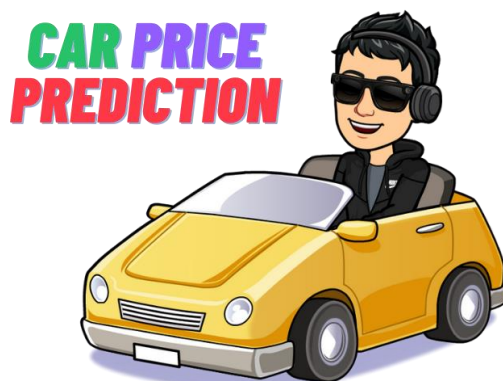




TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO CUỐI KỲ  
HỌC PHẦN: KHOA HỌC DỮ LIỆU



**DỰ ĐOÁN GIÁ XE HƠI**

**GVHD: Phạm Công Thắng**

HỌ VÀ TÊN SINH VIÊN	LỚP HỌC PHẦN	MÃ SINH VIÊN
Huỳnh Đình Hoàng Viên	20Nh13	102200160
Đặng Văn Nhật Minh	20Nh13	102200139
Trần Công Nguyên Hải	20Nh13	102200127
Phạm Văn Nhật Nam	20Nh13	102200140

## TÓM TẮT

Bộ dữ liệu bao gồm các thông tin, thông số kỹ thuật tổng quát, giá bán của xe hơi, được thu thập từ trang web bán các xe hơi ở Việt Nam. Bộ dữ liệu này được dùng để dự đoán giá xe hơi với các thông số kỹ thuật, mẫu mã được đưa ra. Dữ liệu sau khi thu thập là các chuỗi ký tự trong đó có các đại lượng cần dùng cho mô hình dự đoán. Khi làm sạch dữ liệu, cần trả về đúng kiểu dữ liệu, thay thế các giá trị dữ liệu trống và chọn lựa ra các đặc trưng phù hợp cho mô hình dự đoán. Mô hình dự đoán được xây dựng bởi thuật toán Random Forest Regression và cải tiến độ chính xác bằng mô hình XGBoost Regression. Đánh giá hiệu quả của mô hình dự đoán sử dụng RMSE, MAE và R2.

## BẢNG PHÂN CÔNG NHIỆM VỤ

Sinh viên thực hiện	Các nhiệm vụ	Tự đánh giá theo 3 mức
Huỳnh Đình Hoàng Viên	- Thu thập dữ liệu - Làm sạch dữ liệu	- Đã hoàn thành - Đã hoàn thành
Đặng Văn Nhật Minh	- Làm sạch dữ liệu, xử lý dữ liệu trống - Chuẩn hóa dữ liệu	- Đã hoàn thành - Đã hoàn thành
Phạm Văn Nhật Nam	- Trực quan hóa dữ liệu - Mô tả dữ liệu	- Đã hoàn thành - Đã hoàn thành
Trần Công Nguyên Hải	- Xây dựng mô hình - Đánh giá thuật toán	- Đã hoàn thành - Đã hoàn thành

# MỤC LỤC

## Mục lục

<b>1. Giới thiệu .....</b>	<b>6</b>
<b>2. Thu thập và mô tả dữ liệu .....</b>	<b>6</b>
2.1. Thu thập dữ liệu.....	6
2.1.1 Nguồn dữ liệu .....	6
2.1.2 Công cụ thu thập.....	6
2.1.3 Cách thức sử dụng công cụ.....	7
2.1.4 Format & làm sạch dữ liệu .....	11
2.2. Mô tả dữ liệu.....	12
<b>3. Trích xuất đặc trưng.....</b>	<b>13</b>
3.1. Loại bỏ các cột dữ liệu không cần thiết, thêm cột dùng cho dự đoán.....	13
3.2. Làm sạch dữ liệu với đúng kiểu dữ liệu.....	14
3.3. Xử lý dữ liệu trống.....	15
3.4. Chuyển các dữ liệu phân loại thành dữ liệu dạng số (Encode Categorical) .....	15
3.5. Chia dữ liệu thành tập input X, nhãn Y .....	15
3.6. Chuẩn hóa dữ liệu.....	16
3.7. Tương quan giữa các cột thuộc tính của dữ liệu .....	17
<b>4. Mô hình hóa dữ liệu.....</b>	<b>18</b>
4.1. Mô hình Random Forest Regression .....	18
4.2. Mô hình Hồi quy của thư viện XGBoost - XGBoost Regression .....	21
4.3. So sánh hiệu quả của các mô hình .....	22
<b>5. Dự đoán và kết luận.....</b>	<b>23</b>
5.1. Dự đoán giá một chiếc xe hơi bất kì.....	23
5.2. Mô hình hóa dữ liệu.....	25
<b>6. Tài liệu tham khảo .....</b>	<b>25</b>

## DANH SÁCH HÌNH ẢNH

Hình 1. Mô hình Web Scrapping sử dụng BeautifulSoup .....	6
Hình 2. Đường dẫn tới các trang chứa danh sách xe ô tô (300 trang) .....	7
Hình 3. Dạng sách sản phẩm theo từng trang .....	7
Hình 4. Phân tích cấu trúc của website .....	8
Hình 5. Tìm đường link chứa url dẫn tới trang chi tiết.....	8
Hình 6. Lấy dữ liệu thông tin chi tiết của sản phẩm .....	9
Hình 7. Lấy các class .....	9
Hình 8. Trích xuất thông tin lưu vào file .csv .....	10
Hình 9. Source code .....	10
Hình 10. Format dữ liệu động cơ .....	11
Hình 11. Format thông tin tên hãng .....	11
Hình 12. Format tên xe .....	11
Hình 13. Format đơn vị tiền.....	11
Hình 14. Bảng mô tả các cột dữ liệu.....	12
Hình 15. Thông tin về các cột giá trị trong dữ liệu.....	12
Hình 16. Bảng phân bố số lượng xe theo giá tiền.....	13
Hình 17. Dữ liệu sau khi thêm bớt các cột.....	13
Hình 18. Chuyển kiểu dữ liệu của cột “Giá tiền” .....	14
Hình 19. Dữ liệu sau khi làm sạch.....	14
Hình 20. Dữ liệu sau khi chuyển các cột Object thành số bằng LabelEncoder của thư viện sk-learn.....	15
Hình 21. Khởi tạo tập dữ liệu X và nhãn Y .....	16
Hình 22. Chia dữ liệu thành X_train, Y_train, X_test, Y_test .....	16
Hình 23. Chuẩn hóa dữ liệu tập X_train .....	17
Hình 24. Ma trận tương quan giữa các thuộc tính nhận dạng.....	18
Hình 25. Sử dụng thuật toán RandomForest để xây dựng mô hình dự đoán giá.....	19
Hình 26. Tương quan giữa tập test và predict khi sử dụng thuật toán RandomForest .....	19
Hình 27. Khởi tạo bộ tham số cho mô hình XGBoost.....	21
Hình 28. Tương quan giữa tập test và predict khi sử dụng mô hình XGBoost .....	22
Hình 29. Load file.pkl .....	23
Hình 30. Tạo một đối tượng xe hơi mới .....	23
Hình 31. Sử dụng lớp OrdinalEncoder .....	24
Hình 32. Sử dụng lớp StandardScaler.....	24
Hình 33. Hàm dự đoán giá.....	24
Hình 34. Kết quả dự đoán .....	24

# NỘI DUNG BÁO CÁO

## 1. Giới thiệu

Mô hình dự đoán giá xe hơi dựa trên các thông số kỹ thuật tổng quát, đặc trưng của xe hơi sẽ giúp các hãng xe ước tính được giá bán xe hơi phù hợp để cạnh tranh với các hãng khác. Ngoài ra, mô hình này cũng giúp cho khách hàng xác định được giá tốt nhất cho xe hơi họ muốn mua với các thông số cụ thể.

## 2. Thu thập và mô tả dữ liệu

### 2.1. Thu thập dữ liệu

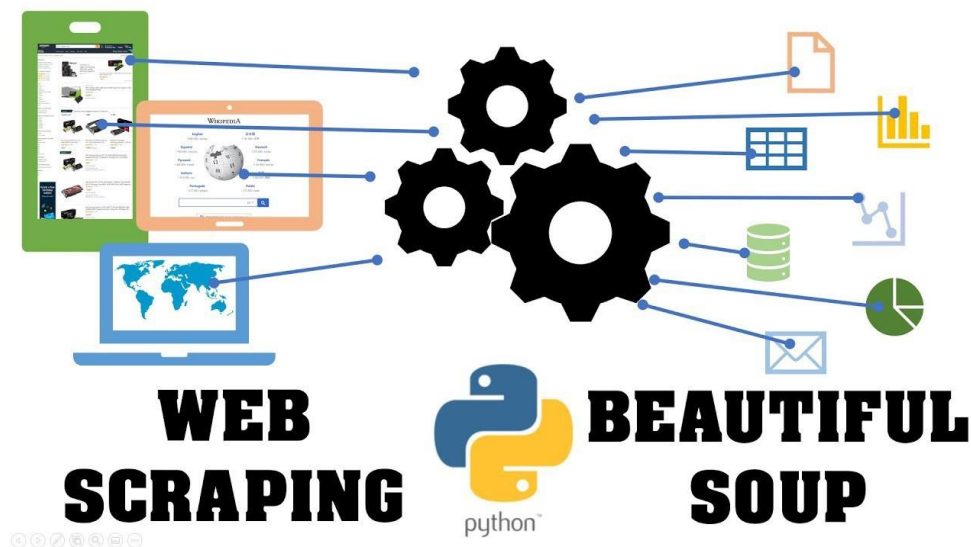
#### 2.1.1 Nguồn dữ liệu

Dữ liệu được thu thập từ trang web bán xe hơi lớn ở Việt Nam, các xe được lựa chọn có mức giá  $< 1$  tỉ 600 triệu:

<https://bonbanh.com/oto-xe-moi-gia-duoi-1600-trieu/page>

#### 2.1.2 Công cụ thu thập

Thu thập dữ liệu từ web sử dụng thư viện **Beautiful Soup**. Đây là một thư viện Python dùng để chuyển đổi HTML và XML. Nó tạo ra một cây phân tích của trang web dùng để trích xuất dữ liệu từ HTML giúp ích cho việc cào dữ liệu.



Hình 1. Mô hình Web Scrapping sử dụng BeautifulSoup

### 2.1.3 Cách thức sử dụng công cụ






- Duyệt qua 300 trang danh sách các sản phẩm
- Lặp qua từng dữ liệu sản phẩm để lấy các thông tin tổng quan
- Lấy ra link chi tiết sản phẩm -> Tiến hành phân tích trang sản phẩm chi tiết để lấy các dữ liệu cần thiết
- Format làm sạch dữ liệu
- Kết quả: File chứa dữ liệu car-data.csv

#### Bước 1: Lấy ra tất cả link của các sản phẩm

```
req = requests.get('https://bonbanh.com/oto-xe-moi-gia-duoi-1600-trieu/page,{}'.format(page),  
content = req.content
```

Hình 2. Đường dẫn tới các trang chứa danh sách xe ô tô (300 trang)

Sau khi kiểm tra thấy được trang web gọi đến một api.

BÁN XE MỚI Ô TÔ DƯỚI 1 TỶ 600 TRIỆU			
Trang 1 / 321 ( Tổng: 6,414 tin )		Sắp xếp theo: Không sắp xếp	
Xe mới 2023	<b>Honda Civic RS 1.5 AT - 2023</b> Ban xe oto Honda Civic 2023 RS 1.5 AT gia 875 Triệu - TP HCM	875 Triệu	TP HCM
	Honda Ô Tô Sài Gòn - Bình Chánh bán xe Honda Civic RS cao cấp 2023 phiên bản mới. - Xe sẵn kho đủ màu giao ngay. - Lái thử trải nghiệm xe tận nhà. - Cam kết không kèm ...	Liên hệ: Honda Ô Tô Sài Gòn - Bình Chánh Số 4 QL1A, TT. Tân Túc, Quận Bình Chánh TP HCM ĐT: 0909 168 511	
Mã: 4771894			
Xe mới 2023	<b>Toyota Camry 2.0Q - 2023</b> *Xe nhập khẩu, màu trắng, máy xăng 2.0 L, số tự động ...	1 Tỷ 185 Triệu	Hải Phòng
	Toyota Camry 2.0Q phiên bản mới Xe trang bị: -Hệ thống An toàn cao cấp -Lấy số vô lăng -Sạc không dây -Cảnh báo điểm mù -Hệ thống cảnh báo phương tiện cắt ngang -Cửa ...	Liên hệ: Toyota Nankai Hải Phòng Km 88, Lương Quán, P. Nam Sơn, Q. An Dương Hải Phòng ĐT: 0936 655 966	
Mã: 4857637			
Xe mới 2023	<b>Mitsubishi Xpander Premium 1.5 AT - 2023</b> *Xe nhập khẩu, màu đen, máy xăng 1.5 L, số tự động ...	648 Triệu	Hà Nội
	Xpander AT Premium 2023 phiên bản mới - Hỗ trợ trả góp lãi suất thấp, thủ tục nhanh gọn - Liên hệ ngay để biết thêm thông tin *** - Đường nét thiết kế khỏe khoắn kết ...	Liên hệ: Mitsubishi Việt Hùng 936 Quang Trung, Phường Yên Nghĩa, Quận Hà Đông Hà Nội ĐT: 0937 779 998	
Mã: 4438925			
Xe mới 2022	<b>Suzuki XL7 1.5 AT - 2022</b> *Xe nhập khẩu, màu đen, máy xăng 1.5 L, số tự động ...	599 Triệu	Hà Nội
	Bán xe Suzuki XL 7. Xe được trang bị : Đèn pha nổi liền vào hai bên lưới tản nhiệt, tích hợp dải đèn LED chiếu sáng ban ngày. Cụm đèn hậu cũng dùng bóng LED. Thiết kế mắ ...	Liên hệ: Suzuki Việt Anh Km14+600 Quốc lộ 6, Hà Đông Hà Nội ĐT: 0972 251 486	
Mã: 3201559			
Xe mới 2023	<b>Kia Carnival Premium 2.2D - 2023</b> *Xe lắp ráp trong nước, màu đen, máy dầu 2.2 L, số tự động ...	1 Tỷ 369 Triệu	Hà Nội
	Kia Carnival 2.2D Premium. Xe trang bị: Cụm đèn hậu Led 53, 2 màn hình 12,3 inch, ghế lái và ghế phụ đều và tích hợp sưởi làm mát cần số dạng nút	Liên hệ: Kia Đồng Đa 568 Đường Láng, Đống Đa Hà Nội ĐT: 0962 381 186	
Mã: a-civic-rs-1.5-at-2023-4771894			

Hình 3. Dạng sách sản phẩm theo từng trang

## Bước 2: Phân tích cấu trúc trang web của từng sản phẩm

The screenshot displays a car listing website. At the top, two car models are listed: Toyota Camry 2.0Q - 2023 (Mã: 4771894) and Mitsubishi Xpander Premium 1.5 AT - 2023 (Mã: 4857637). The Toyota Camry listing includes a photo, a description, and a price of 1 Tỷ 185 Triệu. The Mitsubishi Xpander listing includes a price of 648 Triệu. Below the listings, a GraphQL Network tab is open, showing the JSON structure of the page. The structure includes a 'login-dlg' element, a 'breadcrum' element, a 'mainc2' element, a 'filter-box' element, a 'city-box' element, a 'loading\_data' element, and a 'search\_content' element. The 'search\_content' element contains a 'gray-box' element with a 's-list-car' element. The 's-list-car' element contains a 'g-box-content' element with a 'g-b-header' element, a 'paggging' element, and a 'ul' element. The 'ul' element contains two 'li' elements, each representing a car item. The first 'li' element contains a 'car-item row1' element with a 'url' field pointing to 'xe-toyota-camry-2.0q-2023-4857637' and a 'price' field with a value of '1185000000'.

Hình 4. Phân tích cấu trúc của website

Tìm chứa thông tin về thông số kỹ thuật cơ bản của xe

Tìm kiếm class và id của các thẻ

Sau khi đã lấy được các thông số kỹ thuật. Tìm thẻ chứa link chuyển sang trang chi tiết sản phẩm.

The screenshot shows the GraphQL Network tab with the JSON structure of the page. The 'url' field is highlighted, showing the value 'xe-toyota-camry-2.0q-2023-4857637'. The 'price' field is also highlighted, showing the value '1185000000'.

Hình 5. Tìm đường link chứa url dẫn tới trang chi tiết

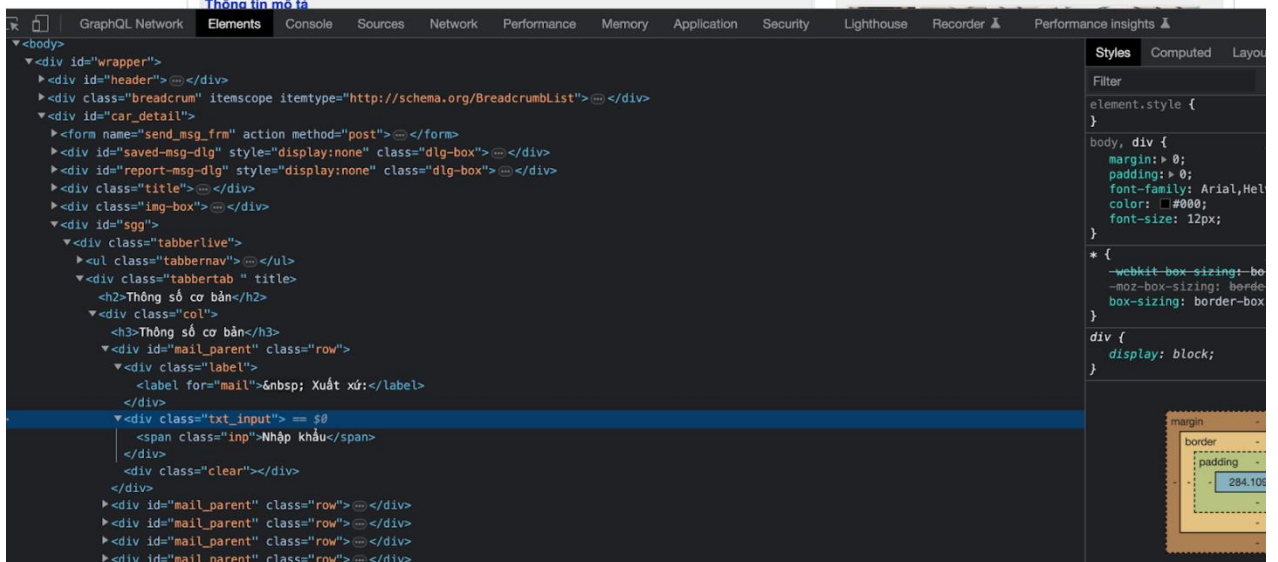


### Lấy dữ liệu các thông tin chi tiết của sản phẩm

```
p_url = product.find('a').get('href')
detailCar = requests.get('https://bonbanh.com/{}'.format(str(p_url)), headers=headers)
detailSoup = BeautifulSoup(detailCar.content, 'html.parser')
details = detailSoup.find_all('div', class_='row')
p_seat = detailSoup.find('div', class_='row_last').find('span', class_='inp').text.strip()
seatQuantity.append(p_seat.replace('chỗ', ''))
for detail in details:
    p_label = detail.find('label').text.strip()
    p_data = detail.find('span', class_='inp').text.strip()
    if('Dòng xe' in p_label):
        types.append(p_data)
    if('Động cơ' in p_label):
        [t_material, t_capacity] = formatEngine(p_data)
        material.append(t_material)
        capacity.append(t_capacity)
    if('Xuất xứ' in p_label):
        origin.append(p_data)
    if('Hộp số' in p_label):
        gears.append(p_data)
```

Hình 6. Lấy dữ liệu thông tin chi tiết của sản phẩm

Thông số cơ bản	
Xuất xứ:	Nhập khẩu
Tình trạng:	Xe mới
Dòng xe:	Sedan
Số Km đã đi:	0 Km
Màu ngoại thất:	Trắng
Màu nội thất:	Vàng
Số cửa:	4 cửa
Số chỗ ngồi:	5 chỗ
Chi tiết khác	
Hãng xe:	Ford
Model:	Mustang
Loại xe:	Coupe
Động cơ:	2.0L EcoBoost
Truyền động:	6 cấp tự động
Giá niêm yết:	789 triệu đồng
Giá thực tế:	750 triệu đồng
Ưu đãi:	Giảm 39 triệu đồng
Chế độ bảo hành:	3 năm hoặc 100.000 km
Thời gian giao hàng:	Trong vòng 1 tuần



### Hình 7. Lấy các class

### Bước 3: Trích xuất thông tin và lưu vào file csv

```

def save(name, prices, year, types, seatQuantity, material, capacity, origin, gears, company):
    data = {
        'Hãng xe': company,
        'Tên xe': name,
        'Giá (VNĐ)': prices,
        'Năm sản xuất': year,
        'Dòng xe': types,
        'Số chỗ ngồi': seatQuantity,
        'Nguyên liệu': material,
        'Dung tích (L)': capacity,
        'Xuất xứ': origin,
        'Hộp số': gears,
    }
    df = pd.DataFrame.from_dict(data, orient='index')

    df = df.transpose()
    dataset = pd.DataFrame(data=df)
    dataset.to_csv('car-data.csv')
    dataset.to_excel('data.xlsx', index=False)

    csv_reader = csv.reader('car-data.csv', delimiter=",")

if name == 'main':

```

Hình 8. Trích xuất thông tin lưu vào file .csv

## Source code:

```

def crawl(page):

    req = requests.get('https://bonbanh.com/oto-xe-moi-gia-duoi-1600-trieu/page,{0}'.format(page), headers=headers)
    content = req.content
    soup = BeautifulSoup(content, 'html.parser')
    products = soup.find_all('li', class_='car-item')
    for product in products:
        p_title = product.find('h3').text.strip()
        p_year = product.find('div', class_='cb1').find('b').text.strip()
        p_price = product.find('div', class_='cb3').find('b').text.strip()
        year.append(p_year)
        prices.append(formatCurrency(p_price))
        name.append(formatCarName(p_title))
        company.append(formatCompany(p_title))
        p_url = product.find('a').get('href')
        detailCar = requests.get('https://bonbanh.com/{0}'.format(str(p_url)), headers=headers)
        detailSoup = BeautifulSoup(detailCar.content, 'html.parser')
        details = detailSoup.find_all('div', class_='row')
        p_seat = detailSoup.find('div', class_='row_last').find('span', class_='inp').text.strip()
        seatQuantity.append(p_seat.replace('chỗ', ''))
        for detail in details:
            p_label = detail.find('label').text.strip()
            p_data = detail.find('span', class_='inp').text.strip()
            if('Dòng xe' in p_label):
                types.append(p_data)
            if('Động cơ' in p_label):
                [t_material, t_capacity] = formatEngine(p_data)
                material.append(t_material)
                capacity.append(t_capacity)
            if('Xuất xứ' in p_label):
                origin.append(p_data)
            if('Hộp số' in p_label):
                gears.append(p_data)

```

Hình 9. Source code

### 2.1.4 Format & làm sạch dữ liệu

- Format thông tin động cơ: ví dụ Xăng 2.0L -> Xăng vào 2.0

```
def formatEngine(string):  
    words = string.split()  
    if(len(words) == 1):  
        material = string  
        capacity = 0  
    else:  
        material = words[0]  
        capacity = words[1]  
    return material, capacity
```

Hình 10. Format dữ liệu động cơ

- Format thông tin tên hãng:

```
def formatCompany(data):  
    return data.split(' ')[0]
```

Hình 11. Format thông tin tên hãng

- Format tên xe:

```
def formatCarName(data):  
    return data.replace(formatCompany(data), '')
```

Hình 12. Format tên xe

- Format đơn vị tiền:

```
def formatCurrency(number):  
    number = number.replace(" ", "")  
    if "Tỷ" in number and "Triệu" in number:  
        billion = int(number.split("Tỷ")[0])  
        million = int(number.split("Tỷ")[1].replace("Triệu", ""))  
        value = billion * 1000000000 + million * 1000000  
    elif "Tỷ" in number:  
        billion = int(number.replace("Tỷ", ""))  
        value = billion * 1000000000  
    else:  
        million = int(number.replace("Triệu", ""))  
        value = million * 1000000  
    return '{:,}'.format(value)
```

Hình 13. Format đơn vị tiền

## 2.2. Mô tả dữ liệu

Dữ liệu được crawl từ website bao gồm 6000 hàng và 10 cột bao gồm:

- Hãng xe: hãng sản xuất (ví dụ: Ford, Toyota, Hyundai,...)
- Tên xe: tên đầy đủ xe (ví dụ: Ford Everest Titanium 2.0L 4x2AT,..)
- Năm sản xuất: năm sản xuất của xe
- Dòng xe (ví dụ: SUV, Crossover,...)
- Số chỗ ngồi
- Nhiên liệu
- Dung tích xi lanh
- Xuất xứ
- Hộp số

	brand	name	year	model	seat	fuel	capacity	origin	gear	price
0	Ford	Ford Everest Titanium 2.0L 4x2 AT - 2023	2023	SUV	7	Dầu	2.0	Nhập khẩu	Số tự động	1276000000
1	Peugeot	Peugeot 3008 GT - 2022	2022	Crossover	5	Xăng	1.6	Lắp ráp trong nước	Số tự động	1209000000
2	Kia	Kia K5 Luxury 2.0 AT - 2023	2023	Sedan	5	Xăng	2.0	Lắp ráp trong nước	Số tự động	884000000
3	Ford	Ford Ranger Wildtrak 2.0L 4x4 AT - 2023	2023	Bán tải / Pickup	5	Dầu	2.0	Lắp ráp trong nước	Số tự động	924000000
4	Mazda	Mazda cx3 Luxury 1.5 AT - 2023	2023	Crossover	5	Xăng	1.5	Nhập khẩu	Số tự động	620000000

Hình 14. Bảng mô tả các cột dữ liệu

### Đánh giá về số các ô giá trị trống trong dữ liệu:

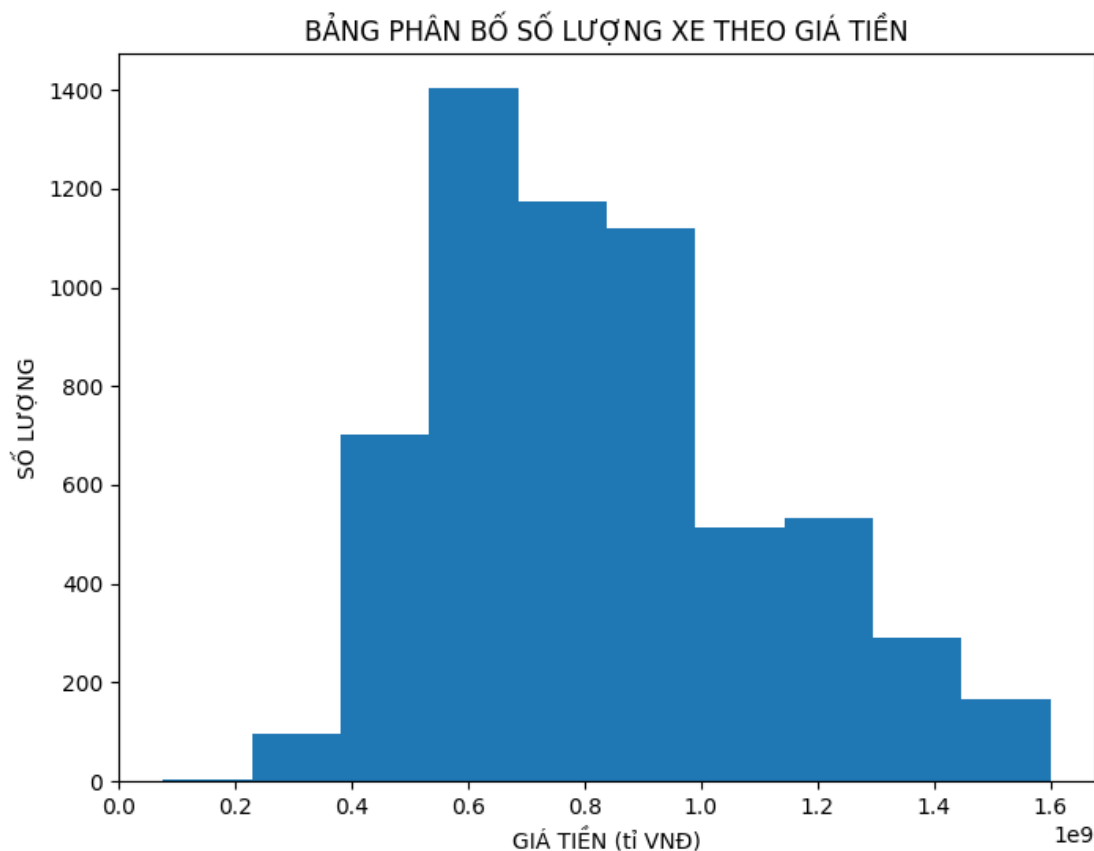
- Các cột dữ liệu đều có đủ dữ liệu (ở cột non-null thì có đủ 6000 giá trị)  
=> crawl dữ liệu rất tốt
- Có nhiều cột dữ liệu kiểu Object, điều đó gây khó khăn ở việc chuẩn hóa dữ liệu

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6000 entries, 0 to 5999
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   Unnamed: 0            6000 non-null   int64  
 1   Hãng xe               6000 non-null   object  
 2   Tên xe                6000 non-null   object  
 3   Giá (VNĐ)            6000 non-null   object  
 4   Năm sản xuất          6000 non-null   int64  
 5   Dòng xe               6000 non-null   object  
 6   Số chỗ ngồi           6000 non-null   int64  
 7   Nguyên liệu           6000 non-null   object  
 8   Dung tích (L)         6000 non-null   float64  
 9   Xuất xứ               6000 non-null   object  
10   Hộp số                6000 non-null   object  
dtypes: float64(1), int64(3), object(7)
memory usage: 515.8+ KB
```

Hình 15. Thông tin về các cột giá trị trong dữ liệu

## Phân bố dữ liệu của cột dự đoán “price”



Hình 16. Bảng phân bố số lượng xe theo giá tiền

Đa phần giá xe tập trung ở phân khúc giá từ 400 triệu - 1 tỉ 200 triệu. Có một số giá trị khá nhỏ (<200 triệu) nhưng không đáng kể.

### 3. Trích xuất đặc trưng

#### 3.1. Loại bỏ các cột dữ liệu không cần thiết, thêm cột dùng cho dự đoán

Toàn bộ dữ liệu thô bao gồm tổng cộng **6000 hàng** với **10 cột**. Trong đó, có các cột không cần thiết cho mô hình dự đoán giá, ta sẽ loại bỏ các cột này. Đó là cột “Unnamed: 0” – đây là cột được tạo ra tự động khi chúng ta load file “.csv”.

Do các thông số tổng quát của xe hơi đều khá tương đồng với mỗi chiếc xe, nhưng cùng một thông số thì giá tiền giữa các hãng xe lại có sự chênh lệch. Do đó chúng ta tách từ cột “Tên xe” thành một cột mới là “Hãng xe”.

Cuối cùng ta đổi tên các cột dữ liệu để thuận tiện trong quá trình chuẩn hóa dữ liệu.

	brand		name	year	model	seat	fuel	capacity	origin	gear	price
0	Ford	Ford Everest Titanium 2.0L 4x2 AT - 2023		2023	SUV	7	Dầu	2.0	Nhập khẩu	Số tự động	1276000000
1	Peugeot	Peugeot 3008 GT - 2022		2022	Crossover	5	Xăng	1.6	Lắp ráp trong nước	Số tự động	1209000000
2	Kia	Kia K5 Luxury 2.0 AT - 2023		2023	Sedan	5	Xăng	2.0	Lắp ráp trong nước	Số tự động	884000000
3	Ford	Ford Ranger Wildtrak 2.0L 4x4 AT - 2023		2023	Bán tải / Pickup	5	Dầu	2.0	Lắp ráp trong nước	Số tự động	924000000
4	Mazda	Mazda cx3 Luxury 1.5 AT - 2023		2023	Crossover	5	Xăng	1.5	Nhập khẩu	Số tự động	620000000

Hình 17. Dữ liệu sau khi thêm bớt các cột

### 3.2. Làm sạch dữ liệu với đúng kiểu dữ liệu

Bộ dữ liệu bao gồm 10 cột với 7 dữ liệu dạng đối tượng và 3 dữ liệu dạng số.

#### a) Dữ liệu đối tượng

Các dữ liệu dạng đối tượng:

- Tên xe
- Dòng xe
- Nhiên liệu
- Xuất xứ
- Hộp số
- Giá xe

Vì các thông số bao quát của xe chưa phản ánh hết giá của xe nên chúng ta tách từ cột “Tên xe” thêm một cột “Hãng xe”.

Cột “Dung tích” ta loại bỏ ký tự đơn vị “L”.

Với cột “Giá xe” định dạng ban đầu ở web là “X tỉ Y triệu”, do đó trong quá trình clean chúng ta cần xử lý bằng cách viết hàm để format thành dạng XXX,XXX,XXX,XXX.

#### b) Dữ liệu dạng số

Đối với các dữ liệu dạng số, sau khi xử lý cột “Giá tiền” ở phần crawl thì chúng ta nhận được cột với kiểu dữ liệu Object, để thuận tiện cho quá trình xử lý khi training data thì chúng ta cần chuyển nó sang kiểu số.

```
[ ] df.price.dtype #kiểu hiện tại của price là Object
dtype('O')

[ ] df.price = df.price.apply(lambda x : int(x.replace(',',' '))) #chuyển kiểu dữ liệu của price sang int

[ ] df.price.dtype
dtype('int64')
```

Hình 18. Chuyển kiểu dữ liệu của cột “Giá tiền”

Dữ liệu sau khi làm sạch:

	brand	name	year	model	seat	fuel	capacity	origin	gear	price
0	Ford	Ford Everest Titanium 2.0L 4x2 AT - 2023	2023	SUV	7	Dầu	2.0	Nhập khẩu	Số tự động	1276000000
1	Peugeot	Peugeot 3008 GT - 2022	2022	Crossover	5	Xăng	1.6	Lắp ráp trong nước	Số tự động	1209000000
2	Kia	Kia K5 Luxury 2.0 AT - 2023	2023	Sedan	5	Xăng	2.0	Lắp ráp trong nước	Số tự động	884000000
3	Ford	Ford Ranger Wildtrak 2.0L 4x4 AT - 2023	2023	Bán tải / Pickup	5	Dầu	2.0	Lắp ráp trong nước	Số tự động	924000000
4	Mazda	Mazda cx3 Luxury 1.5 AT - 2023	2023	Crossover	5	Xăng	1.5	Nhập khẩu	Số tự động	620000000

Hình 19. Dữ liệu sau khi làm sạch

### 3.3. Xử lý dữ liệu trống

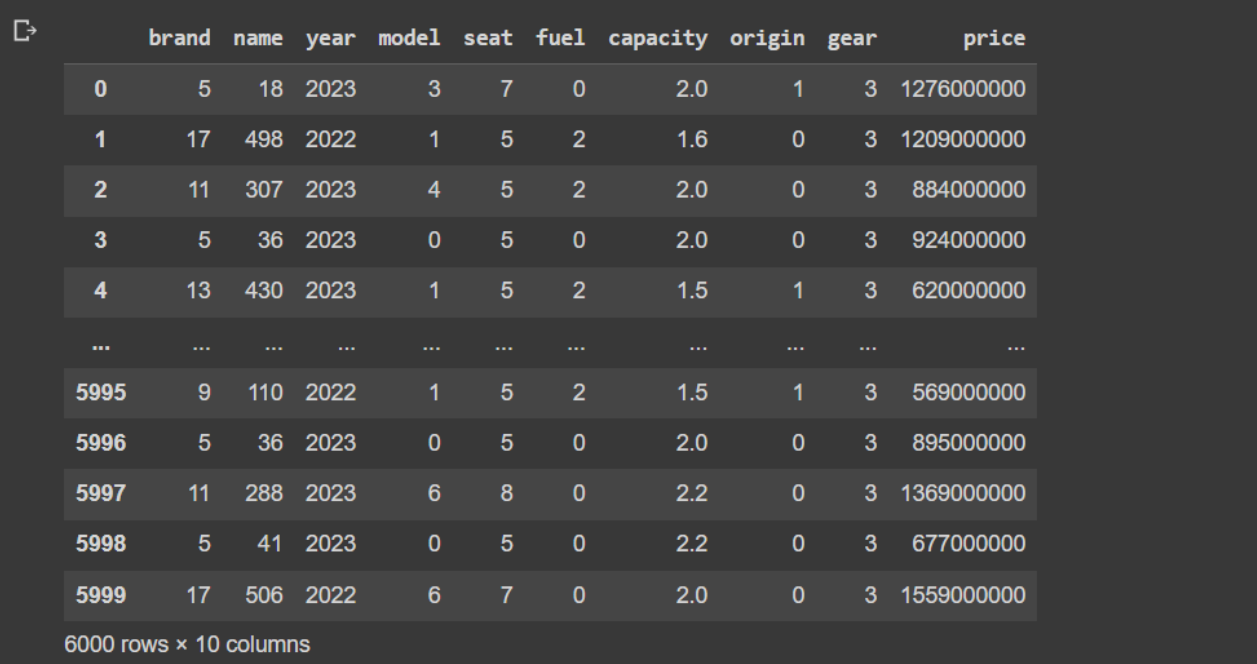
Như đã trình bày ở phần 2.2, do dữ liệu khi crawl về không có giá trị trống nên chúng ta cũng sẽ bỏ qua bước xử lý này.

### 3.4. Chuyển các dữ liệu phân loại thành dữ liệu dạng số (Encode Categorical)

Sử dụng *LabelEncoder* của thư viện *sklearn* để chuyển các kiểu dữ liệu Object thành dữ liệu số cho mô hình máy học có thể hiểu.

Chuyển các cột dữ liệu 'brand', 'name', 'model', 'fuel', 'origin', 'gear' thành dạng số.

Dữ liệu sau khi chuyển thành dạng số:



	brand	name	year	model	seat	fuel	capacity	origin	gear	price
0	5	18	2023	3	7	0	2.0	1	3	1276000000
1	17	498	2022	1	5	2	1.6	0	3	1209000000
2	11	307	2023	4	5	2	2.0	0	3	884000000
3	5	36	2023	0	5	0	2.0	0	3	924000000
4	13	430	2023	1	5	2	1.5	1	3	620000000
...	...	...	...	...	...	...	...	...	...	...
5995	9	110	2022	1	5	2	1.5	1	3	569000000
5996	5	36	2023	0	5	0	2.0	0	3	895000000
5997	11	288	2023	6	8	0	2.2	0	3	1369000000
5998	5	41	2023	0	5	0	2.2	0	3	677000000
5999	17	506	2022	6	7	0	2.0	0	3	1559000000

6000 rows x 10 columns

Hình 20. Dữ liệu sau khi chuyển các cột Object thành số bằng *LabelEncoder* của thư viện *sk-learn*

### 3.5. Chia dữ liệu thành tập input X, nhãn Y

Khi thực hiện xử lý với mô hình máy học, thông thường chúng ta cần chuyển các dữ liệu sau khi đã Label – Encode thành dạng numpy-array

Chúng ta lấy các cột đầu tiên làm tập input X và tập dữ liệu giá làm output Y



```
▶ X = df.iloc[:, :-1].values
X

array([[5.000e+00, 1.800e+01, 2.023e+03, ..., 2.000e+00, 1.000e+00,
        3.000e+00],
       [1.700e+01, 4.980e+02, 2.022e+03, ..., 1.600e+00, 0.000e+00,
        3.000e+00],
       [1.100e+01, 3.070e+02, 2.023e+03, ..., 2.000e+00, 0.000e+00,
        3.000e+00],
       ...,
       [1.100e+01, 2.880e+02, 2.023e+03, ..., 2.200e+00, 0.000e+00,
        3.000e+00],
       [5.000e+00, 4.100e+01, 2.023e+03, ..., 2.200e+00, 0.000e+00,
        3.000e+00],
       [1.700e+01, 5.060e+02, 2.022e+03, ..., 2.000e+00, 0.000e+00,
        3.000e+00]])

[ ] Y = df.iloc[:, -1].values
Y

array([1276000000, 1209000000, 884000000, ..., 1369000000, 677000000,
       1559000000])
```

Hình 21. Khởi tạo tập dữ liệu X và nhãn Y

Sau khi có dữ liệu X và nhãn Y, chúng ta tiếp tục chia 2 tập dữ liệu thành dữ liệu train và test để đưa vào mô hình học máy. Sở dĩ chúng ta cần chia thành tập train và test để tránh tình trạng overfitting model và giúp chúng ta xây dựng được các tham số dùng cho đánh giá mô hình.

Thực hiện chia X và Y thành tập train và test với tỉ lệ 4:1 (80% train và 20% test)

```
▼ Split dataset to X_train Y_train, X_test Y_test

[ ] from sklearn.model_selection import train_test_split

    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=1)
```

Hình 22. Chia dữ liệu thành X\_train, Y\_train, X\_test, Y\_test

### 3.6. Chuẩn hóa dữ liệu

Tiếp theo trong quá trình chuẩn bị dữ liệu để đưa vào mô hình máy học là công đoạn chuẩn hóa dữ liệu. Quan sát các giá trị số sau khi Label-Encode, ta dễ dàng nhận thấy sự chênh lệch các giá trị giữa các cột giá trị, điều này dễ dẫn đến sai khác trong quá trình train data, làm giảm hiệu suất của mô hình, do đó ta cần chuẩn hóa các giá trị này về cùng một miền dữ liệu.

Sử dụng lớp StandardScaler của lớp scikit – learn để đưa các giá trị về phân phối chuẩn, phương pháp này được thực hiện theo công thức:

$$X\_scaler = (X - \text{mean}) / \text{standard\_deviation}$$



Trong đó: X là giá trị ban đầu  
Mean là giá trị trung bình của tập dữ liệu  
Standard\_deviation là độ lệch chuẩn của tập giá trị

```
[ ] from sklearn.preprocessing import StandardScaler

    scaler = StandardScaler()

    X_train[:, :3] = scaler.fit_transform(X_train[:, :3])

[ ] X_train

array([[ 1.40421628,  1.18328171,  0.47096624, ...,  1.8      ,
         1.         ,  3.         ],
       [-0.70884352, -0.96915392,  0.47096624, ...,  1.5      ,
         0.         ,  3.         ],
       [-0.56797286, -0.84204158,  0.47096624, ...,  1.5      ,
         1.         ,  3.         ],
       ...,
       [-0.56797286, -0.57086858,  0.47096624, ...,  2.5      ,
         0.         ,  3.         ],
       [-1.13145548, -1.15134827,  0.47096624, ...,  2.2      ,
         0.         ,  3.         ],
       [-1.13145548, -1.09626626,  0.47096624, ...,  1.5      ,
         0.         ,  3.         ]])
```

Hình 23. Chuẩn hóa dữ liệu tập  $X_{train}$

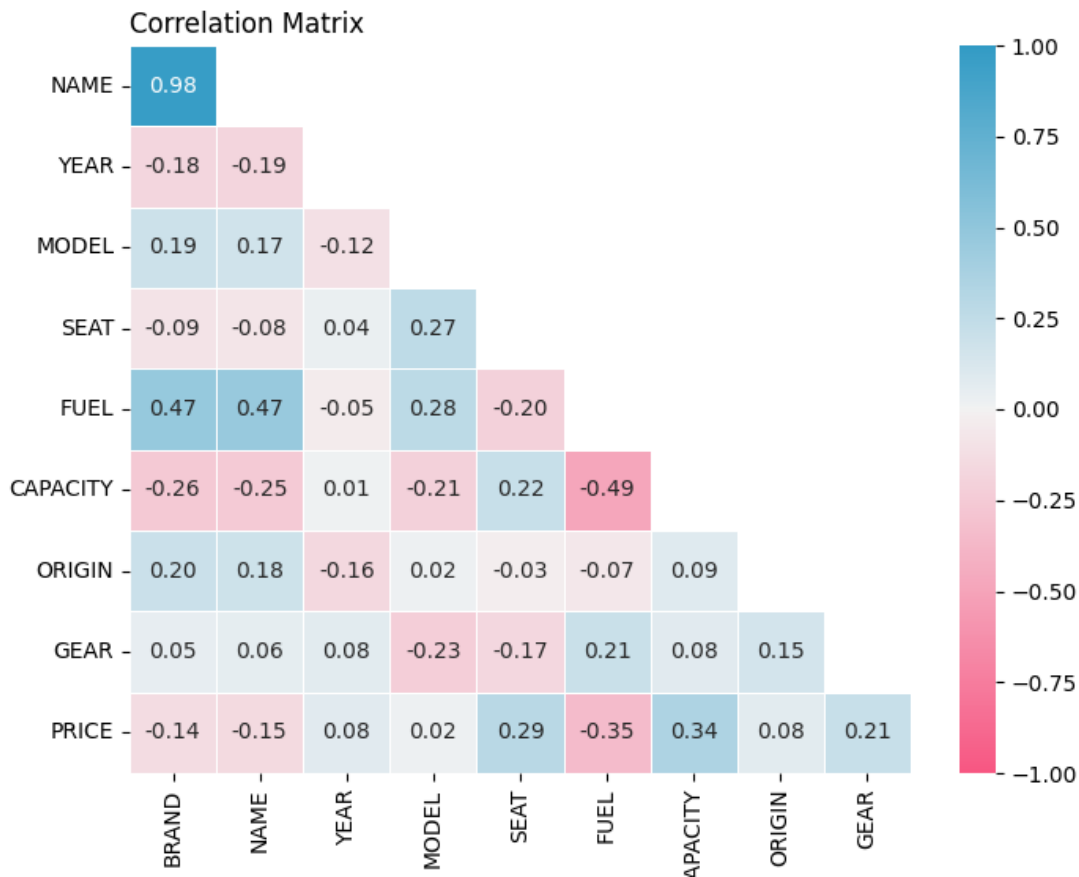
Thực hiện tương tự với tập  $X_{test}$ .

### 3.7. Tương quan giữa các cột thuộc tính của dữ liệu

Việc xem xét sự tương quan giữa các thuộc tính dự đoán giá xe hơi giúp ta có thể chọn lựa các thuộc tính có độ tương quan cao với giá để đưa vào dự đoán.

Ở đây do có khá ít cột dữ liệu (10 cột) và các dữ liệu đa phần đều chỉ mang tính khái quát về một chiếc xe nên không có sự tương quan cao với giá xe.

Trong một số trường hợp, độ tương quan cũng không hoàn toàn thể hiện tính phù hợp của một thuộc tính trong bài toán dự đoán. Trong trường hợp này, mặc dù độ tương quan thấp nhưng chúng ta nhưng có thể mang lại vài thông tin hữu ích, ví dụ như giá xe sẽ phụ thuộc phần lớn vào hãng sản xuất.



Hình 24. Ma trận tương quan giữa các thuộc tính nhận dạng

## 4. Mô hình hóa dữ liệu

Để đáp ứng cho yêu cầu bài toán Dự đoán giá xe hơi, nhóm đã chọn ra 2 mô hình cho để giải quyết bài toán này: Hồi quy tuyến tính (Random Forest Regression) và Hồi quy của thư viện XGBoost (XGBoost Regression).

### 4.1. Mô hình Random Forest Regression

**Cơ sở lý thuyết:** Random Forest là một thuật toán học máy dựa trên việc xây dựng một tập hợp các cây quyết định độc lập. Mỗi cây quyết định được xây dựng dựa trên một tập con dữ liệu ngẫu nhiên và không tương tác với nhau trong quá trình dự đoán. Kết quả dự đoán cuối cùng được tính toán bằng cách kết hợp kết quả của tất cả các cây quyết định. Random Forest có khả năng xử lý dữ liệu rộng, chống overfitting, xử lý dữ liệu thiếu và xếp hạng đặc trưng, là một trong những thuật toán học máy phổ biến và mạnh mẽ.

Trong thuật toán Decision Tree (cây quyết định), khi xây dựng cây quyết định nếu để độ sâu tùy ý thì cây sẽ phân loại đúng hết các dữ liệu trong tập training dẫn đến mô hình có

thể dự đoán tệ trên tập validation/test, khi đó mô hình bị overfitting, hay nói cách khác là mô hình có high variance (thể hiện tốt với dữ liệu có sẵn nhưng kém hiệu quả với dữ liệu mới hoàn toàn).

Thuật toán Random Forest gồm nhiều cây quyết định, mỗi cây quyết định đều có những yếu tố ngẫu nhiên:

- Lấy ngẫu nhiên dữ liệu để xây dựng cây quyết định
- Lấy ngẫu nhiên các thuộc tính để xây dựng cây quyết định

**Bộ tham số của mô hình:** RandomForestRegressor(n\_estimators, random\_state)

Trong đó:

- Estimators: số lượng cây quyết định được xây dựng trong random forest, với estimators càng lớn càng tăng hiệu suất của mô hình, nhưng cũng tăng thời gian huấn luyện
- Random\_state: bằng cách đặt random\_state thành một giá trị cố định, chúng ta đảm bảo rằng mỗi lần chạy mô hình, chúng ta sẽ nhận được kết quả nhất quán.

```
Use RandomForestRegressor

[ ] from sklearn.ensemble import RandomForestRegressor
    from sklearn.metrics import mean_squared_error

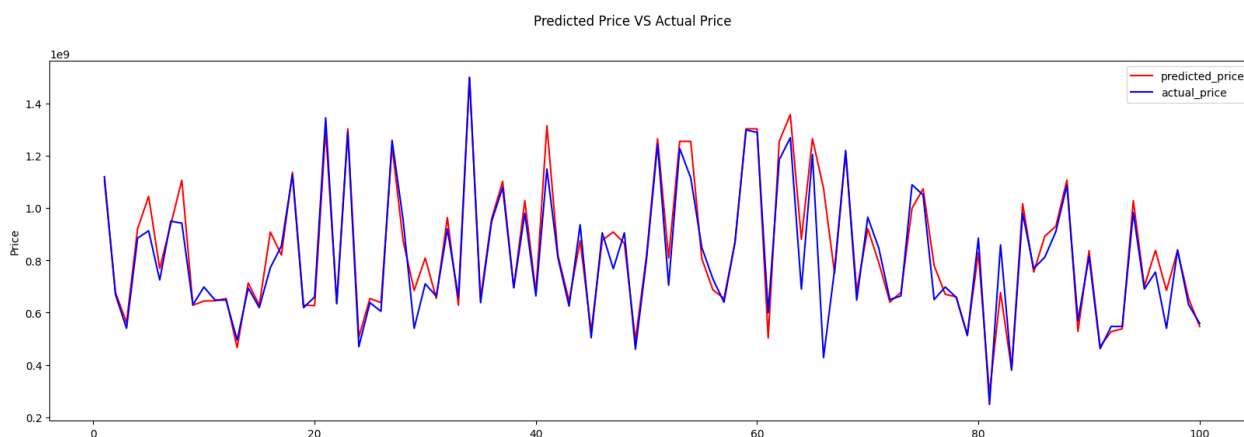
rf = RandomForestRegressor(n_estimators=100, random_state=1)

rf.fit(X_train, Y_train)

RandomForestRegressor
RandomForestRegressor(random_state=1)
```

Hình 25. Sử dụng thuật toán RandomForest để xây dựng mô hình dự đoán giá

**Biểu đồ tương quan giữa tập test và predict:**



Hình 26. Tương quan giữa tập test và predict khi sử dụng thuật toán RandomForest

Chú thích: Biểu đồ biểu diễn kết quả chênh lệch giữa giá dự đoán và giá thực tế của từng mẫu (lấy kết quả của 100 mẫu đầu tiên trong tập kiểm thử), đường màu đỏ là giá dự đoán, đường màu xanh là giá thực tế.

**Kết quả:** Nhìn trên đồ thị ta thấy đường màu đỏ và màu xanh chênh lệch không nhiều, dao động trong khoảng 20 đến 100 triệu. Kết quả dự đoán đánh giá trên hàm score có sẵn của mô hình của mô hình có độ chính xác theo tiêu chuẩn đánh giá  $R^2(\%)$  đạt **85.31%** trên tập dữ liệu kiểm thử.

**Đánh giá:** Độ hiệu quả dự đoán với các metrics, các metrics sử dụng để đánh giá bao gồm: *RMSE*, *MAE* và *R2*

	<b>RMSE</b> (nghìn đồng)	<b>MAE</b> (nghìn đồng)	<b>R2</b> (%)
<b>Kết quả</b>	105,769,763.10	57,789,856.07	85.31

*Chú thích:*

- **RMSE:** Root Mean Square Error là phép đo dùng để đo lường độ lớn sai số dự đoán giữa giá trị dự đoán của mô hình và giá trị thực tế trong dữ liệu.  
RMSE đo lường độ lớn trung bình của sai số dự đoán theo đơn vị của biến phụ thuộc. Điều này có nghĩa là giá trị RMSE càng nhỏ, mô hình càng có khả năng dự đoán chính xác và gần với giá trị thực tế.
- **MAE:** Mean Absolute Error là phép đo dùng để đo lường trung bình giá trị tuyệt đối của sai số dự đoán giữa giá trị dự đoán của mô hình và giá trị thực tế trong dữ liệu.  
MAE đo lường độ lớn trung bình của sai số dự đoán dựa trên đơn vị tuyệt đối của biến phụ thuộc. Giá trị MAE càng nhỏ, mô hình càng có khả năng dự đoán chính xác và gần với giá trị thực tế.  
MAE thường được sử dụng trong các bài toán hồi quy và có ưu điểm là nhạy cảm với các giá trị ngoại lệ (outliers), vì nó không bình phương các sai số như RMSE. Tuy nhiên, MAE không nhận biết được hướng của sai số (dương hay âm), chỉ xem xét độ lớn tuyệt đối của sai số.
- **R2:** R-squared là phép đo dùng để đo lường tỉ lệ phương sai của biến phụ thuộc mà mô hình có thể giải thích được so với tổng phương sai của biến phụ thuộc  
R2 càng gần 1 thì mô hình càng tốt trong việc dự đoán biến phụ thuộc  
R2 càng gần 0 thì mô hình càng không tốt trong việc dự đoán biến phụ thuộc  
Tuy nhiên R2 không đánh giá toàn diện được mọi khía cạnh của mô hình, do đó cần kết hợp với các phép đo khác để thực hiện đánh giá

## 4.2. Mô hình Hồi quy của thư viện XGBoost - XGBoost Regression

**Cơ sở lý thuyết:** XGBoost (Extreme Gradient Boosting) là một giải thuật được base trên gradient boosting, tuy nhiên kèm theo đó là những cải tiến to lớn về mặt tối ưu thuật toán, về sự kết hợp hoàn hảo giữa sức mạnh phần mềm và phần cứng, giúp đạt được những kết quả vượt trội cả về thời gian training cũng như bộ nhớ sử dụng. Có nhiều các mô hình cho việc Hồi quy (Regression), Phân lớp (Classification),...

**Bộ tham số của mô hình:** Sử dụng RandomizedSearchCV để tìm siêu tham số cho mô

Tên tham số	Mảng giá trị	Ý nghĩa
n_estimators	[100, 500, 900, 1100, 1500]	Số estimators trong mô hình tìm kiếm
max_depth	[2, 3, 5, 10, 15]	Độ sâu tìm kiếm tối đa
booster	['gbtree', 'gblinear']	Mô hình để chạy
learning_rate	[0.05, 0.1, 0.15, 0.20]	Tốc độ học
min_child_weight	[1,2,3,4]	Trọng lượng tối thiểu của giá trị con
base_score	[0.25, 0.5, 0.75, 1]	Điểm khởi tạo tìm tham số

```
[ ] import xgboost as xgb
    from sklearn.model_selection import RandomizedSearchCV

[ ] n_estimators = [100, 500, 900, 1100, 1500]
    max_depth = [2, 3, 5, 10, 15]
    booster=['gbtree', 'gblinear']
    learning_rate=[0.05,0.1,0.15,0.20]
    min_child_weight=[1,2,3,4]
    base_score=[0.25,0.5,0.75,1]
```

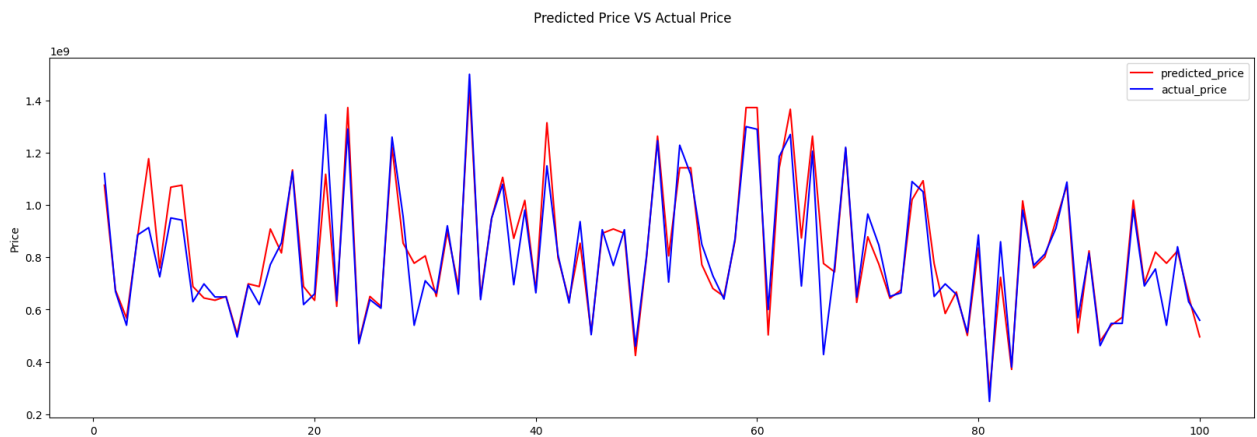
Hình 27. Khởi tạo bộ tham số cho mô hình XGBoost

Các tham số sử dụng trong quá trình huấn luyện mô hình: Sau khi sử dụng RandomizedSearchCV để tìm siêu tham số cho mô hình, các tham số tìm được như sau:

Tên tham số	Giá trị	Ý nghĩa
n_estimators	1100	Số estimators trong tìm kiếm
max_depth	5	Độ sâu tìm kiếm tối đa

booster	gbtree	Mô hình để chạy
learning_rate	0.1	Tốc độ học
min_child_weight	2	Trọng lượng tối thiểu của giá trị con
base_score	0.75	Mức điểm làm mốc tìm tham số

## Biểu đồ tương quan giữa tập test và predict



Hình 28. Tương quan giữa tập test và predict khi sử dụng mô hình XGBoost

## Đánh giá:

	<b>RMSE</b> (nghìn đồng)	<b>MAE</b> (nghìn đồng)	<b>R2</b> (%)
<b>Kết quả</b>	98,721,948.65	61,073,155.8	87.2

## 4.3. So sánh hiệu quả của các mô hình

	<b>RMSE (nghìn đồng)</b>	<b>MAE (nghìn đồng)</b>	<b>R2(%)</b>
<b>RandomForest Regression</b>	105,769,763.10	57,789,856.07	85.31
<b>XGBoost Regression</b>	98,721,948.65	61,073,155.8	87.2

### Đánh giá:

- Thông số MAE của thuật toán RandomForest Regression nhỏ hơn XGBoosts Regression, điều đó cho thấy độ sai lệch các giá trị của thuật toán RandomForest Regression ít hơn
- Hiệu suất R2 của XGBoost lại cao hơn một ít và RMSE thấp hơn cho thấy mô hình XGBoost dự đoán tốt hơn.
- Điều này xảy ra có thể do chọn các thông số của XGBoost chưa đạt hiệu quả cao nhất.
- Trong thực tế tùy vào dữ liệu và mục đích sử dụng có thể chọn một trong 2 mô hình trên.

## 5. Dự đoán và kết luận

### 5.1. Dự đoán giá một chiếc xe hơi bất kì

Ở phần này, chúng ta sẽ sử dụng mô hình sau khi lưu dưới dạng file .pkl để tiến hành áp dụng dự đoán giá trong thực tế

#### Load file.pkl:

```
[ ] import pickle

with open('model_XGB.pkl', 'rb') as file:
    model_xgb_loaded = pickle.load(file)
```

Hình 29. Load file.pkl

**Tạo đối tượng mới:** ở đây chúng ta sẽ giả định tạo một đối tượng mới để dự đoán giá của nó theo mô hình đã train được

```
[ ] car = {'brand': ['Toyota'],
           'name': ['Toyota Avanza Premio'],
           'year': [2022],
           'model': ['SUV'],
           'seat': [7],
           'fuel': ['Xăng'],
           'capacity': [1.5],
           'origin': ['Lắp ráp trong nước'],
           'gear': ['Số tự động']}
```

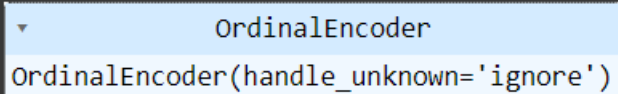
Hình 30. Tạo một đối tượng xe hơi mới

### Fit các thuộc tính của dữ liệu vào 2 lớp OrdinalEncoder và StandardScaler:

- OrdinalEncoder: Fit các đối tượng có dữ liệu Object thành dạng số
- StandardScaler: Chuẩn hóa các dữ liệu dạng số sau khi đã encode

```
[ ] from sklearn.preprocessing import OrdinalEncoder

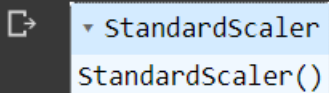
encoder = OrdinalEncoder()
encoder.fit(df)
encoder.set_params(handle_unknown='ignore')
```



Hình 31. Sử dụng lớp OrdinalEncoder

```
▶ from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(df)
```



Hình 32. Sử dụng lớp StandardScaler

### Fit các thuộc tính của đối tượng mới tạo vào 2 lớp trên:

```
[ ] def predict_car(car):
    car_df = pd.DataFrame(car)
    car_df_transform = encoder.transform(car_df)
    car_df_transform = scaler.transform(car_df_transform)
    prediction = model_xgb_loaded.predict(car_df_transform)
    prediction = int(float(prediction))
    return prediction
```

Hình 33. Hàm dự đoán giá

### Kết quả:

```
▶ print("Giá dự đoán: ", predict_car(car), "VNĐ")

Giá dự đoán: 706853184 VNĐ
```

Hình 34. Kết quả dự đoán



## 5.2. Mô hình hóa dữ liệu

- Đã xây dựng được 2 mô hình dự đoán giá xe hơi dựa trên 2 thuật toán là Random Forest Regression và XGBoostRegression với độ chính xác được đánh giá khá cao. Random Forest Regression cho  $R^2$  là 85.31%. Với mô hình XGBoost Regression cho số liệu đáng tin cậy hơn Linear Regression với kết quả  $R^2$  là 87.2%.
- Có thể tăng độ chính xác của mô hình XGBoost Regression bằng việc thay đổi các bộ tham số đầu vào của mô hình để chọn ra bộ tham số tối ưu hơn
- Nhìn chung, việc sử dụng thuật toán nào trong các bài toán khác phụ thuộc vào dữ liệu đầu vào và kết quả mong muốn của bài toán đó.

## 6. Tài liệu tham khảo

[1] [Beautiful Soup](#)

[2] [Random Forest Regression](#)

[3] [XGBoost For Regression](#)

[4] [MAE, MSE and RMSE](#)

[5] [Visualize Data](#)