

Report Car Price Prediction

1. GIỚI THIỆU CHUNG

1.1. Giới thiệu tổng quát về mục tiêu của báo cáo

1.1.1. Problem Statement:

Một công ty ô tô Trung Quốc Geely Auto muốn thâm nhập thị trường Mỹ bằng cách thành lập đơn vị sản xuất của họ ở đó và sản xuất ô tô để cạnh tranh với các đối tác Mỹ và châu Âu. Do vậy, họ đã ký hợp đồng với một công ty tư vấn ô tô để hiểu các yếu tố ảnh hưởng đến việc định giá ô tô tại thị trường Mỹ, vì những yếu tố đó có thể rất khác so với thị trường Trung Quốc. Cụ thể, họ muốn biết:

- Đâu là những yếu tố quan trọng để định giá ô tô?
- Các thông số nào của ô tô sẽ ảnh hưởng đến giá của chiếc ô tô đó?

1.1.2. Business Goal:

Mục tiêu báo cáo là xây dựng một mô hình giá ô tô với các features trong tập dữ liệu cho team Product sử dụng để hiểu chính xác mức giá thay đổi như thế nào với các features đó. Theo đó, họ có thể đưa ra các phương án thiết kế của ô tô, chiến lược kinh doanh, v.v. để đáp ứng với từng phân khúc giá nhất định. Hơn nữa, mô hình này sẽ là một cách tốt để BOD hiểu được cách định giá của một thị trường mới.

1.2. Giới thiệu một số biến nổi bật, quan trọng

Việc định giá xe ô tô phụ thuộc vào nhiều yếu tố và thông số khác nhau. Dưới đây là một số yếu tố và thông số quan trọng thường ảnh hưởng đến quá trình định giá xe ô tô:

- Thương hiệu (brand) và mẫu xe (carbody): Nhãn hiệu và mẫu xe có sự ảnh hưởng lớn đến giá trị xe. Các thương hiệu nổi tiếng và các mẫu xe phổ biến thường có giá cao hơn so với các thương hiệu và mẫu xe ít được biết đến.
- Động cơ (enginetype, enginesize) và hiệu suất (horsepower, peakrpm): Loại động cơ, công suất và hiệu suất của xe cũng có thể ảnh hưởng đến giá trị. Xe với động cơ mạnh mẽ và hiệu suất tốt thường có giá trị cao hơn.
- Tiện nghi và tính năng: Các tiện nghi và tính năng bổ sung trên xe như hệ thống âm thanh, hệ thống điều hòa không khí, hệ thống giải trí, hệ thống an toàn và các tính năng khác cũng có thể ảnh hưởng đến giá trị xe.

2. DATA PRE-PROCESSING

2.1. Mô tả những thông tin chung của bảng data

Tập dữ liệu là 1 bảng tổng hợp các thông số và giá bán của một số dòng xe ô tô từ nhiều hãng, gồm 205 dòng và 26 cột, không chứa dữ liệu trùng lặp và không bị null.

Dữ liệu thống kê cho thấy có 1 sự chênh lệch lớn giữa giá tối thiểu và giá tối đa, khả năng cao là có outlier ở cột giá.

```
df.describe()

  curbweight enginesize boreratio      stroke compressionratio horsepower    peakrpm   citympg highwaympg      price
205.000000  205.000000 205.000000  205.000000  205.000000 205.000000 205.000000 205.000000 205.000000 2.050000e+02
2555.565854 126.907317 316.863415 329.492683  74.565854 104.117073 5125.121951 25.219512 30.751220 1.024690e+05
520.680204  41.642693 68.658153 429.105243  99.368850 39.544167 476.985643 6.542142 6.886443 1.246484e+06
1488.000000 61.000000 35.000000 28.000000  7.000000 48.000000 4150.000000 13.000000 16.000000 5.118000e+03
2145.000000 97.000000 305.000000 268.000000  9.000000 70.000000 4800.000000 19.000000 25.000000 7.788000e+03
2414.000000 120.000000 327.000000 319.000000  85.000000 95.000000 5200.000000 24.000000 30.000000 1.059500e+04
2935.000000 141.000000 354.000000 339.000000  94.000000 116.000000 5500.000000 30.000000 34.000000 1.655800e+04
4066.000000 326.000000 394.000000 3255.000000 941.000000 288.000000 6600.000000 49.000000 54.000000 1.785917e+07

  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   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    int64  
 10  carlength      205 non-null    int64  
 11  carwidth       205 non-null    int64  
 12  carheight      205 non-null    int64  
 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    int64  
 19  stroke          205 non-null    int64  
 20  compressionratio 205 non-null    int64  
 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    int64  
dtypes: int64(16), object(10)
memory usage: 41.8+ KB

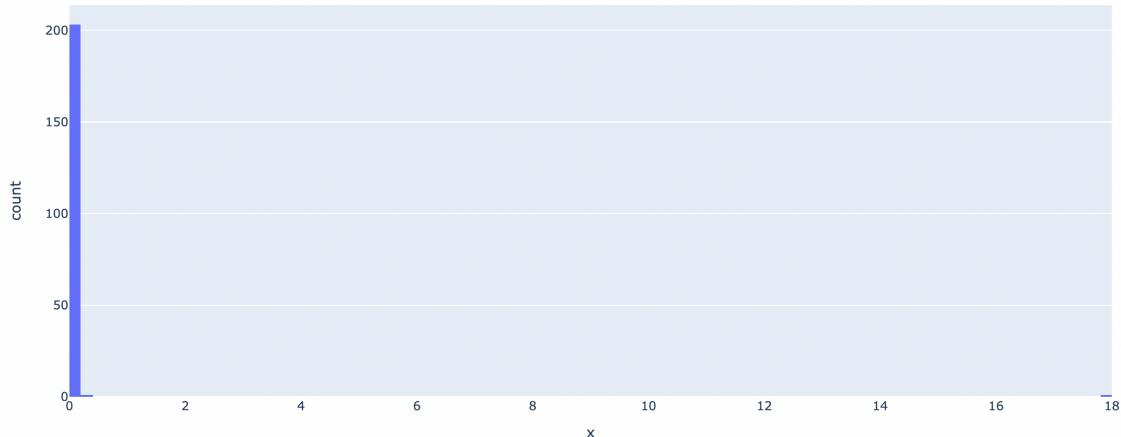
  df.isnull().sum()
car_ID               0
symboling            0
CarName              0
fuelytype             0
aspiration            0
doornumber            0
carbody               0
drivewheel            0
enginelocation        0
wheelbase              0
carlength              0
carwidth               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

[141] df.duplicated().sum()
0
```

2.2. Kiểm tra và xử lý outlier

Qua biểu đồ tần suất phát hiện có 2 outlier ở mức giá 17-18mil USD và 200-390k USD, tuy nhiên search google giá của 2 dòng xe đó thì không cao như trong data, nên có thể kết luận data bị lỗi typing, cần xoá outlier để chuẩn bị cho mô hình dự báo ở bước sau

```
px.histogram(df, df['price']/1000000, nbins=100)
```



```
df[df['price']>200000]
```

```
car_ID symboling CarName fuelytype aspiration doornumber carbody drivewheel enginelocation wheelbase ... enginesize fuels...
```

9	10	0	audi 5000s (diesel)	gas	turbo	two	hatchback	4wd	front	995	...	131
129	130	1	porsche cayenne	gas	std	two	hatchback	rwd	front	984	...	203

2 rows x 26 columns

```
df.drop(9,inplace=True)  
df.drop(129,inplace=True)
```

2.3. Loại bỏ các cột không cần thiết

Nhận thấy thương hiệu cũng là một yếu tố quan trọng ảnh hưởng đến việc định giá xe, do đó cần tạo thêm một cột ‘brand’ lấy thông tin từ cột ‘CarName’.

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

```
[148] df['brand'].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)
```

Có một số dòng bị sai chính tả, điều chỉnh lại như sau:

```
[149] df['brand'] = df['brand'].str.replace('vw', 'volkswagen')
      df['brand'] = df['brand'].str.replace('vokswagen', 'volkswagen')
      df['brand'] = df['brand'].str.replace('toyouta', 'toyota')
      df['brand'] = df['brand'].str.replace('maxda', 'mazda')
      df['brand'] = df['brand'].str.replace('porcshce', 'porsche')
      df['brand'] = df['brand'].str.replace('Nissan', 'nissan')
```

```
[150] df['brand'].unique()
```

```
array(['alfa-romero', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda',
       'isuzu', 'jaguar', 'mazda', 'buick', 'mercury', 'mitsubishi',
       'nissan', 'peugeot', 'plymouth', 'porsche', 'renault', 'saab',
       'subaru', 'toyota', 'volkswagen', 'volvo'], dtype=object)
```

Xoá 2 cột không cần thiết là ‘CarName’ và ‘car_ID’

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

2.4. Mã hoá dữ liệu

Mã hoá dữ liệu bằng phương pháp LabelEncoder để thuận tiện cho việc xem xét sự tương quan giữa các biến và chuẩn bị dữ liệu cho mô hình dự báo

```
| #Mã hóa data cho bước xây dựng mô hình
#Chọn data object để mã hóa
selected_columns = df.select_dtypes(include=['object'])
#Mã hóa bằng LabelEncoder
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for column in selected_columns.columns:
    df[column] = label_encoder.fit_transform(df[column])
```

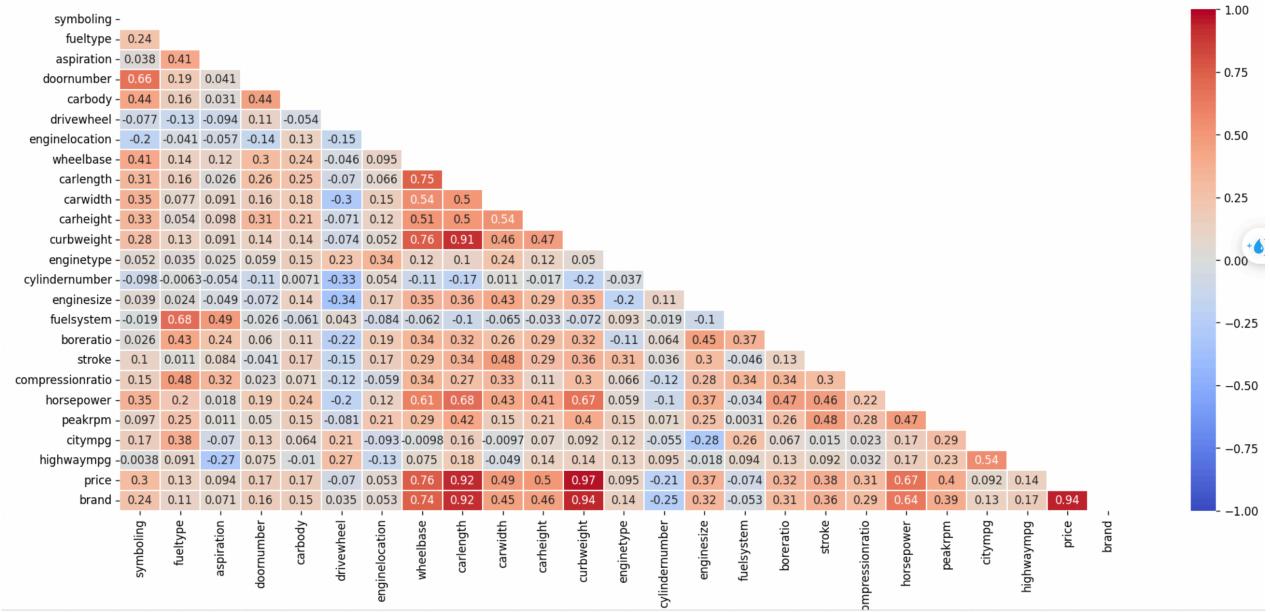
```
df.head(5)
```

	symboling	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth	...	fuel
0	3	1	0	1	0	2		0	886	1688	641	...
1	3	1	0	1	0	2		0	886	1688	641	...
2	1	1	0	1	2	2		0	945	1712	655	...
3	2	1	0	0	3	1		0	998	1766	662	...
4	2	1	0	0	3	0		0	994	1766	664	...

5 rows × 25 columns

3. PHÂN TÍCH MÔ TẢ, PHÂN TÍCH CHUẨN ĐOÁN/EDA

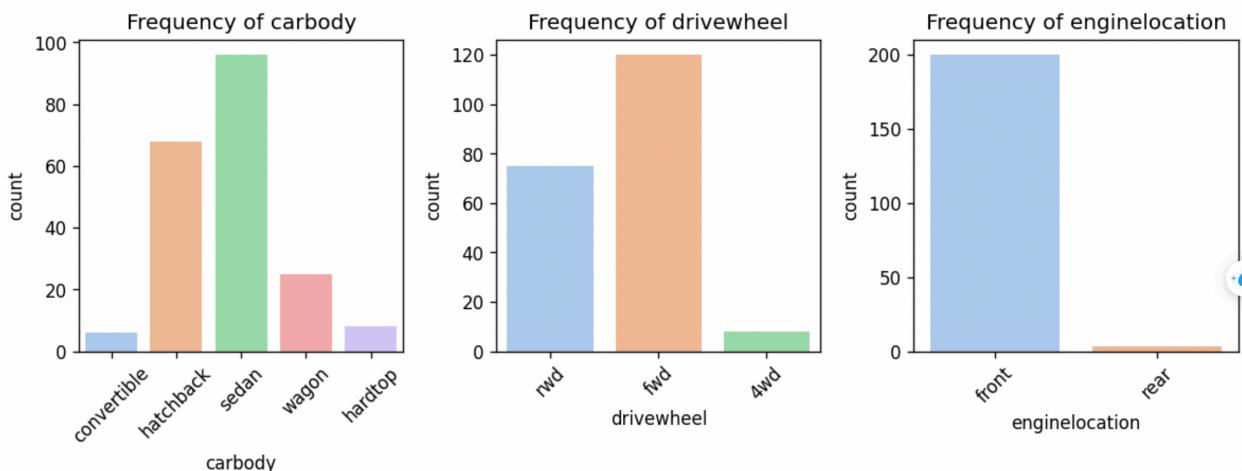
Kiểm tra sự tương quan của các biến bằng ma trận tương quan Pearson

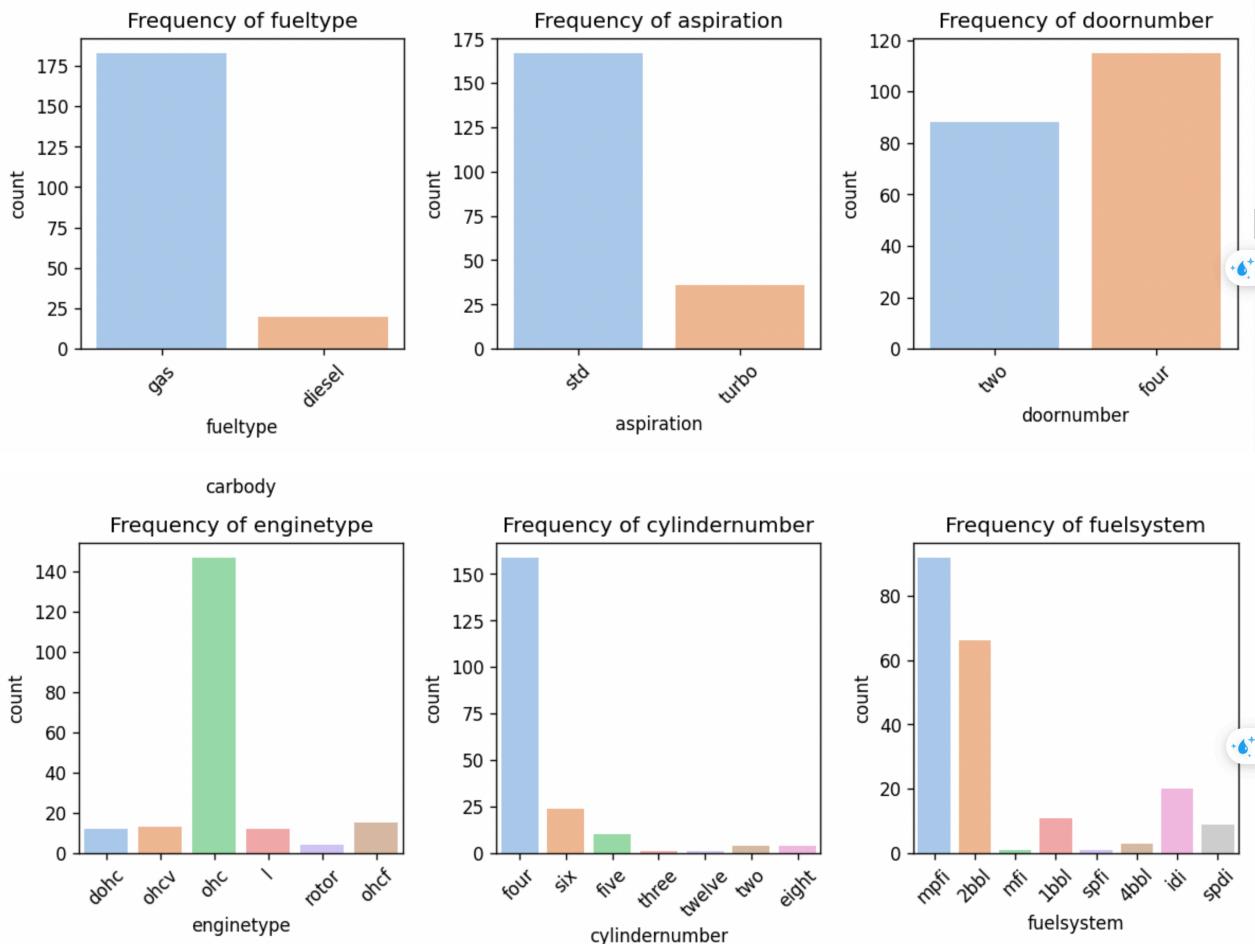


Nhân xét:

- Các yếu tố tương quan cao với giá là (lựa chọn các biến có tương quan lớn hơn 0.3): wheelbase, carlength, carwidth, carheight, curbweight, enginesize, bore ratio, stroke, compression ratio, horsepower, peakrpm, brand.
- Carlength tương quan cao với curbweight, wheelbase cũng tương quan cao với carlength và curbweight, nên khi lựa chọn biến để chạy mô hình dự báo thì chỉ cần lấy 1 biến curbweight.

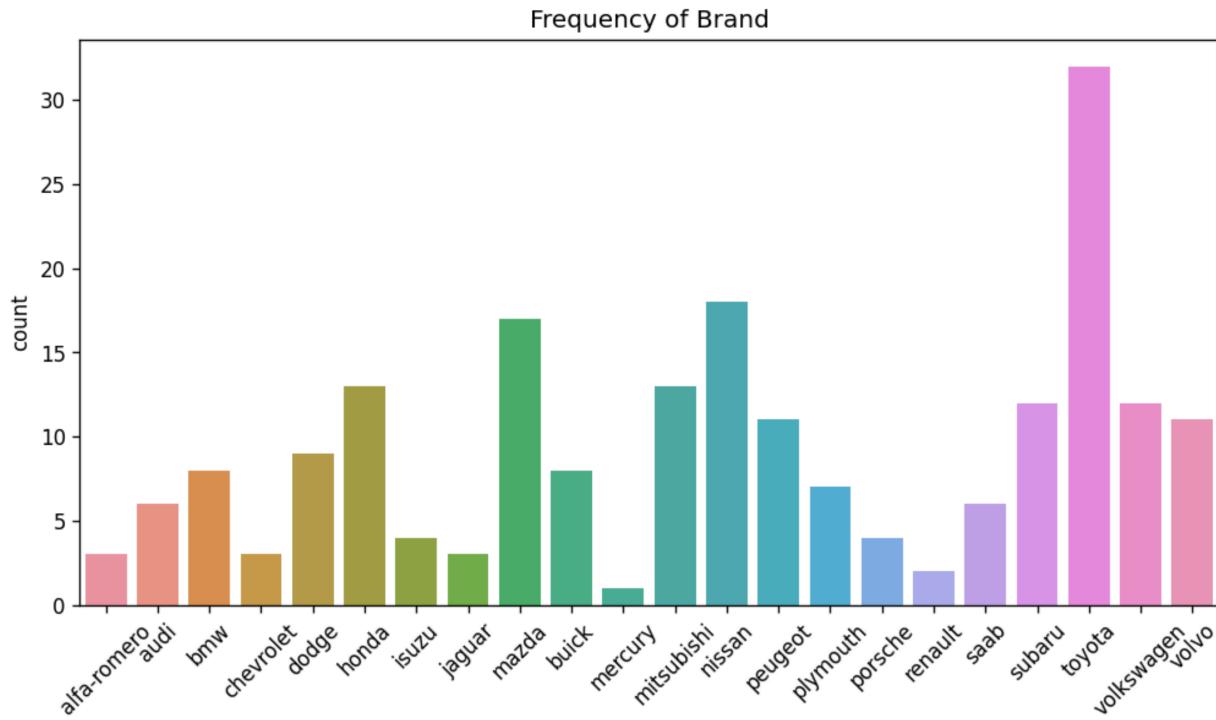
Kiểm tra tần suất của các biến có dtype là object bằng biểu đồ countplot



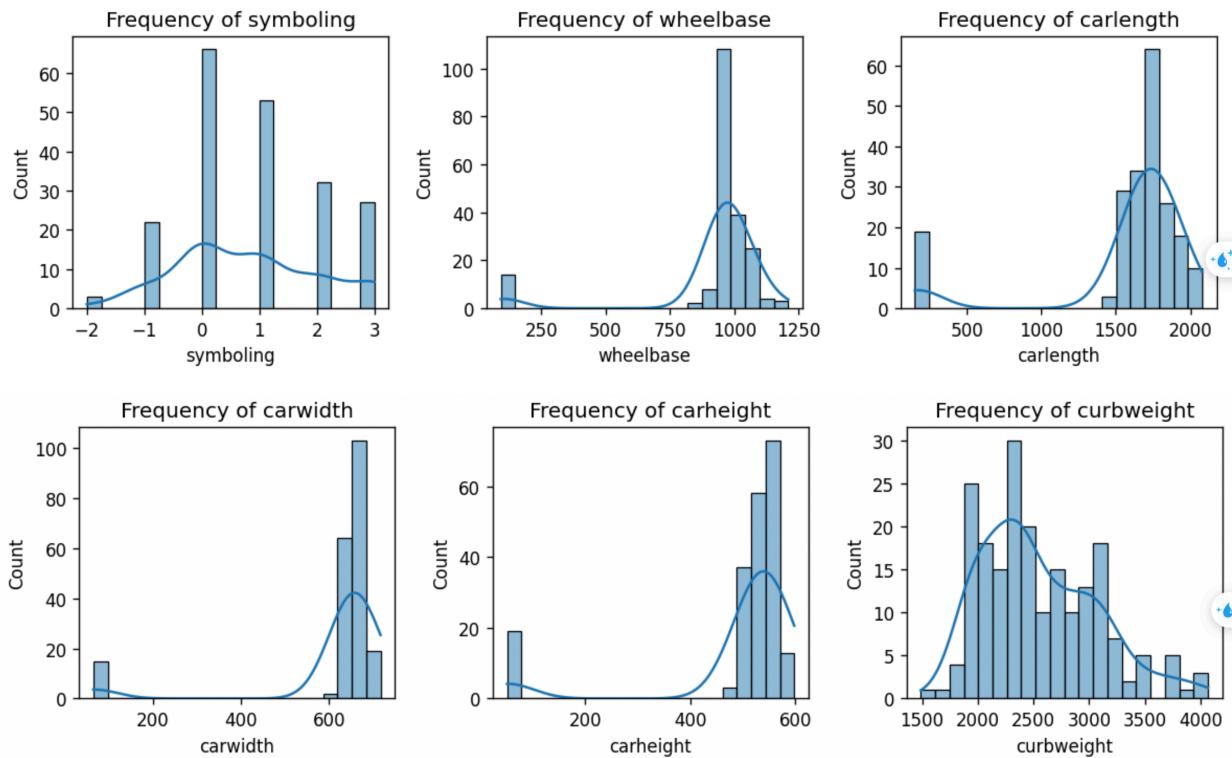


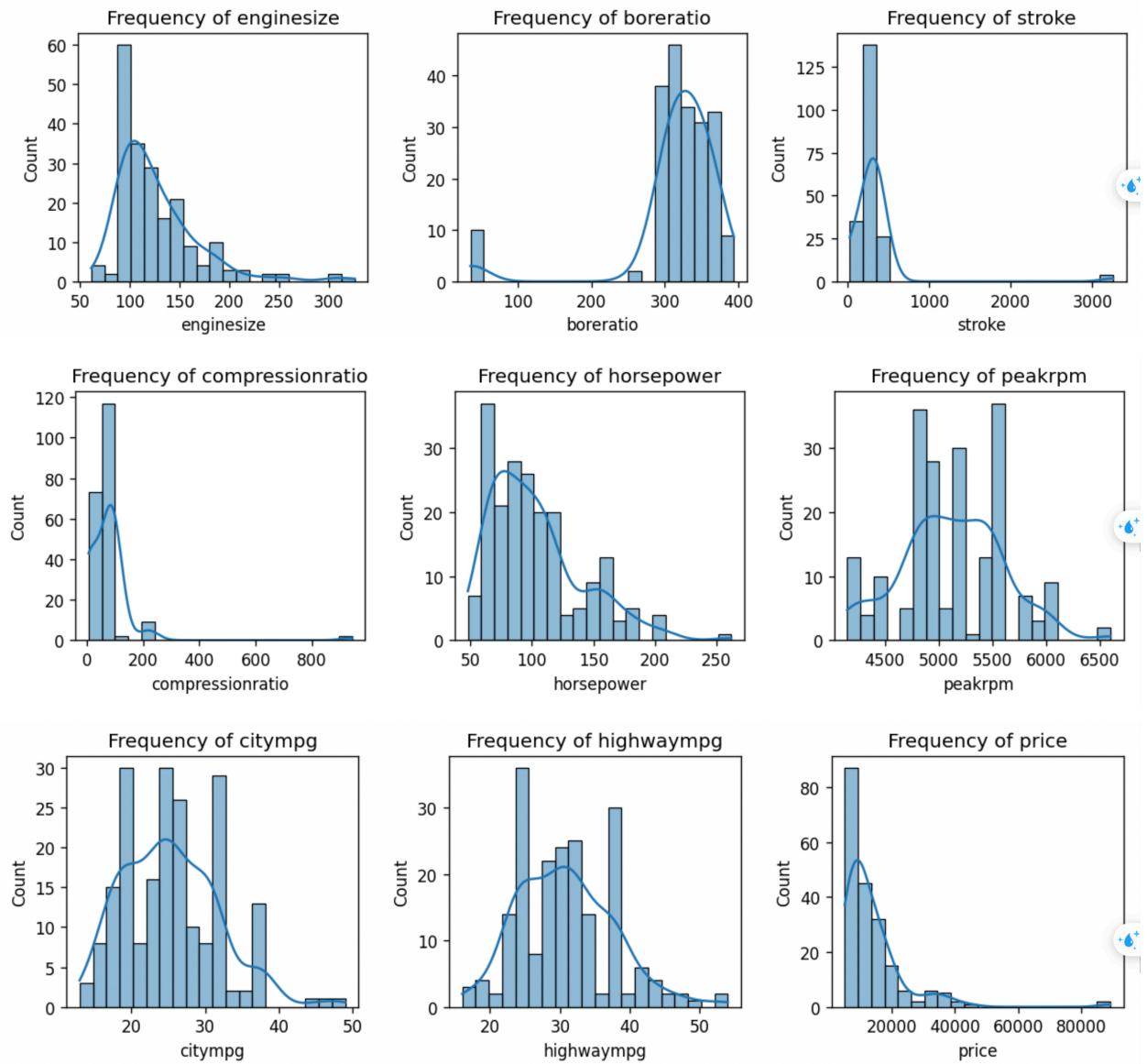
Nhân xét:

- Fueltype (Nhiên liệu): chủ yếu là chạy bằng xăng.
- Aspiration (Hệ thống hút khí vào động cơ): phần lớn là hệ thống hút khí thông thường.
- Doornumber (số cửa xe): tuy không chênh lệch nhiều nhưng xe có thiết kế 4 cửa vẫn chiếm phần nhiều hơn.
- Carbody (thân xe): 2 dòng xe phổ biến nhất lần lượt là sedan và hatchback.
- Drivewheel (hệ thống dẫn động): chủ yếu là fwd (hệ thống dẫn động cầu trước).
- Enginelocation (vị trí đặt động cơ): đa phần đặt phía trước.
- Enginetype: loại động cơ nồi trội và chiếm đa số là OHC (Overhead Camshaft: Trục cam trên đầu)
- Cylindernumber: số lượng xi-lanh thường là 4
- Fuelsystem: Hệ thống nhiên liệu phần lớn là mpfi, 2bbl
- Biến ‘brand’ có mối tương quan rất cao với biến dự đoán ‘price’, và trong các brand thì Toyota là thương hiệu phổ biến nhất, sau đó là Nissan và Mazda



Kiểm tra tần suất của các biến có dtype là numerical bằng biểu đồ histplot





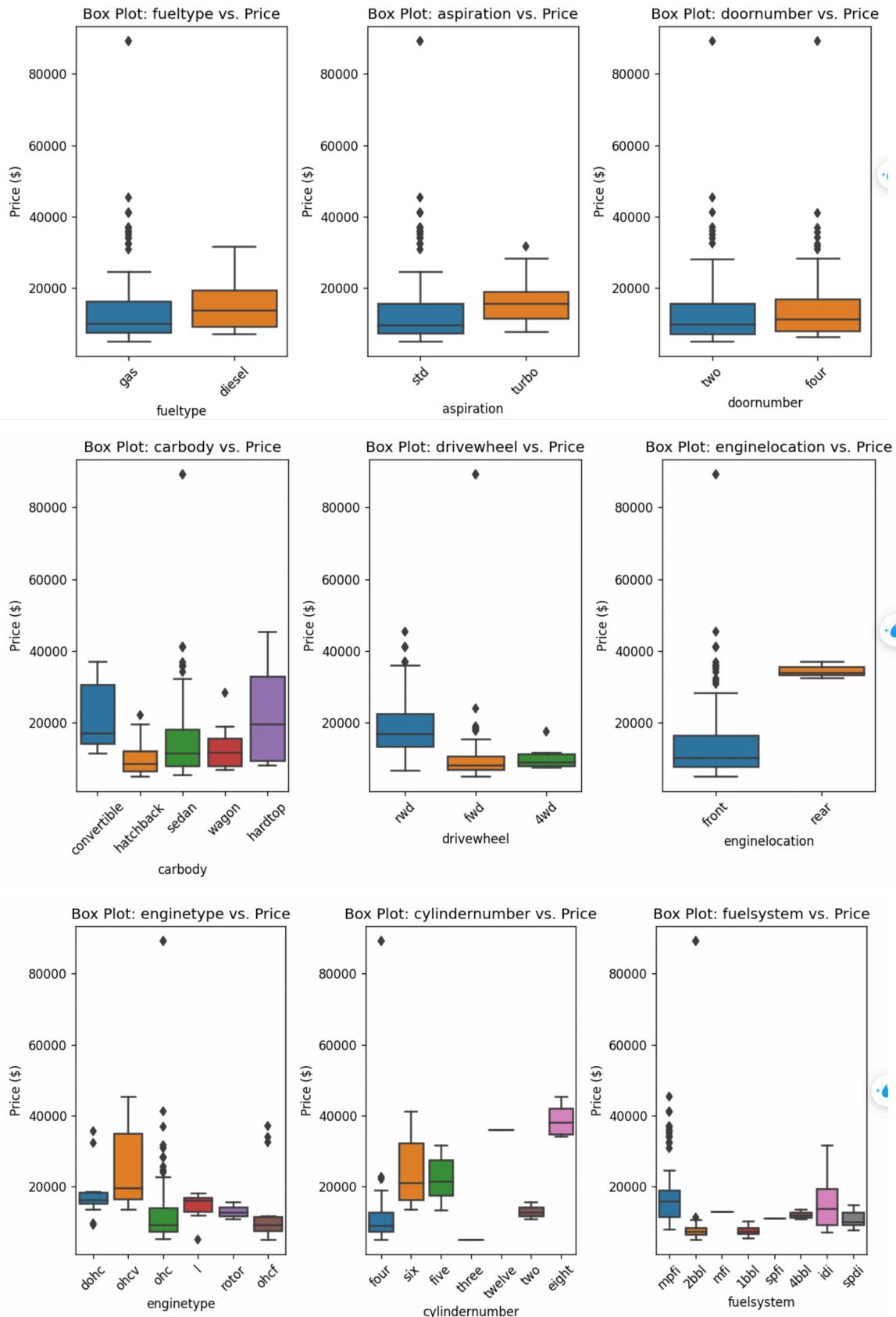
Nhân xét:

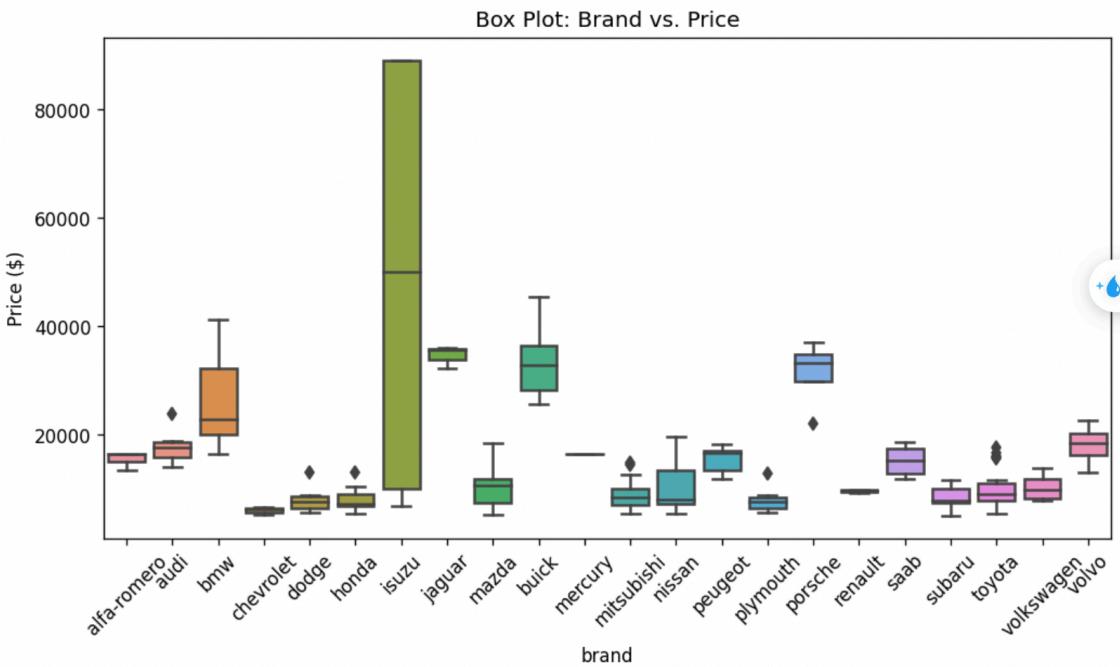
- Symboling (đánh giá an toàn bảo hiểm): không có xe nào đạt ở mức an toàn tuyệt đối, phần lớn ở mức 0-3 theo thứ tự giảm dần.
- Wheelcase (khoảng cách 2 trực bánh xe): trung bình là 1000, có nhiều outlier ở mức thấp hơn 250 (dữ liệu không hợp lý).
- Carlength (chiều dài xe): dao động từ 1500-2000, trung bình ở mức 1700, có nhiều outlier ở mức thấp hơn 200 (dữ liệu không hợp lý).
- Carwidth (chiều rộng xe): dao động từ 600-700, có nhiều outlier thấp hơn 100 là không hợp lý.
- Carheight (chiều cao xe): dao động từ 500-600, nhiều outlier thấp hơn 100 là ko hợp lý
- Curbweight (trọng lượng xe): dao động từ 2000-3000.
- Enginesize (dung tích xi lanh): thường dao động từ 90-120.

- Boreratio (tỉ số đường kính xylanh): dao động trong khoảng 300-400, có 1 số outlier thấp hơn 50 là ko hợp lý.
- Stroke (hành trình piston): trung bình ở mức 300, có outlier ở mức 3000.
- Compressionratio (tỉ số nén): trung bình là 100, có outlier ở mức 800.
- Horsepower (công suất): phần lớn ở mức 60-120 mã lực.
- PeakPRM (hiệu suất): phần nhiều ở mức 4700-5600.
- Price: xe giá rẻ chiếm đa số, thường ít hơn 20000 usd, vẫn có outlier lớn hơn 80000 usd.
- Cân xem xét lại tính chính xác của dữ liệu

Kiểm tra giá của các biến có dtype là object bằng biểu đồ boxplot, rút được các nhận xét sau:

- Có outlier ở mức giá lớn hơn 80k, nằm khá xa các điểm dữ liệu còn lại
- Không có sự khác biệt nhiều về giá giữa các nhãn/giá trị của các biến fueltype, aspiration, doornumber.
- Giá của xe hardtop và convertible hơi cao hơn một chút so với các dòng khác
- Các dòng xe dùng hệ thống dẫn động cầu sau (rwd) thường dùng trong các xe hạng sang và xe thể thao thì có giá cao hơn là fwd hay 4wd.
- Có một sự chênh lệch cao về giá giữa các giá trị của biến enginelocation, cụ thể là xe có động cơ đặt phía sau (thường được sử dụng trong một số loại xe đặc biệt như một số mẫu xe thể thao Porsche 911 và xe hơi cổ Volkswagen Beetle) thì có giá cao hơn so với xe có động cơ đặt phía trước (là vị trí phổ biến nhất trong hầu hết các loại xe hơi, bao gồm xe hạng nhỏ, xe gia đình, xe thể thao và nhiều loại khác).
- Xe có nhiều xylanh thì công suất và hiệu suất của động cơ càng lớn, giá sẽ càng cao.
- Xe sử dụng động cơ OHCV có giá hơi cao hơn xe sử dụng các loại động cơ khác.
- Outlier của biến giá nằm ở brand Isuzu. Search giá xe Isuzu Dmax 2023 dao động khoảng 34k USD nên có thể cân nhắc loại bỏ để mô hình dự báo được chính xác hơn.





4. PREDICTION MODEL

Dự báo các yếu tố ảnh hưởng đến giá xe là một quá trình quan trọng trong lĩnh vực phân tích thị trường ô tô và dự báo giá cả. Việc dự báo đúng các yếu tố ảnh hưởng đến giá xe có thể mang lại nhiều lợi ích quan trọng như sau:

- Hỗ trợ quyết định kinh doanh: Dự báo chính xác các yếu tố ảnh hưởng đến giá xe có thể giúp các nhà sản xuất ô tô, nhà phân phối và đại lý ô tô hiểu được xu hướng thị trường và thay đổi trong nhu cầu của khách hàng. Điều này giúp họ đưa ra các quyết định kinh doanh thông minh, bao gồm quyết định về lựa chọn sản phẩm, giá cả, chiến lược tiếp thị và quản lý dự trữ.
- Dự báo giá cả: Dự báo chính xác các yếu tố ảnh hưởng đến giá xe có thể giúp người mua và người bán dự đoán giá trị của các loại xe trong tương lai. Điều này hỗ trợ việc đưa ra quyết định mua hoặc bán xe với giá hợp lý và tối ưu hóa lợi nhuận.
- Phân tích thị trường: Dự báo chính xác các yếu tố ảnh hưởng đến giá xe cung cấp thông tin quan trọng để phân tích thị trường ô tô. Nó giúp nhà nghiên cứu và chuyên gia phân tích hiểu và dự đoán xu hướng thị trường, tìm hiểu ưu điểm và nhược điểm của các loại xe, và đưa ra đề xuất cải tiến cho sản phẩm và chiến lược kinh doanh.
- Dự báo tài chính: Dự báo giá xe và các yếu tố ảnh hưởng liên quan cũng có thể hỗ trợ dự báo tài chính, bao gồm việc dự đoán doanh thu, lợi nhuận và các chỉ số tài chính khác của các công ty ô tô và ngành công nghiệp liên quan.

Tóm lại, việc dự báo đúng các yếu tố ảnh hưởng đến giá xe có tầm quan trọng lớn trong việc hỗ trợ quyết định kinh doanh, dự báo giá cả, phân tích thị trường và dự báo tài chính. Nó giúp tăng tính hiệu

quả và sự cạnh tranh trong ngành công nghiệp ô tô và là công cụ hữu ích để đưa ra các quyết định thông minh và thành công.

Trong bài phân tích này, sẽ sử dụng ba mô hình dự báo gồm Linear Regression, Decision Tree và Random Forest.

4.1. Xây dựng mô hình hồi quy (3 mô hình) với toàn bộ bảng data

Chọn biến và chia tập dữ liệu training và test

```
#Chọn dữ liệu X và y
y=df['price']
X=df.drop(columns = 'price', inplace = False)

#Chia dữ liệu thành training set và test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=0)
```

4.1.1. Mô hình Linear Regression

Xây dựng mô hình

```
▶ #Xây dựng mô hình
from sklearn.linear_model import LinearRegression
model_lin= LinearRegression()
model_lin.fit(X_train,y_train)
print('a=',model_lin.coef_)
print('b=',model_lin.intercept_)

⇒ a= [-7.98516047e+02 -3.50143193e+02 -1.31562048e+03  3.99792662e+03
      1.78042206e+03 -1.85231880e+02  2.56144357e+04  1.67560940e-01
      -1.69050835e+00  6.31357953e+00 -1.23965695e+01  8.48316436e+00
      -2.04633804e+02 -2.60995727e+03  6.59325461e+01  6.39319503e+02
      -1.12263170e-01  4.54711864e+00 -9.62025807e+00  4.86908686e+01
      2.18454317e+00  8.75973588e+02 -2.88162554e+02 -3.39363938e+02]
b= -39017.63641701072
```

Đánh giá mô hình

```
[ ] #Dự báo với test set
y_pred_lin=model_lin.predict(X_test)

[ ] #Đánh giá mô hình bằng chỉ số r-square
from sklearn.metrics import r2_score
Ac2_lin=r2_score(y_test, y_pred_lin)
print('R-square=','{:.2f}'.format(Ac2_lin))
```

R-square= 0.24

▶ #Đánh giá mô hình bằng chỉ số root mean squared error(RMSE)

```
from sklearn.metrics import mean_squared_error
import math
rmse_lin = math.sqrt(mean_squared_error(y_test, y_pred_lin))
print('RMSE=','{:.2f}'.format(rmse_lin))
```

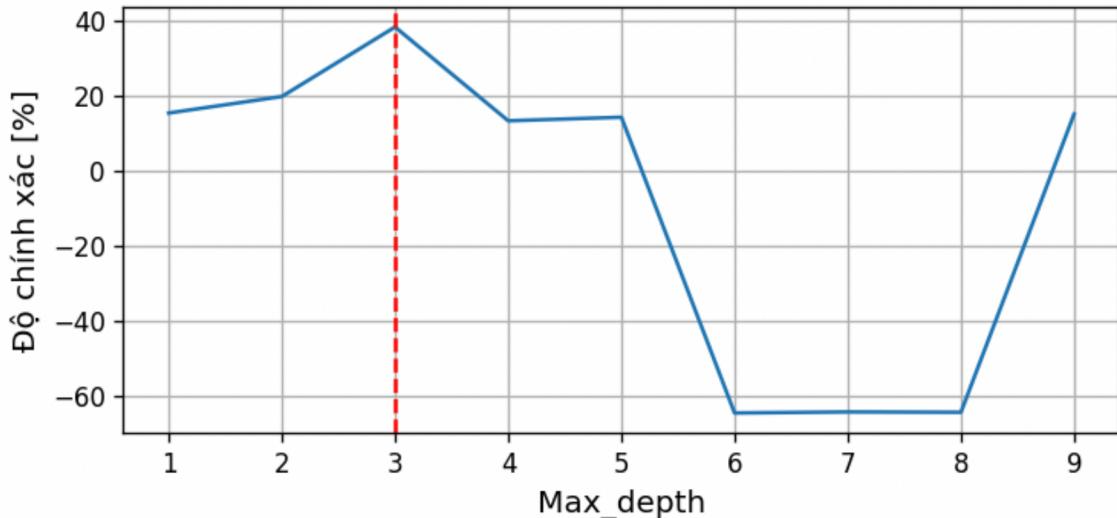
→ RMSE= 10378.21

4.1.2. Mô hình Decision Tree for regression

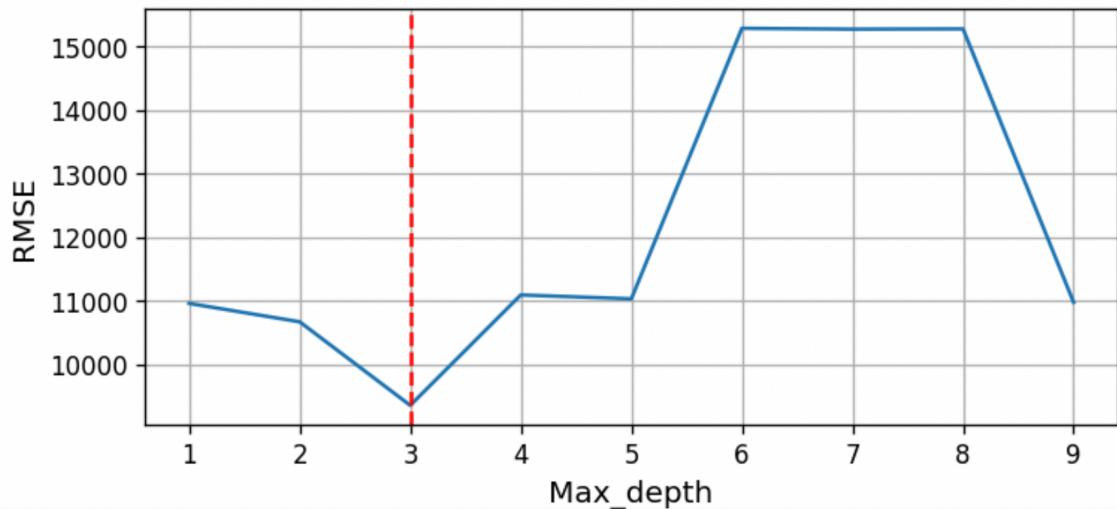
Dùng vòng lặp for để tìm max_depth tối ưu

```
# Dùng vòng lặp tìm hyperparameter tối ưu để xây dựng và đánh giá mô hình
from sklearn import tree
import math
score_total_dt=[]
rmse_total_dt=[]
for i in range(1,10):
    model_tree = tree.DecisionTreeRegressor(max_depth=i)
    model_tree.fit(X_train, y_train)
    y_pred_dt=model_tree.predict(X_test)
    Ac2_dt=r2_score(y_test, y_pred_dt)
    rmse_dt = math.sqrt(mean_squared_error(y_test, y_pred_dt))
    score_total_dt.append(Ac2_dt*100)
    rmse_total_dt.append(rmse_dt)
```

```
#Vẽ hình thế hiện độ chính xác theo giá trị max_depth
plt.rcParams.update({'figure.figsize':(7,3), 'figure.dpi':120})
plt.ylabel('Độ chính xác [%]', fontsize=12)
plt.xlabel('Max_depth', fontsize=12)
plt.plot(range(1, 10), score_total_dt)
plt.xticks(range(1, 10))
plt.axvline(x=3, color='r', linestyle='--')
plt.grid('minor')
```



```
#Vẽ hình thế hiện độ lệch theo giá trị max_depth
plt.rcParams.update({'figure.figsize':(7,3), 'figure.dpi':120})
plt.ylabel('RMSE', fontsize=12)
plt.xlabel('Max_depth', fontsize=12)
plt.plot(range(1, 10), rmse_total_dt)
plt.xticks(range(1, 10))
plt.axvline(x=3, color='r', linestyle='--')
plt.grid('minor')
```

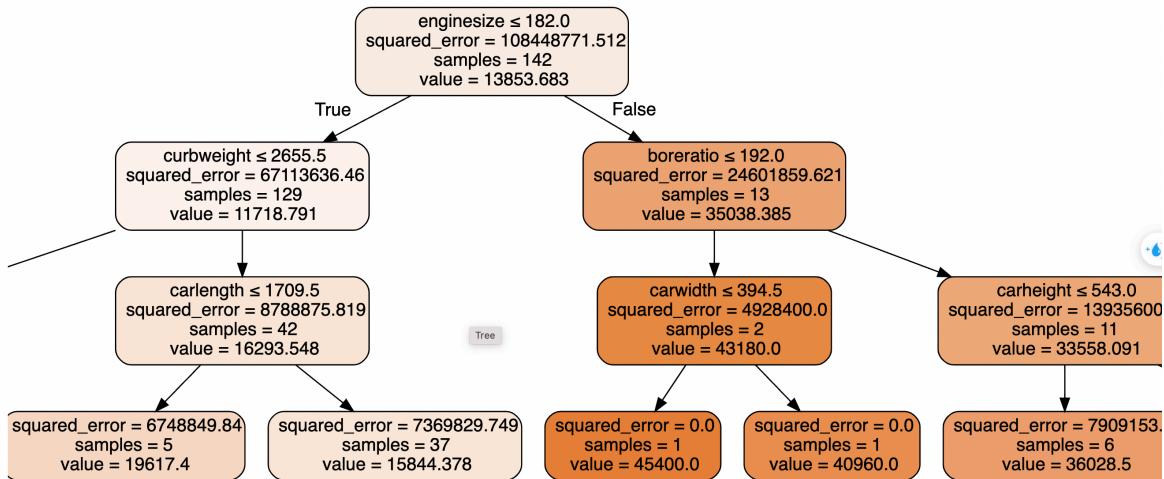


Xây dựng và đánh giá mô hình với tham số tối ưu

```
#Xây dựng và đánh giá mô hình với tham số tối ưu
model_tree = tree.DecisionTreeRegressor(max_depth=3)
model_tree.fit(X_train, y_train)
y_pred_dt=model_tree.predict(X_test)
Ac2_dt=r2_score(y_test, y_pred_dt)
rmse_dt = math.sqrt(mean_squared_error(y_test, y_pred_dt))
print('R_square: ',Ac2_dt)
print('RMSE: ',rmse_dt)
```

R_square: 0.3831069528742649
RMSE: 9354.051767000858

Vẽ hình cây quyết định



4.1.3. Mô hình Random Forest for regression

Xây dựng và đánh giá mô hình với hyperparameter bất kỳ

```
#Xây dựng mô hình với n_estimators bất kỳ
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=3)
rf.fit(X_train, y_train)
y_pred_rf=rf.predict(X_test)
```

```
#Đánh giá mô hình
Ac2_rf=r2_score(y_test, y_pred_rf)
print("R2=", Ac2_rf)
rmse_rf = math.sqrt(mean_squared_error(y_test, y_pred_rf))
print('RMSE=',"{:.2f}".format(rmse_rf))
```

R2= -0.2746229818101491
RMSE= 13445.77

Lựa chọn hyperparameter tối ưu

```
#Lựa chọn hyperparameters tối ưu
from sklearn.model_selection import GridSearchCV
rf = RandomForestRegressor()
grid_space = {'max_depth':[1,2,3,4,5,6,7,8,9,10, None],
              'n_estimators':[1,2,3,4,5,6,7,8,9,10,15,20,25,30,35,40,45,50],
              'criterion':['squared_error', 'absolute_error'],
              }
rf_grid = GridSearchCV(rf, param_grid=grid_space, scoring='neg_mean_squared_error', cv=5)
model_grid = rf_grid.fit(X_train, y_train)
best_rf = rf_grid.best_estimator_
print("Hyperparameters tối ưu:", rf_grid.best_params_)
y_pred_brf = best_rf.predict(X_test)
Ac2_brf=r2_score(y_test, y_pred_brf)
print("R2=", Ac2_brf)
rmse_brf = math.sqrt(mean_squared_error(y_test, y_pred_brf))
print("Root Mean Squared Error trên tập kiểm tra:", rmse_brf)

Hyperparameters tối ưu: {'criterion': 'absolute_error', 'max_depth': 4, 'n_estimators': 3}
R2= 0.17725399940732245
Root Mean Squared Error trên tập kiểm tra: 10802.585721695994
```

4.2. Xây dựng mô hình hồi quy (3 mô hình) với data đã được xử lý (xử lý outlier/ lựa chọn biến)

Như đã phân tích ở phần 3, biến price vẫn còn outlier nằm khá xa các điểm dữ liệu còn lại, dẫn đến việc các mô hình dự báo giá ở mục 4.1 không được chính xác, giá dự báo lệch rất nhiều. Do đó cần loại bỏ outlier để đạt được độ chính xác cao hơn trong các mô hình dự báo.

```
#Xoá outlier để tăng độ chính xác cho mô hình

df[df['price']>80000]

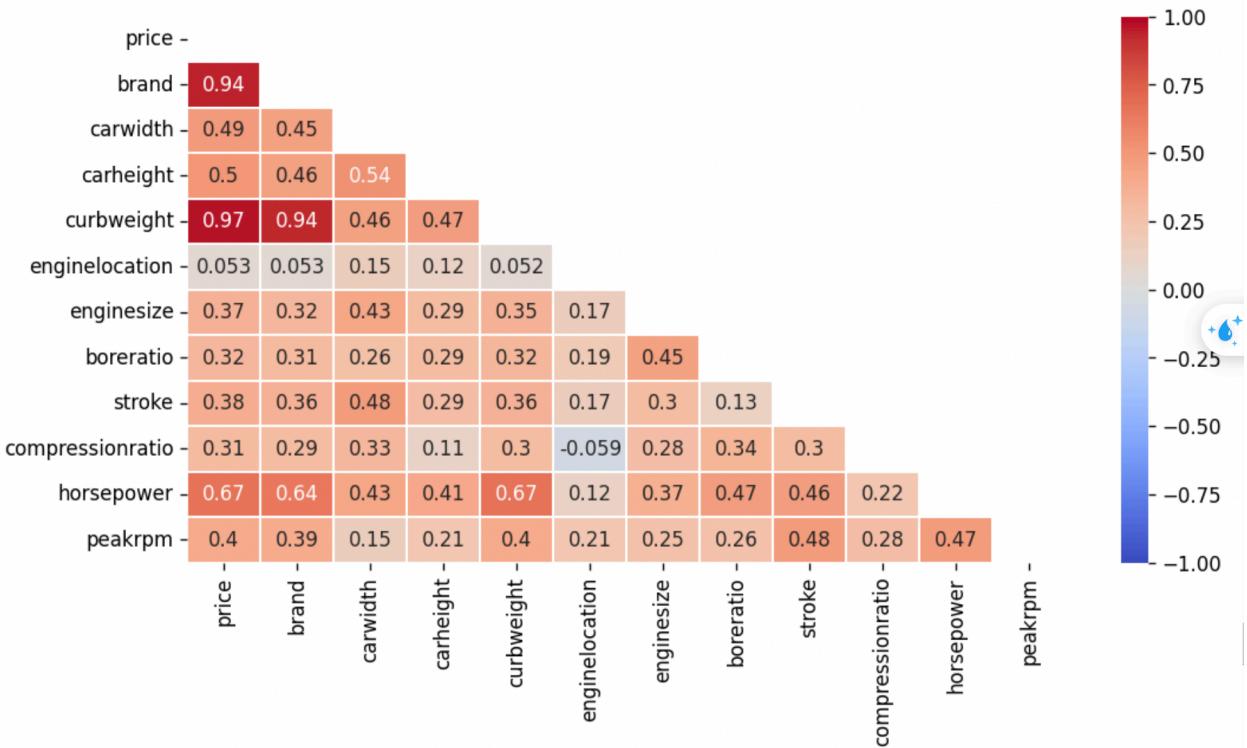
symboling fueltype aspiration doornumber carbody drivewheel enginelocation wheelbase carlength carwidth ... fuelsystem borera
44      1        1         0       1       3       1           0      945     1559      636     ...      1
45      0        1         0       0       3       1           0      945     1559      636     ...      1
2 rows x 25 columns

df.drop(44,inplace=True)
df.drop(45,inplace=True)
```

Ngoài ra, việc lựa chọn những biến quan trọng có ảnh hưởng nhiều đến việc định giá xe sẽ giúp:

- Tinh gọn hơn mô hình dự báo, giảm độ phức tạp của mô hình, dễ dàng hiểu hơn, giảm thiểu sự rối loạn và giúp tăng tính khả thi của việc triển khai mô hình.
- Tăng tốc độ tính toán, dự báo nhanh hơn, đặc biệt quan trọng khi làm việc với các tập dữ liệu lớn hoặc khi cần đưa ra dự báo trong thời gian ngắn.
- Tránh overfitting. Overfitting xảy ra khi mô hình quá phức tạp và quá khớp với dữ liệu huấn luyện, dẫn đến hiệu suất dự báo kém trên dữ liệu mới. Bằng cách lựa chọn một số biến quan trọng, có thể giảm nguy cơ overfitting và tạo ra một mô hình tổng quát hơn.
- Giảm chi phí nghiên cứu thị trường

Sau khi kiểm tra sự tương quan của các biến bằng ma trận tương quan Pearson ở phần 3, ta chọn được một số biến quan trọng như sau:



Chọn biến và chia tập dữ liệu training và test

```
#Data New
y=df_new['price']
X=df_new.drop('price',axis='columns',inplace=False)
#Chia dữ liệu thành training set và test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=0)
```

4.2.1. Mô hình Linear Regression

```

# 1. Linear Regression
#Xây dựng mô hình
from sklearn.linear_model import LinearRegression
model_lin= LinearRegression()
model_lin.fit(X_train,y_train)
print('a=',model_lin.coef_)
print('b=',model_lin.intercept_)
#Dự báo với test set
y_pred_lin=model_lin.predict(X_test)
#Đánh giá mô hình bằng chỉ số r-square
from sklearn.metrics import r2_score
Ac2_lin=r2_score(y_test, y_pred_lin)
print('R-square=',"{:.2f}".format(Ac2_lin))
#Đánh giá mô hình bằng chỉ số root mean squared error(RMSE)
from sklearn.metrics import mean_squared_error
import math
rmse_lin = math.sqrt(mean_squared_error(y_test, y_pred_lin))
print('RMSE=',"{:.2f}".format(rmse_lin))

```

```

a= [-1.95672160e+02  4.91408555e-02  3.64410348e+00  7.51049354e+00
 -5.50670620e-14  6.72225244e+01  1.35816577e+00  8.07646274e-01
 -8.13786732e-01  7.51977365e+00  7.08733371e-01]
b= -19349.95519353119
R-square= 0.77
RMSE= 4454.26

```

4.2.2. Mô hình Decision Tree for regression

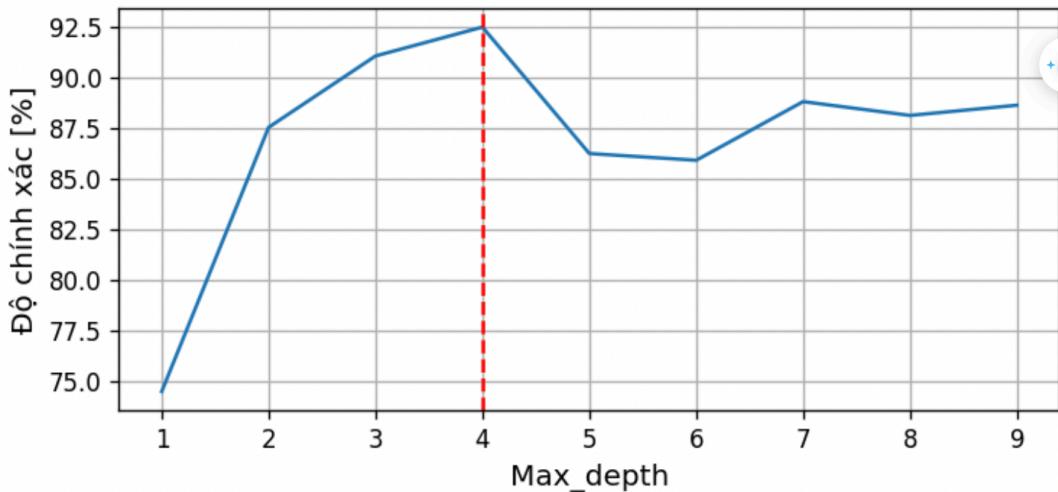
Tìm hyperparameter tối ưu

```

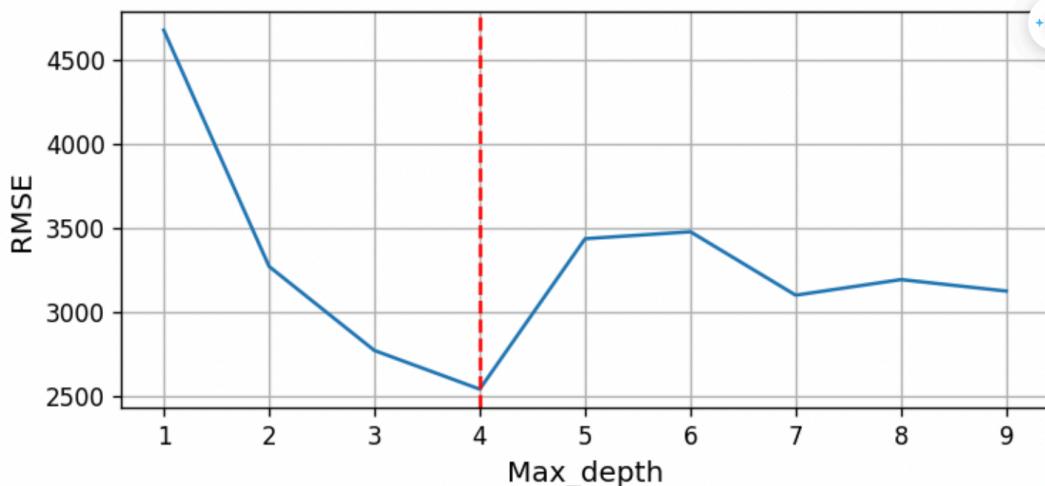
# Dùng vòng lặp tìm hyperparameter tối ưu để xây dựng và đánh giá mô hình
from sklearn import tree
import math
score_total_dt=[]
rmse_total_dt=[]
for i in range(1,10):
    model_tree = tree.DecisionTreeRegressor(max_depth=i)
    model_tree.fit(X_train, y_train)
    y_pred_dt=model_tree.predict(X_test)
    Ac2_dt=r2_score(y_test, y_pred_dt)
    rmse_dt = math.sqrt(mean_squared_error(y_test, y_pred_dt))
    score_total_dt.append(Ac2_dt*100)
    rmse_total_dt.append(rmse_dt)

```

```
#Vẽ hình thế hiện độ chính xác theo giá trị max_depth
plt.rcParams.update({'figure.figsize':(7,3), 'figure.dpi':120})
plt.ylabel('Độ chính xác [%]', fontsize=12)
plt.xlabel('Max_depth', fontsize=12)
plt.plot(range(1, 10), score_total_dt)
plt.xticks(range(1, 10))
plt.axvline(x=4, color='r', linestyle='--')
plt.grid('minor')
```



```
#Vẽ hình thế hiện độ lệch theo giá trị max_depth
plt.rcParams.update({'figure.figsize':(7,3), 'figure.dpi':120})
plt.ylabel('RMSE', fontsize=12)
plt.xlabel('Max_depth', fontsize=12)
plt.plot(range(1, 10), rmse_total_dt)
plt.xticks(range(1, 10))
plt.axvline(x=4, color='r', linestyle='--')
plt.grid('minor')
```

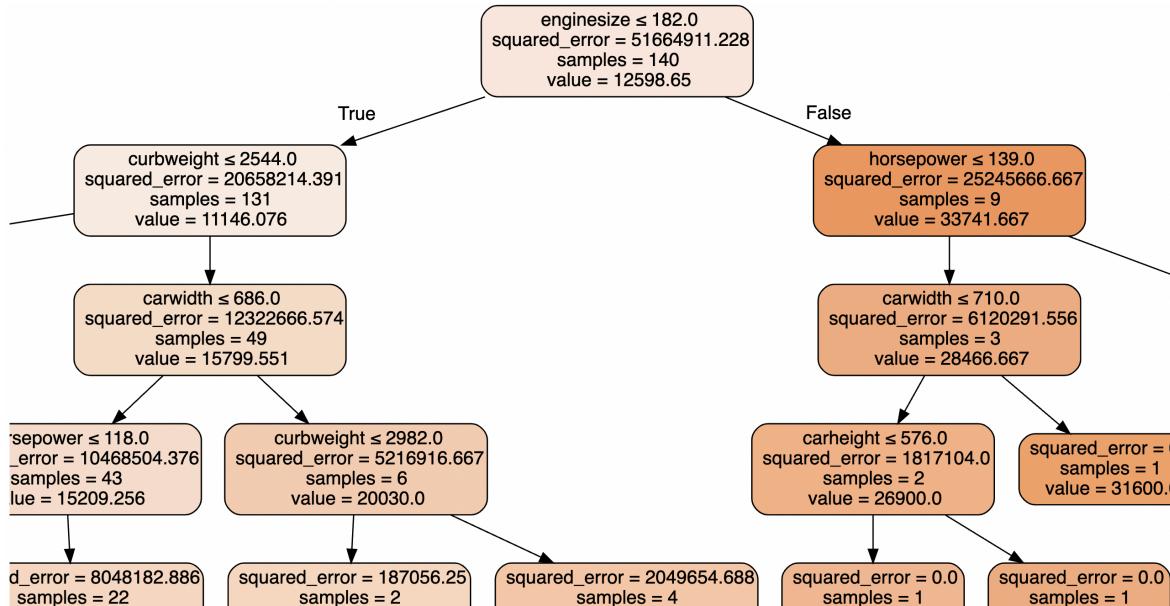


Xây dựng mô hình với tham số tối ưu

```
#Xây dựng mô hình với tham số tối ưu
model_tree = tree.DecisionTreeRegressor(max_depth=4)
model_tree.fit(X_train, y_train)
y_pred_dt=model_tree.predict(X_test)
Ac2_dt=r2_score(y_test, y_pred_dt)
rmse_dt = math.sqrt(mean_squared_error(y_test, y_pred_dt))
print('R_square: ',Ac2_dt)
print('RMSE: ',rmse_dt)
```

R_square: 0.9256729445210339
RMSE: 2523.7519781129795

Vẽ cây quyết định



4.2.3. Mô hình Random Forest for regression

Xây dựng và đánh giá mô hình với n_estimators bất kỳ sau đó cập nhật lại n_estimators tối ưu.

```
#Xây dựng mô hình với n_estimators bất kỳ
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=25)
rf.fit(X_train, y_train)
y_pred_rf=rf.predict(X_test)
#Đánh giá mô hình
Ac2_rf=r2_score(y_test, y_pred_rf)
print("R2=", Ac2_rf)
rmse_rf = math.sqrt(mean_squared_error(y_test, y_pred_rf))
print('RMSE=',{:.2f}.format(rmse_rf))
```

R2= 0.930596396032244
RMSE= 2438.73

Lựa chọn hyperparameter tối ưu

```

#Lựa chọn hyperparameters tối ưu
from sklearn.model_selection import GridSearchCV
rf = RandomForestRegressor()
grid_space = {'max_depth':[1,2,3,4,5,6,7,8,9,10, None],
              'n_estimators':[1,2,3,4,5,6,7,8,9,10,15,20,25,30,35,40,45,50],
              'criterion':['squared_error', 'absolute_error'],
             }
rf_grid = GridSearchCV(rf, param_grid=grid_space, scoring='neg_mean_squared_error', cv=5)
model_grid = rf_grid.fit(X_train, y_train)
best_rf = rf_grid.best_estimator_
print("Hyperparameters tối ưu:", rf_grid.best_params_)
y_pred_brf = best_rf.predict(X_test)
Ac2_brf=r2_score(y_test, y_pred_brf)
print("R2=", Ac2_brf)
rmse_brf = math.sqrt(mean_squared_error(y_test, y_pred_brf))
print("Root Mean Squared Error trên tập kiểm tra:", rmse_brf)

Hyperparameters tối ưu: {'criterion': 'squared_error', 'max_depth': None, 'n_estimators': 25}
R2= 0.8908110819314853
Root Mean Squared Error trên tập kiểm tra: 3058.8803496106766

```

Trong các mô hình dự báo đã thực hiện thì mô hình Random Forest kết hợp xử lý outlier và lựa chọn biến cho kết quả chính xác nhất và độ sai lệch thấp nhất.

5. KẾT LUẬN

Từ các phân tích ở trên có thể thấy được việc định giá của xe ô tô phụ thuộc một số biến quan trọng như: 'brand', 'carwidth', 'carheight', 'curbweight', 'enginelocation', 'enginesize', 'boreratio', 'stroke', 'compressionratio', 'horsepower', 'peakrpm'. Team Product có thể dựa vào các yếu tố đó và các phân tích ở phần 3 để đưa ra các phương án thiết kế của ô tô, chiến lược kinh doanh,... đáp ứng với từng phân khúc giá nhất định. Sau đó, có thể dùng mô hình Random Forest (có xử lý outlier và lựa chọn biến) để dự báo giá bán phù hợp.