liệu cột aspiration

```
df_final['aspiration'] = df_final['aspiration'].map(
{
    'turbo': 1,
    'std': 0
})
```

Tiến hành map lại dữ liệu cột drivewheel

```
df_final['drivewheel'] = df_final['drivewheel'].map(
{
    'rwd': 2,
    '4wd': 1,
    'fwd': 0
})
```

Tiến hành map lại dữ liệu cột fuelsystem

```
df_final['fuelsystem'] = df_final['fuelsystem'].map(
{
    'mpfi': 7,
    'spdi': 6,
    'mfi': 5,
    'spfi': 4,
    'idi': 3,
    '4bbl': 2,
    '2bbl': 1,
    '1bbl': 0
})
```

Sử dụng One Hot Encoding với các cột carbbody, enginetype, cylindernumber và CarCompany.

```
df_final = pd.get_dummies(df_final, columns=['carbody', 'enginetype', 'cylindernumber', 'CarCompany'],
drop_first=False, dtype='int')
```

```
df_final.info()
```

Executed at 2024.04.12 16:02:06 in 177ms

```
<class 'pandas.core.frame.DataFrame'>
Index: 201 entries, 0 to 204
Data columns (total 50 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   fueltype          201 non-null    int64  
 1   aspiration        201 non-null    int64  
 2   drivewheel        201 non-null    int64  
 3   curbweight        201 non-null    int64  
 4   enginesize        201 non-null    int64  
 5   fuelsystem        201 non-null    int64  
 6   horsepower         201 non-null    int64  
 7   citympg           201 non-null    int64  
 8   highwaympg        201 non-null    int64  
 9   price              201 non-null    float64 
 10  carbody_convertible 201 non-null    int32  
 11  carbody_hardtop   201 non-null    int32  
 12  carbody_hatchback 201 non-null    int32  
 13  carbody_sedan     201 non-null    int32  
 14  carbody_wagon     201 non-null    int32  
 15  enginetype_dohc   201 non-null    int32  
 16  enginetype_l       201 non-null    int32  
 17  enginetype_ohc    201 non-null    int32  
 18  enginetype_ohcf   201 non-null    int32  
 19  enginetype_ohcv   201 non-null    int32  
 20  enginetype_rotor  201 non-null    int32  
 21  cylindernumber_2  201 non-null    int32  
 22  cylindernumber_3  201 non-null    int32
```

```

23 cylindernumber_4          201 non-null    int32
24 cylindernumber_5          201 non-null    int32
25 cylindernumber_6          201 non-null    int32
26 cylindernumber_8          201 non-null    int32
27 cylindernumber_12         201 non-null    int32
28 CarCompany_alfa-romero   201 non-null    int32
29 CarCompany_audi           201 non-null    int32
30 CarCompany_bmw            201 non-null    int32
31 CarCompany_buick          201 non-null    int32
32 CarCompany_chevrolet     201 non-null    int32
33 CarCompany_dodge          201 non-null    int32
34 CarCompany_honda          201 non-null    int32
35 CarCompany_isuzu          201 non-null    int32
36 CarCompany_jaguar         201 non-null    int32
37 CarCompany_mazda          201 non-null    int32
38 CarCompany_mercury        201 non-null    int32
39 CarCompany_mitsubishi    201 non-null    int32
40 CarCompany_nissan         201 non-null    int32
41 CarCompany_peugeot        201 non-null    int32
42 CarCompany_plymouth       201 non-null    int32
43 CarCompany_porsche        201 non-null    int32
44 CarCompany_renault        201 non-null    int32
45 CarCompany_saab           201 non-null    int32
46 CarCompany_subaru         201 non-null    int32
47 CarCompany_toyota         201 non-null    int32
48 CarCompany_volkswagen    201 non-null    int32
49 CarCompany_volvo          201 non-null    int32
dtypes: float64(1), int32(40), int64(9)

```

Sau khi tiến hành Encode dữ liệu chúng ta sẽ có tổng cộng 49 cột như 2 hình trên.

## 2. Tiến hành chia tập X, y và tập train, test

```
X = df_final.drop(columns=target_var)

y = df_final[target_var]
Executed at 2024.04.12 16:02:06 in 172ms

from sklearn.model_selection import train_test_split
Executed at 2024.04.12 16:02:06 in 394ms

# Chia tập train và test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
Executed at 2024.04.12 16:02:07 in 151ms

R2_train = list()
R2_test = list()
Time_Running = list()
Executed at 2024.04.12 16:02:07 in 162ms
```

Ta sẽ tạo luôn 3 list trống là R2\_train, R2\_test, Time\_Running để ghi lại các kết quả train, test và thời gian chạy các mô hình để có thể dễ dàng chọn mô hình hơn.

### 3. Xây dựng mô hình dự đoán sử dụng Linear Regression

```

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
Executed at 2024.04.12 16:02:07 in 102ms

start_time = time.time()

model_linear = LinearRegression()

model_linear.fit(X_train, y_train)

y_pred = model_linear.predict(X_test)
Executed at 2024.04.12 16:02:07 in 200ms

print(f'Mean Square Error: {mean_squared_error(y_test, y_pred)}')
Executed at 2024.04.12 16:02:07 in 116ms

Mean Square Error: 10555930.07242978

print(f'Root Mean Square Error: {np.sqrt(mean_squared_error(y_test, y_pred))}')
Executed at 2024.04.12 16:02:07 in 159ms

Root Mean Square Error: 3248.989084689233

# R2 Score cho kết quả là 0.888 => khá tốt
print(f'R2 Score: {r2_score(y_test, y_pred)}')

end_time = time.time()

R2_train.append(r2_score(y_train, model_linear.predict(X_train)))
R2_test.append(r2_score(y_test, y_pred))
Time_Running.append(round(end_time - start_time, 2))
Executed at 2024.04.12 16:02:07 in 125ms

R2 Score: 0.8886407277906978

```

Sử dụng Linear Regression cho kết quả R2 Score là 0.888 là khá tốt. Tuy nhiên ta sẽ tiếp tục thử các mô hình khác để xem kết quả trả về là như thế nào.

## 4. Xây dựng mô hình dự đoán sử dụng Linear Regression và Adaboosting

```
from sklearn.ensemble import AdaBoostRegressor
Executed at 2024.04.12 16:02:08 in 210ms

start_time = time.time()

ada_reg_model = AdaBoostRegressor(estimator=model_linear, n_estimators=500, random_state=42, learning_rate=0.1)

ada_reg_model.fit(X_train, y_train)

y_pred = ada_reg_model.predict(X_test)
Executed at 2024.04.12 16:02:08 in 547ms

print(f'Mean Square Error: {mean_squared_error(y_test, y_pred)}')
Executed at 2024.04.12 16:02:08 in 54ms

Mean Square Error: 9653958.12945051

print(f'Root Mean Square Error: {np.sqrt(mean_squared_error(y_test, y_pred))}')
Executed at 2024.04.12 16:02:08 in 72ms

Root Mean Square Error: 3107.081931563844

# Kết quả thu được cao hơn so với ban đầu 1 chút (0.898)
print(f'R2 Score: {r2_score(y_test, y_pred)}')

end_time = time.time()

R2_train.append(r2_score(y_train, ada_reg_model.predict(X_train)))
R2_test.append(r2_score(y_test, y_pred))
Time_Running.append(round(end_time - start_time, 2))
Executed at 2024.04.12 16:02:08 in 225ms

R2 Score: 0.8981560370466506
```

Kết quả có cao hơn 1 chút đạt 0.898 điểm (Tốt)

## 5. Xây dựng mô hình dự đoán sử dụng Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor
Executed at 2024.04.12 16:52:11 in 252ms

start_time = time.time()

model_random_forest_regressor = RandomForestRegressor(n_estimators=100, random_state=42)

model_random_forest_regressor.fit(X_train, y_train)

y_pred = model_random_forest_regressor.predict(X_test)
Executed at 2024.04.12 16:52:11 in 318ms

print(f'Mean Square Error: {mean_squared_error(y_test, y_pred)}')
Executed at 2024.04.12 16:52:11 in 193ms

Mean Square Error: 8325855.8577951845

print(f'Root Mean Square Error: {np.sqrt(mean_squared_error(y_test, y_pred))}')
Executed at 2024.04.12 16:52:12 in 264ms

Root Mean Square Error: 2885.455918532665

# Kết quả cao hơn so với mô hình ban đầu.
print(f'R2 Score: {r2_score(y_test, y_pred)}')
Executed at 2024.04.12 16:52:12 in 276ms

R2 Score: 0.9121667875325161

end_time = time.time()

R2_train.append(r2_score(y_train, model_random_forest_regressor.predict(X_train)))
R2_test.append(r2_score(y_test, y_pred))
Time_Running.append(round(end_time - start_time, 2))
Executed at 2024.04.12 16:52:12 in 184ms
```

Có thể thấy kết quả của mô hình đạt 0.912 (Là kết quả cao nhất đến lúc này)

Ta sẽ thử Tuning thêm 1 chút xem có đạt kết quả tốt hơn không.

```

end_time = time.time()

R2_train.append(r2_score(y_train, model_random_forest_regressor.predict(X_train)))
R2_test.append(r2_score(y_test, y_pred))
Time_Running.append(round(end_time - start_time, 2))
Executed at 2024.04.12 16:52:12 in 184ms

# Ta sẽ thử tuning bằng GridSearchCV
from sklearn.model_selection import GridSearchCV
Executed at 2024.04.12 16:52:12 in 142ms

start_time = time.time()

rf_model = RandomForestRegressor(random_state=42)

param_grid = {
    'max_depth': np.arange(3, 22, 2),  # Range of max_depth values from 3 to 21 with a step of 2
    'n_estimators': range(1, 22, 2),    # Range of n_estimators values from 1 to 21 with a step of 2
    'criterion': ['squared_error', 'absolute_error']          # Criterion options for RandomForestRegressor
}

grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv=5, n_jobs=-1)

grid_search.fit(X_train, y_train)
Executed at 2024.04.12 16:52:17 in 5s 247ms

    ▶      GridSearchCV      ⓘ ⓘ
    ▶ estimator: RandomForestRegressor
        ▶ RandomForestRegressor ⓘ
            |
```

print("Best Parameters:", grid\_search.best\_params\_)

Executed at 2024.04.12 16:52:17 in 31ms

Best Parameters: {'criterion': 'squared\_error', 'max\_depth': 7, 'n\_estimators': 21}

```
rf_model_tuning = grid_search.best_estimator_
Executed at 2024.04.12 16:52:17 in 58ms

y_pred = rf_model_tuning.predict(X_test)
Executed at 2024.04.12 16:52:17 in 244ms

print(f'Mean Square Error: {mean_squared_error(y_test, y_pred)}')
Executed at 2024.04.12 16:52:18 in 286ms

Mean Square Error: 8166588.598924138

print(f'Root Mean Square Error: {np.sqrt(mean_squared_error(y_test, y_pred))}')
Executed at 2024.04.12 16:52:18 in 411ms

Root Mean Square Error: 2857.7243742047863

# Đây là kết quả tốt nhất cho mô hình đến lúc này
print(f'R2 Score: {r2_score(y_test, y_pred)}')

end_time = time.time()

R2_train.append(r2_score(y_train, rf_model_tuning.predict(X_train)))
R2_test.append(r2_score(y_test, y_pred))
Time_Running.append(round(end_time - start_time, 2))
Executed at 2024.04.12 16:52:18 in 446ms

R2 Score: 0.9138469697535951
```

Kết quả có tốt hơn 1 chút (0.913)

## 6. Xây dựng mô hình dự đoán sử dụng SVR

Để sử dụng mô hình SVR thì ta phải scale dữ liệu.

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
Executed at 2024.04.12 16:54:34 in 135ms

scaler = StandardScaler()

X_train_scaler = scaler.fit_transform(X_train)
X_test_scaler = scaler.fit_transform(X_test)
Executed at 2024.04.12 16:54:34 in 115ms

model_svr = SVR()

model_svr.fit(X_train_scaler, y_train)

y_pred = model_svr.predict(X_test_scaler)
Executed at 2024.04.12 16:54:34 in 143ms
```

```
start_time = time.time()

# Khởi tạo mô hình SVR
svr_model = SVR()

# Định nghĩa các siêu tham số cần tìm kiếm
param_grid = {
    'svr__C': [0.001, 0.01, 0.1, 1, 10, 20, 30],
    'svr__gamma': ['scale', 'auto'],
    'svr__epsilon': [0.01, 0.1, 0.2, 0.5, 1],
    'svr__kernel': ['linear', 'rbf', 'poly', 'sigmoid']
}

grid_search.fit(X_train_scaler, y_train)

# In ra các siêu tham số tốt nhất
print("Best Parameters:", grid_search.best_params_)
Executed at 2024.04.12 16:54:38 in 3s 85ms

Best Parameters: {'criterion': 'squared_error', 'max_depth': 7, 'n_estimators': 21}

svr_model_tuning = grid_search.best_estimator_

y_pred = svr_model_tuning.predict(X_test_scaler)
Executed at 2024.04.12 16:54:38 in 160ms

print(f'Mean Square Error: {mean_squared_error(y_test, y_pred)}')
Executed at 2024.04.12 16:54:38 in 164ms

Mean Square Error: 15137251.309764024

print(f'Root Mean Square Error: {np.sqrt(mean_squared_error(y_test, y_pred))}')
Executed at 2024.04.12 16:54:38 in 170ms

Root Mean Square Error: 3890.6620657368876
```

```
# Sau khi tuning thi kết quả mô hình đã tốt hơn nhưng vẫn không bằng Random Forest
print(f'R2 Score: {r2_score(y_test, y_pred)}')

end_time = time.time()

R2_train.append(r2_score(y_train, svr_model_tuning.predict(X_train_scaler)))
R2_test.append(r2_score(y_test, y_pred))
Time_Running.append(round(end_time - start_time, 2))
Executed at 2024.04.12 16:54:38 in 241ms

R2 Score: 0.8403103016467199
```

Kết quả thu được không tốt lắm và còn thấp hơn các mô hình trước đó (Chỉ đạt 0.840)

## 7. Kết luận

Model	R Squared(Train)	R Squared(Test)	Time Running
0 RandomForestRegressor Tuning	0.975377	0.913847	3.11
1 RandomForestRegressor	0.984763	0.912167	0.85
2 Linear Regression + Adaboosting	0.963682	0.898156	0.58
3 LinearRegression	0.962885	0.888641	0.28
4 SVR	0.975450	0.840310	2.68

Trên đây là các kết quả điểm R2 Score và thời gian chạy tương ứng với từng mô hình.

Ta có thể thấy mô hình có điểm R2 Score test tốt nhất là RandomForestRegressor sau khi Tuning.

Tuy nhiên điểm R2 Test của mô hình này với mô hình Random Forest Regressor ban đầu không chênh lệch nhau quá nhiều nhưng thời gian chạy của Random Forest Regressor Tuning lại hơn nhiều lần (Nếu bạn test trên colab thì kết quả còn cách xa hơn nhiều).

Model	R Squared(Train)	R Squared(Test)	Time Running
0 RandomForestRegressor Tuning	0.975377	0.913847	44.68
1 RandomForestRegressor	0.984763	0.912167	0.26
2 Linear Regression + Adaboosting	0.963682	0.898156	0.98
3 LinearRegression	0.962885	0.888641	0.07
4 SVR	0.975450	0.840310	35.24

Vì thế ta sẽ chọn mô hình Random Forest Regressor là mô hình cuối cùng để dự đoán giá xe dựa theo các Feature mà chúng ta đã chọn (Mô hình ở index = 1)