TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

TIỂU LUẬN CUỐI KỲ

HỌC PHẦN: KHOA HỌC DỮ LIỆU

Dự đoán giá ô tô

HỌ VÀ TÊN SINH VIÊN	LỚP HỌC PHẦN	ĐIỂM BẢO VỆ
Nguyễn Hoàng Trường	19N13	
Quách Minh Nhật	19N13	
Ngô Lê Gia Hưng	19N13	

ĐÀ NẴNG, 06/2022

TÓM TẮT

Bài toán đặt ra: Thực hiện thu thập dữ liệu về các loại xe ô tô trên thế giới và thực hiện dự đoán giá tiền của chiếc xe đó thông qua các đặc trưng của ô tô.

Phương pháp giải quyết vấn đề:

• Crawl dữ liệu các loại ô tô từ web và trích xuất các đặc trưng cần thiết có thể dùng để dự đoán giá tiền của loại xe đó.

• Minh họa dữ liệu để xem các đặt trưng của dữ liệu như thế nào để tiến hành thực hiện các công việc tiếp theo.

• Phân tích các đặc trưng dựa EDA và thực hiện các Feature Engineering với các hành động như bổ sung đặc trưng bị trống, chuẩn hóa dữ liệu và giảm chiều dữ liệu về dạng số để thực hiện bài toán dự đoán.

• Dự đoán giá xe đó thông qua các đặc trưng mà bộ phim đó có.

Kết quả đạt được: Giá tiền của chiếc ô tô thông qua dự đoán và các độ lệch trong việc dự đoán dựa trên các metrics từ đó rút ra được nhận xét về mô hình dự đoán.

BẢNG PHÂN CÔNG NHIỆM VỤ

Sinh viên thực hiện	Các nhiệm vụ	Tự đánh giá
Nguyễn Hoàng Trường	Xây dựng mô hình RandomForestRegressor	Đã hoàn thành
	Đánh giá độ chính xác của mô hình RandomForestRegressor	Đã hoàn thành
Quách Minh Nhật	Thu thập dữ liệu từ web	Đã hoàn thành
	Làm sạch và trực quan hóa dữ liệu	Đã hoàn thành
Ngô Lê Gia Hưng	Xây dựng mô hình LinearRegression	Đã hoàn thành
	Đánh giá độ chính xác của mô hình LinearRegression	Đã hoàn thành
	Lựa chọn đặc trưng	Đã hoàn thành

MỤC LỤC

1. Giới thiệu

- Vấn đề cần giải quyết:

Thực hiện đề tài dự đoán giá xe được đăng bán tại trang web dựa trên dữ liệu tự thu thập, làm sạch, dự đoán dựa trên hai mô hình (Linear Regression và Random Forest) và đánh giá kết quả với RMSE MAE.

- Giải pháp tổng quan:
  - Thực hiện việc crawl dữ liệu từ một nguồn web ở trên mạng (lựa chọn các website có lượng bài đăng lớn và thông phải cụ thể, đầy đủ).
  - Sau khi thu thập được dữ liệu thì tiến hành phân tích, xử lí, biến đổi và trực quan hoá dữ liệu vừa crawl được.
  - Thực hiện các bước Feature Engineering cho dữ liệu.
  - Thực hiện mục tiêu đề tài là dự đoán giá xe thông qua hai mô hình/ thuật toán . Đánh giá và so sánh kết quả giữa hai mô hình/ thuật toán.

2. Thu thập và mô tả dữ liệu

2.1. Thu thập dữ liệu

2.1.1. Nguồn dữ liệu và công cụ sử dụng:

- Nguồn dữ liệu cần thu thập ở trang https://bonbanh.com/oto là một trang web chuyên trao đổi và mua bán xe.
- Công cụ thu thập sử dụng thư viện BeautifulSoup là một thư viện hỗ trợ phân tách và truy xuất dữ liệu từ các tệp HTML và XML .
- Thư viện requests_html giúp xác định và gửi HTTP request đến đường link cần thu thập.

2.1.2. Cách thức thực hiện crawl dữ liệu:

a. Lấy dữ liệu :

- Khởi tạo vòng for với n lần là số trang crawl dữ liệu. Sử dụng HTMLSession từ thư viện request-html để gửi HTTP request đến đường link "https://bonbanh.com/oto/page," + str(k)

```
for k in range(1,n):
    session = HTMLSession()
```

access_homepage = "https://bonbanh.com/oto/page,"+str(k)
req=session.get(access_homepage)

- Lấy toàn bộ trang html trả về:

element = req.html.html

- Dữ liệu trang html được lưu vào element có dạng:

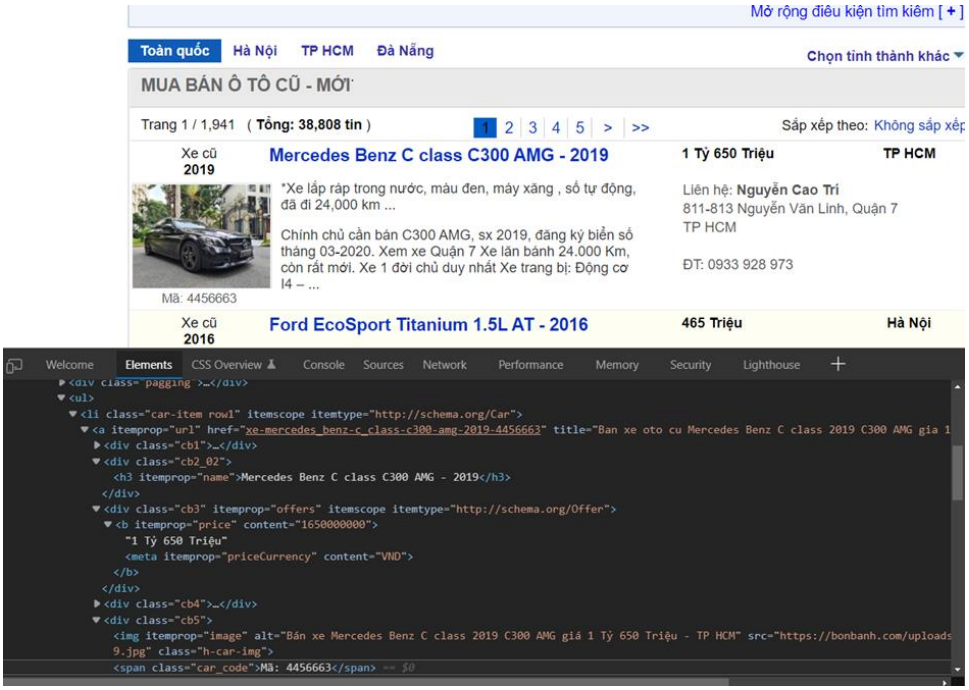
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd"> <html lang="vi"> <head> …

b. Phân tích dữ liệu :

- Sau khi đã lấy được trang html, cần tách các dữ liệu ra thành dạng cây để thuận tiện hơn cho quá trình truy xuất dữ liệu. Cách phổ biến là dùng module Beautiful Soup.

soup = BeautifulSoup(element,'html.parser')

- Phân tích dữ liệu cần thu thập có thể lấy được ở đâu bằng cách ấn F12



Hình 1: Phân tích dữ liệu tại trang bài đăng

- Phân tích thấy được các dữ liệu cần thu thập đều nằm trong thẻ <li> có thuộc tính class="car-item". Tiến hành thực hiện việc thu thập dữ liệu với BeautifulSoup . Sử dụng một vòng for-loop để duyệt qua tất cả các product có trong product_list có trong thẻ <li> với class="car-item"

product_list= soup.findAll("li",{"class":"car-item"})

for product in product_list:

- Tiếp tục phân tích cách thành phần có trong thẻ <li> thì có:
  - Link dẫn đến trang thông tin chi tiết bài đăng ở trong thẻ <a>, thực hiên lấy **url** của các bài viết đó trong thuộc tính href :

    link=product.find("a").get('href')

  - Thu thập các dữ liệu cần thiết trong bài đăng (mã xe, tên xe, giá):

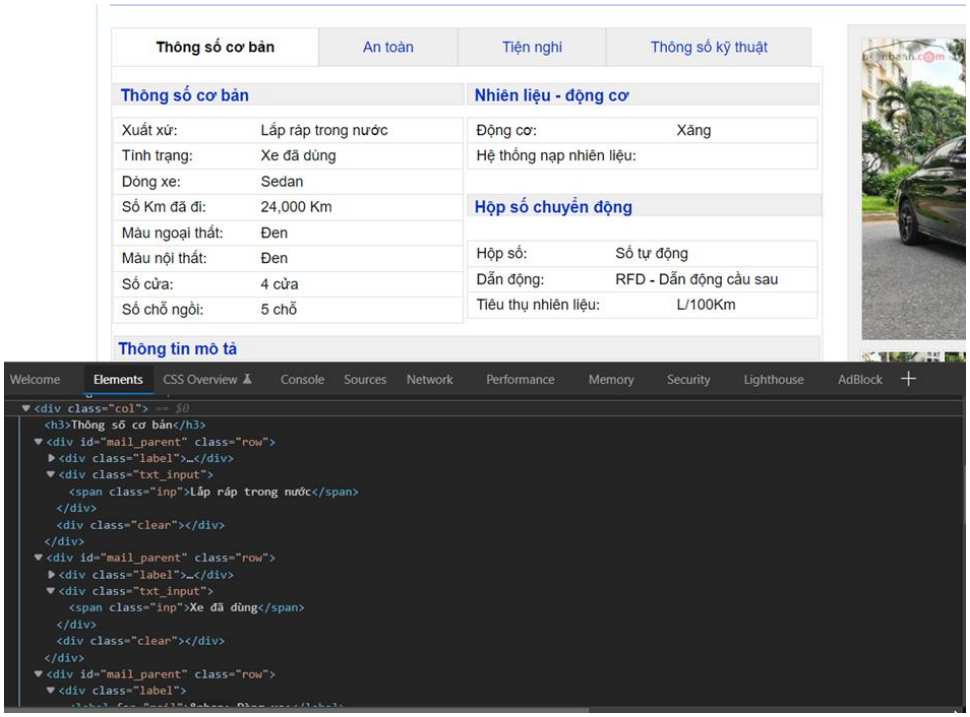
    id=product.find("span",{"class":"car_code"}).text.strip()

    Name=product.find("div",{"class":"cb2_02"}).text.strip()

    Price=product.find("b",{"itemprop":"price"}).get("content")

c. Lấy dữ liệu chi tiết của từng bài :

- Khởi tạo hàm get_detail(id, price, name, url) nhận các tham số là **url** đã có được từ thuộc tính href (thẻ a) cùng với các dữ liệu vừa thu thập của bài đăng , viết hàm để truy cập và lấy thông tin chi tiết từng bài đăng thông qua 'https://bonbanh.com/' + **url**
- Tương tự bước truy cập bài đăng , tiếp tục phân tích trang html và lấy ra các dữ liệu cần thiết



Hình 2: Phân tích dữ liệu tại trang thông tin

- Phân tích thấy các dữ liệu cần thu thập đều nằm trong thẻ nằm trong thẻ với class="inp".Tiến hành thực hiện việc thu thập dữ liệu với BeautifulSoup

  basicInfo= soup.find("div",{"class":"col"})

  origin=basicInfo.findAll("span",{"class":"inp"})[0].text

  condition=basicInfo.findAll("span",{"class":"inp"})[1].text

  carType=basicInfo.findAll("span",{"class":"inp"})[2].text…..

- Khởi tạo **carDetail=list()** và thêm các giá trị crawl được với id tương ứng:

  carDetail.append({'id':id,'car':name,'price':price,…})

- Cuối cùng thực hiện ghi dữ liệu vào file csv để hoàn tất việc crawl data:

  data = pd.DataFrame.from_dict(carDetail)

  print(data)

  data.to_csv('raw_data.csv', index=False,encoding="utf-8-sig",
  columns=['id','car','price',…])

2.2. Mô tả dữ liệu

2.2.1. Tổng quan về tập dữ liệu

- Số lượng mẫu: 5980 mẫu.
- Số đặc trưng của 1 mẫu: 14.
- Kiểu dữ liệu của các đặc trưng:

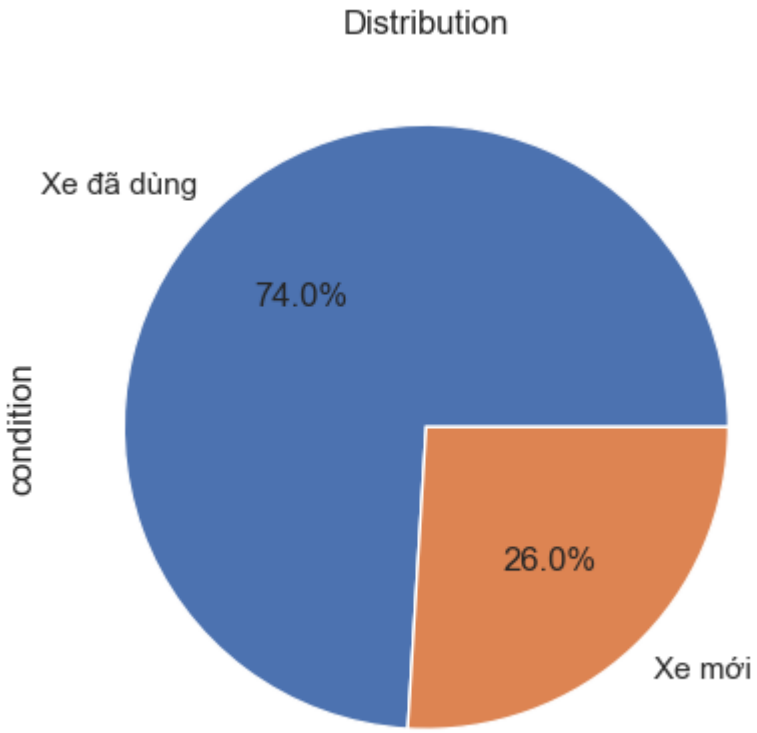
Đặc trưng	Kiểu dữ liệu
id	Object
car	Object
price	Int64
origin	Object
condition	Object
carType	Object
km	Object
carColor	Object
interiorColor	Object
carDoor	Object

carSeat	Object
engine	Object
gear	Object
wheelDrive	Object

- Số mẫu dữ liệu trống của mỗi đặc trưng:

Đặc trưng	Số mẫu dữ liệu trống
Id	0
Car	0
Price	0
Origin	0
Condition	0
CarType	0
Km	0
CarColor	4
InteriorColor	313
CarDoor	0
CarSeat	16
Engine	7
Gear	5
WheelDrive	24

2.2.2. Thống kê mô tả trực quan về các đặc trưng

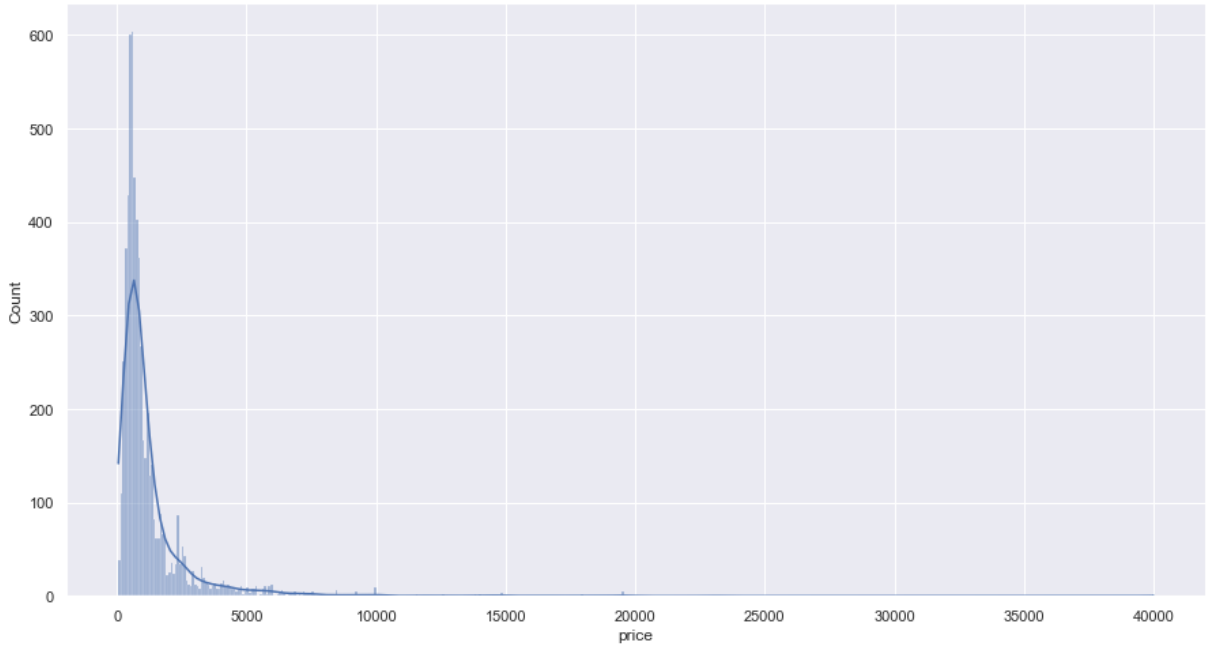


Hình 3: Biểu đồ tròn thể hiện tỉ lệ của xe đã dùng và xe mới có trong dataset

- Nhận xét:

  * Lượng xe đã dùng chênh lệch lớn so với xe mới

  * Nhìn chung trang web chủ yếu là nơi mua bán các dòng xe đã qua sử dụng

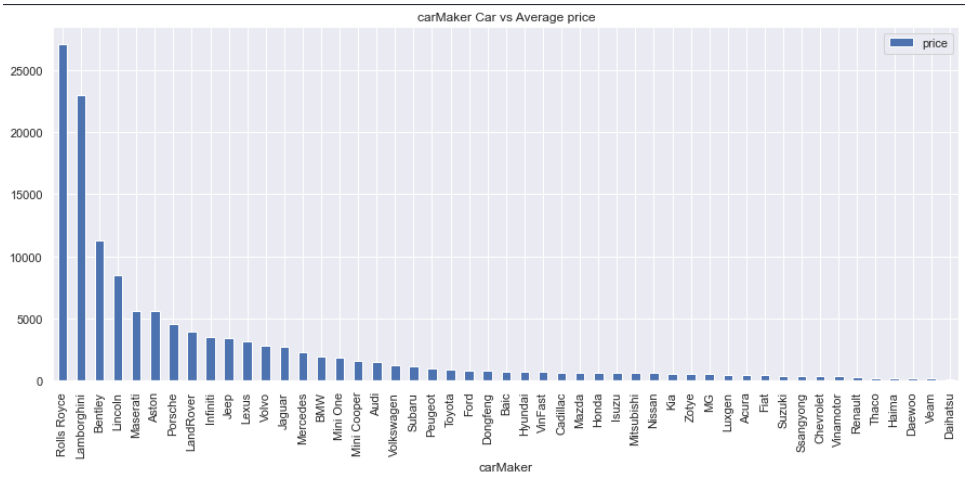


Hình 4: Đồ thị cột thể hiện giá tiền của tất cả xe được đăng bán

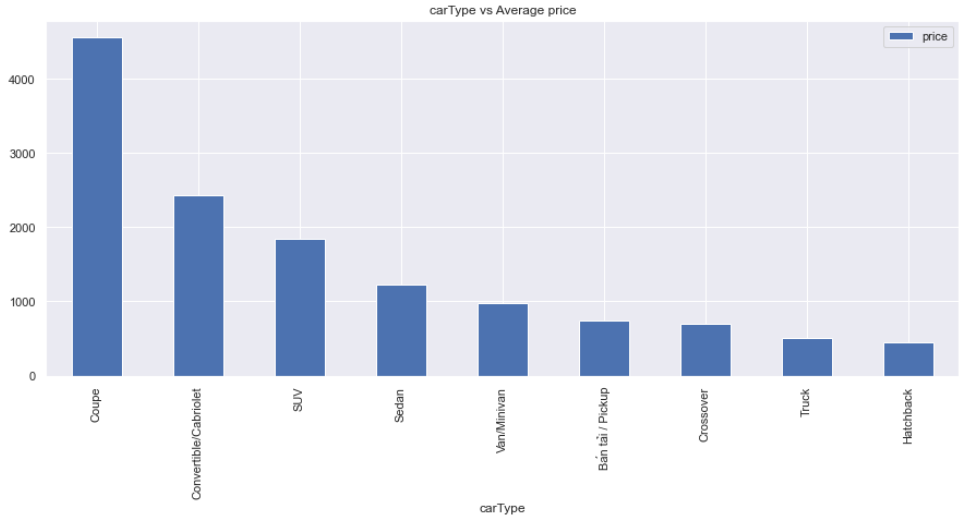
- Nhận xét:

  * Các xe được đăng bán tại trang web có giá thành tập trung trong khoảng < 5 tỷ

  * Giá xe phổ biến nhất là từ 0.5 - 2 tỷ



Hình 5: Khảo sát giá trung bình của từng hãng, loại xe

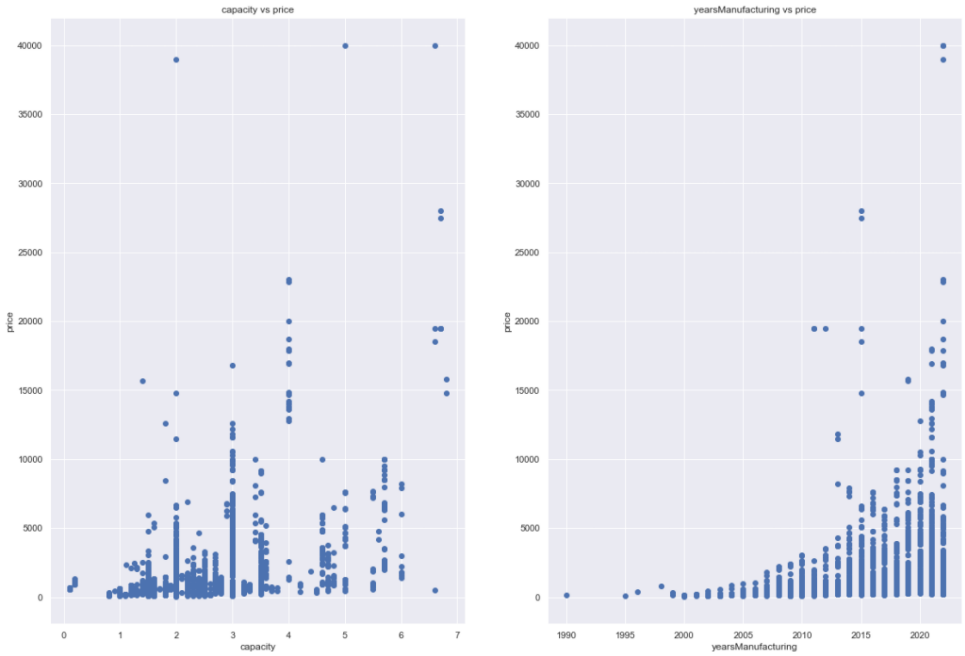


Hình 6: Đồ thị cột thể hiện mức giá trung bình theo từng hãng , loại xe.

- Nhận xét:

  * Các hãng xe cao cấp như Roll Royce, Lambor, Burnley,.. có mức giá trung bình cao hơn hẳn so với các hãng khác.

  * Dòng xe coupe là dòng xe có giá trị đắt đỏ nhất



Hình 7: Đồ thị phân tán thể hiện tương quan giữa giá tiền với các đặc trưng.

- Nhận xét:

  * Giá bán của xe có xu hướng ngày càng tăng qua các năm

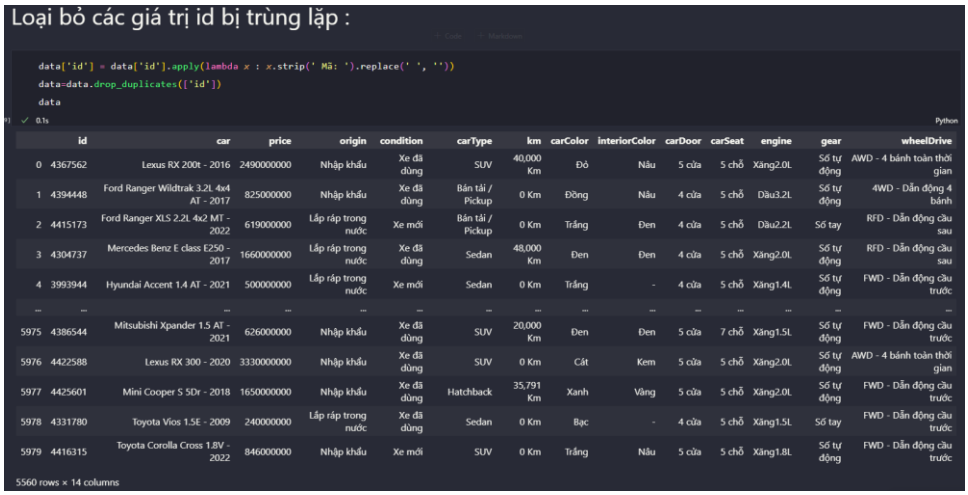
  * Các dòng xe có mức giá phổ thông (< 3 tỷ) có dung tích của động cơ tập trung nhiều trong khoảng từ 1-3L

3. Trích xuất đặc trưng

3.1. Làm sạch dữ liệu:

3.1.1. Loại bỏ các giá trị id bị lặp

- Với dữ liệu được thu thập, dữ liệu có các mẫu với các id trùng nhau, vì thế chúng ta sẽ loại bỏ các mẫu có id bị trùng lặp nhau chỉ giữ lại 1 mẫu.



	id	car	price	origin	condition	carType	km	carColor	interiorColor	carDoor	carSeat	engine	gear	wheelDrive
0	4367562	Lexus RX 200t - 2016	2490000000	Nhập khẩu	Xe đã dùng	SUV	40,000 Km	Đỏ	Nâu	5 cửa	5 chỗ	Xăng2.0L	Số tự động	AWD - 4 bánh toàn thời gian
1	4394448	Ford Ranger Wildtrak 3.2L 4x4 AT - 2017	825000000	Nhập khẩu	Xe đã dùng	Bán tải / Pickup	0 Km	Đồng	Nâu	4 cửa	5 chỗ	Dầu3.2L	Số tự động	4WD - Dẫn động 4 bánh
2	4415173	Ford Ranger XLS 2.2L 4x2 MT - 2022	619000000	Lắp ráp trong nước	Xe mới	Bán tải / Pickup	0 Km	Trắng	Đen	4 cửa	5 chỗ	Dầu2.2L	Số tay	RFD - Dẫn động cầu sau
3	4304737	Mercedes Benz E class E250 - 2017	1660000000	Lắp ráp trong nước	Xe đã dùng	Sedan	48,000 Km	Đen	Đen	4 cửa	5 chỗ	Xăng2.0L	Số tự động	RFD - Dẫn động cầu sau
4	3993944	Hyundai Accent 1.4 AT - 2021	500000000	Lắp ráp trong nước	Xe mới	Sedan	0 Km	Trắng	-	4 cửa	5 chỗ	Xăng1.4L	Số tự động	FWD - Dẫn động cầu trước
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5975	4386544	Mitsubishi Xpander 1.5 AT - 2021	626000000	Nhập khẩu	Xe đã dùng	SUV	20,000 Km	Đen	Đen	5 cửa	7 chỗ	Xăng1.5L	Số tự động	FWD - Dẫn động cầu trước
5976	4422588	Lexus RX 300 - 2020	3330000000	Nhập khẩu	Xe đã dùng	SUV	0 Km	Cát	Kem	5 cửa	5 chỗ	Xăng2.0L	Số tự động	AWD - 4 bánh toàn thời gian
5977	4425601	Mini Cooper S 5Dr - 2018	1650000000	Nhập khẩu	Xe đã dùng	Hatchback	35,791 Km	Xanh	Vàng	5 cửa	5 chỗ	Xăng2.0L	Số tự động	FWD - Dẫn động cầu trước
5978	4331780	Toyota Vios 1.5E - 2009	240000000	Lắp ráp trong nước	Xe đã dùng	Sedan	0 Km	Bạc	-	4 cửa	5 chỗ	Xăng1.5L	Số tay	FWD - Dẫn động cầu trước
5979	4416315	Toyota Corolla Cross 1.8V - 2022	846000000	Nhập khẩu	Xe mới	SUV	0 Km	Trắng	Nâu	5 cửa	5 chỗ	Xăng1.8L	Số tự động	FWD - Dẫn động cầu trước

5560 rows × 14 columns

Hình 8: Loại bỏ dữ liệu trùng lặp

- Sau khi loại bỏ các mẫu có id trùng nhau, dữ liệu còn 5560 mẫu.

3.1.2. Xử lý dữ liệu của các đặc trưng

- Các đặc trưng mang giá trị không được sạch, thừa, xóa bỏ các giá trị đó.

Hình 9: Xử lý giá trị của các đặc trưng

- Price: Chuyển dữ liệu giá xe về đơn vị triệu
- Km: Xóa ký tự “Km” có trong đặc trưng
- CarDoor: Xóa ký tự “cửa” có trong đặc trưng
- CarSeat: Xóa ký tự “chỗ” có trong đặc trưng
- WheelDrive: Đổi kí tự “-“ thành kí tự “,”

3.1.3. Tách dữ liệu đặc trưng

- Ở đặc trưng Car, giá trị của đặc trưng có thể tách làm các đặc trưng khác

Thêm 2 cột carMaker (Hãng sản xuất) và yearsManufacturing (Năm phát hành) từ cột car

```
def toCarMaker(Car):
    if(Car[0] == "Rolls") or (Car[0] == "Mini"):
        return str(Car[0])+" "+str(Car[1])
    else:
        return str(Car[0])
def toyearsManufacturing(year):
    return str(year[len(year)-1])
data['carMaker'] = data['car'].apply(lambda x : x.strip().split())
data['yearsManufacturing'] = data['carMaker'].apply(toyearsManufacturing)

data['carMaker'] = data['carMaker'].apply(toCarMaker)
```

0.1s

Hình 10: Trích xuất đặc trưng từ đặc trưng Car

- Tương tự ở đặc trưng engine có thể tách ra làm các đặc trưng về nhiên liệu, và dung tích:

Tách engine thành 2 cột carEngine và capacity

+ Code + Markdown

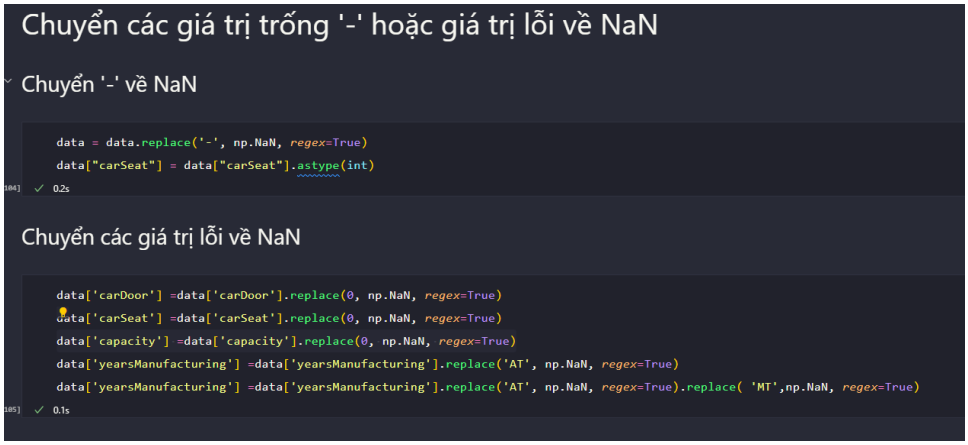
```
def convertCapacity(capacity):
    if(capacity > 100):
        return capacity/100
    else:
        return capacity/10

# Xử lí chuyển dữ liệu trong engine về loại động cơ
data['capacity']=data['engine'].str.replace(r'\D+','')
# data['capacity'] = data['capacity'].apply(lambda x: ','.join(map(str, x)))
data['capacity'] = data['capacity'].apply(lambda x : pd.to_numeric(x))
data['capacity']=data['capacity'].apply(convertCapacity)



data['carEngine'] = data['engine'].str.replace(r'\d+','')
data['carEngine'] = data['carEngine'].apply(lambda x : x.replace('.L',''))
```

0.1s Python

Hình 11: Trích xuất đặc trưng từ đặc trưng Engine

3.1.4. Xử lý các giá trị có kí tự đặc biệt:

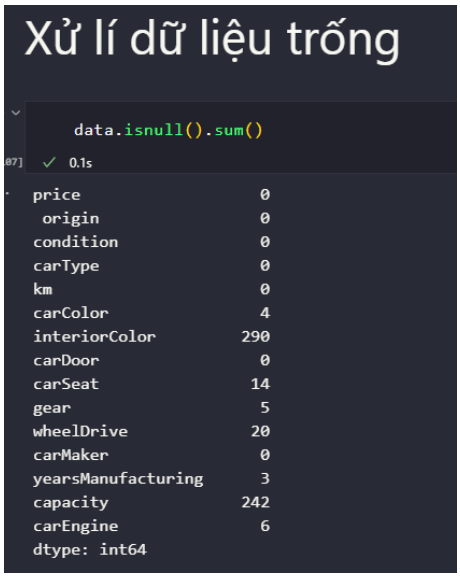
- Ở dữ liệu thu được, các đặc trưng có các kí tự đặc biệt trong giá trị của chúng, như: “-” , “AT”, “MT”



Hình 12: Xữ lý dữ liệu đặc biệt

3.1.5. Xử lý dữ liệu trống

- Tổng số lượng dữ liệu trống

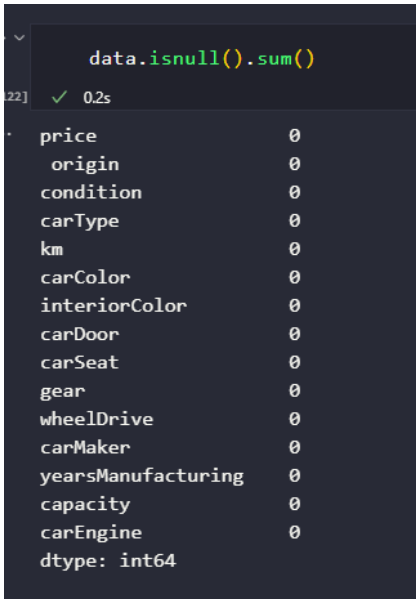


Hình 13: Số lượng các dữ liệu trống

- Thay thế các giá trị trống bằng giá trị ngẫu nhiên

Hình 14: Kỹ thuật xử lý dữ liệu trống bằng giá trị ngẫu nhiên

- Kết quả quá trình xử lý dữ liệu trống



Hình 15: Kết quả quá trình xử lý dữ liệu trống

3.2. Chọn đặc trưng cơ bản:

- Các đặc trưng cơ bản hầu hết được giữ lại, chỉ loại bỏ các đặc trưng như “Id”, “Car”

```
data = data.drop(['id'], axis=1)
data = data.drop(['car'], axis=1)
data
```

✓ 0.2s

	price	origin	condition	carType	km	carColor	interiorColor	carDoor	carSeat	engine	gear	wheelDrive	carMaker	yearsManufacturing
0	2490.0	Nhập khẩu	Xe đã dùng	SUV	40000	Đỏ	Nâu	5	5.0	Xăng	Số tự động	AWD,4 bánh toàn thời gian	Lexus	2016
1	825.0	Nhập khẩu	Xe đã dùng	Bán tải / Pickup	0	Đồng	Nâu	4	5.0	Dầu	Số tự động	4WD,Dẫn động 4 bánh	Ford	2017
2	619.0	Lắp ráp trong nước	Xe mới	Bán tải / Pickup	0	Trắng	Đen	4	5.0	Dầu	Số tay	RFD,Dẫn động cầu sau	Ford	2022
3	1660.0	Lắp ráp trong nước	Xe đã dùng	Sedan	48000	Đen	Đen	4	5.0	Xăng	Số tự động	RFD,Dẫn động cầu sau	Mercedes	2017
4	500.0	Lắp ráp trong nước	Xe mới	Sedan	0	Trắng	NaN	4	5.0	Xăng	Số tự động	FWD,Dẫn động cầu trước	Hyundai	2021
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5975	626.0	Nhập khẩu	Xe đã dùng	SUV	20000	Đen	Đen	5	7.0	Xăng	Số tự động	FWD,Dẫn động cầu trước	Mitsubishi	2021
5976	3330.0	Nhập khẩu	Xe đã dùng	SUV	0	Cát	Kem	5	5.0	Xăng	Số tự động	AWD,4 bánh toàn thời gian	Lexus	2020
5977	1650.0	Nhập khẩu	Xe đã dùng	Hatchback	35791	Xanh	Vàng	5	5.0	Xăng	Số tự động	FWD,Dẫn động cầu trước	Mini Cooper	2018
5978	240.0	Lắp ráp trong nước	Xe đã dùng	Sedan	0	Bạc	NaN	4	5.0	Xăng	Số tay	FWD,Dẫn động cầu trước	Toyota	2009
5979	846.0	Nhập khẩu	Xe mới	SUV	0	Trắng	Nâu	5	5.0	Xăng	Số tự động	FWD,Dẫn động cầu trước	Toyota	2022

5560 rows × 14 columns

Hình 16: Xóa bỏ đặc trưng “Id”, “Car”

3.3. Chuẩn hóa dữ liệu và giảm chiều dữ liệu

3.3.1. Đặc trưng dùng để đánh giá và chia tập dữ liệu

- Sử dụng đặc trưng Price để đánh giá quá trình dự đoán, chia tập dữ liệu làm 2 tập train và test theo tỉ lệ 80/20

```
from sklearn.preprocessing import MinMaxScaler, RobustScaler

df_train, df_test = train_test_split(data, test_size = 0.2, random_state=42)

print(df_train.shape)
print(df_test.shape)

# Có xử lý ngoại lệ
X_train = df_train.copy()
X_test = df_test.copy()

#Dividing data into X and y variables
y_train = X_train.pop('price')
y_test = X_test.pop('price')
```

Hình 17: Chia tập dữ liệu

3.3.2. Chuẩn hóa dữ liệu:

- Sử dụng LabelEncoder để chuẩn hóa dữ liệu thành dữ liệu số

Encode

```
def label_encode_columns(df, columns):
    for col in columns:
        # le = LabelEncoder().fit(df[col])
        # df[col] = le.fit_transform(df[col])
        # encoders[col] = le
        df[col] = LabelEncoder().fit_transform(df[col])

    return df

data = label_encode_columns(data,[' origin', 'condition', 'carType', 'carColor', 'interiorColor', 'engine', 'gear', 'wheelDrive', 'carMaker'])
data
```

[200] ✓ 0.1s

	price	origin	condition	carType	km	carColor	interiorColor	carDoor	carSeat	engine	gear	wheelDrive	carMaker	yearsManufacturing
0	2490	1	1	5	40000	15	8	5	5	3	2	2	23	2016
1	825	1	1	0	0	16	8	4	5	0	2	1	12	2017
2	619	0	0	0	0	9	13	4	5	0	1	4	12	2022
3	1660	0	1	6	48000	14	13	4	5	3	2	4	29	2017
4	500	0	0	6	0	9	13	4	5	3	2	3	15	2021

Hình 18: Encode dữ liệu

3.3.1. Giảm chiều dữ liệu

- MinMaxScaler chia tỷ lệ tất cả các đối tượng dữ liệu trong phạm vi [0, 1] hoặc các đối tượng khác trong phạm vi [-1, 1] nếu có giá trị âm trong tập dữ liệu.

```
sc = MinMaxScaler()
X_train = pd.DataFrame(sc.fit_transform(X_train),columns=X_train.columns)
X_test = pd.DataFrame(sc.fit_transform(X_test),columns=X_test.columns)

X_train
```

✓ 0.1s

```
(4448, 15)
(1112, 15)
```

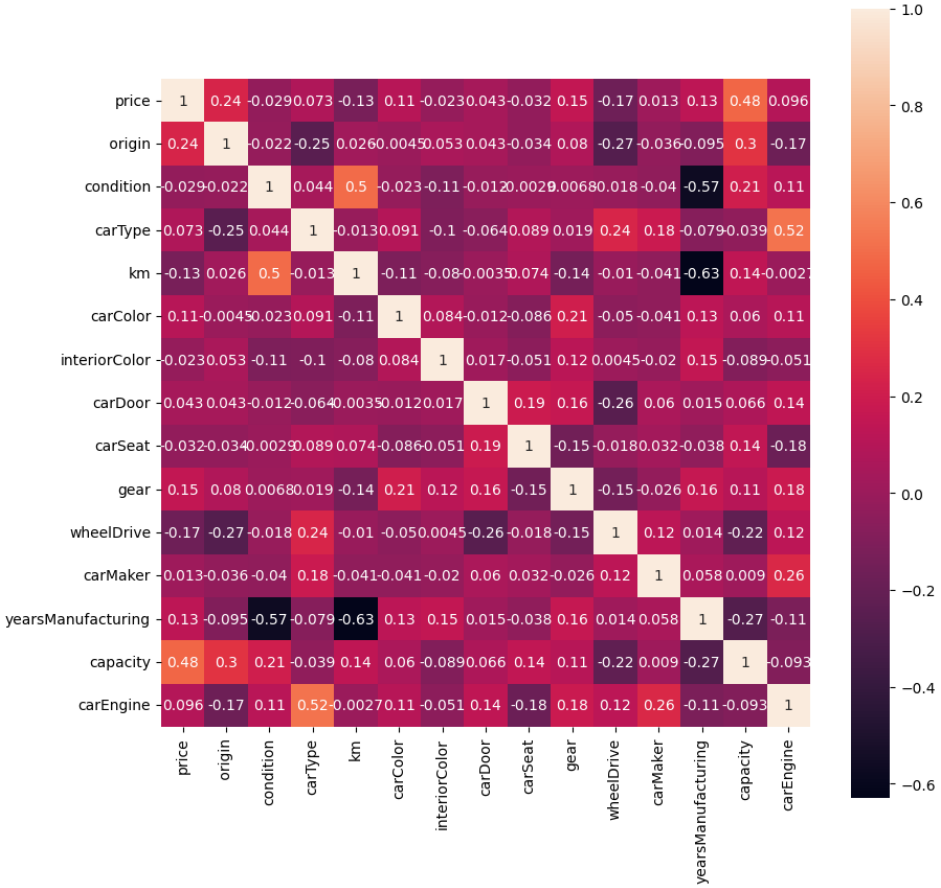
	origin	condition	carType	km	carColor	interiorColor	carDoor	carSeat	gear	wheelDrive	capacity	carEngine	carMaker	yearsManufacturing
0	1.0	1.0	0.750	0.000000	0.8750	0.333333	0.666667	0.086957	1.0	0.75	0.283582	0.75	0.875000	0.888889
1	0.0	1.0	0.750	0.074074	0.5625	0.866667	0.666667	0.086957	1.0	0.75	0.253731	0.75	0.875000	0.962963
2	0.0	1.0	1.000	0.000000	0.7500	0.200000	0.333333	0.717391	0.5	1.00	0.253731	0.00	0.354167	0.851852
3	0.0	1.0	0.375	0.055556	0.1250	0.200000	0.833333	0.152174	0.5	1.00	0.283582	0.75	0.875000	0.629630
4	0.0	1.0	0.625	0.092593	0.8750	0.533333	0.833333	0.130435	1.0	1.00	0.283582	0.75	0.916667	0.925926
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4443	0.0	1.0	0.750	0.324074	0.8750	0.866667	0.666667	0.086957	1.0	0.75	0.358209	0.75	0.875000	0.740741
4444	1.0	1.0	0.000	0.370370	0.0625	0.466667	0.666667	0.086957	1.0	0.25	0.462687	0.00	0.250000	0.777778
4445	1.0	0.0	0.750	0.000000	0.5625	0.866667	0.666667	0.086957	1.0	0.75	0.208955	0.75	0.541667	1.000000
4446	0.0	1.0	0.625	0.000000	0.5625	0.400000	0.833333	0.086957	1.0	0.50	0.358209	0.75	0.583333	0.888889
4447	1.0	1.0	0.750	0.083333	0.8750	0.866667	0.666667	0.086957	1.0	0.75	0.358209	0.75	0.875000	0.888889

4448 rows × 14 columns

Hình 19: MinMaxScaler giảm chiều dữ liệu

3.4. Kết quả

- Độ tương quan giữa các đặc trưng



Hình 20: Đồ thị tương quan giữa các đặc trưng

4. Mô hình hóa dữ liệu

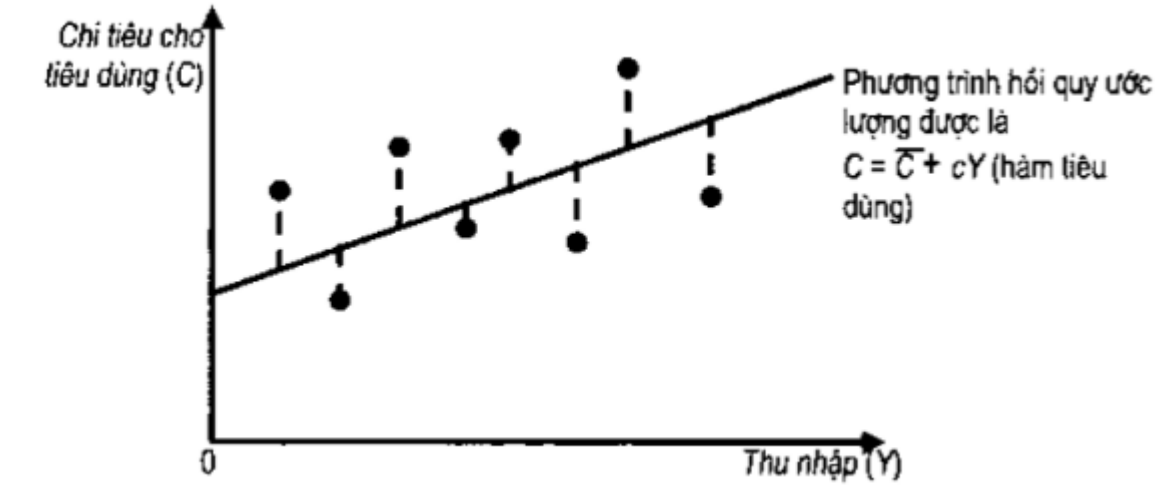
4.1. Cơ sở lý thuyết

4.1.1. Regression analysis – Phân tích hồi quy:

Phân tích hồi qui trong tiếng Anh là Regression analysis. Phân tích hồi qui là kĩ thuật thống kê dùng để ước lượng phương trình phù hợp nhất với các tập hợp kết quả quan sát của biến phụ thuộc và biến độc lập.

Phân tích hồi qui cho phép đạt được kết quả ước lượng tốt nhất về mối quan hệ chân thực giữa các biến số. Từ phương trình ước lượng được này, người ta có thể dự báo về biến phụ thuộc (chưa biết) dựa vào giá trị cho trước của biến độc lập (đã biết).

Ví dụ: Ví dụ đơn giản nhất về một phương trình tuyến tính với một biến độc lập và một biến phụ thuộc, chẳng hạn thu nhập sử dụng và chi tiêu cho tiêu dùng. Vấn đề đặt ra là phải vẽ được đường thẳng phù hợp nhất với tập hợp số liệu bao gồm các cặp kết quả quan sát về thu nhập (Y) và tiêu dùng (C).



Hình 21: Phân tích hồi qui

4.1.2. Mô hình hồi quy được sử dụng trong bài toán dự đoán

- **Linear Regression – Hồi quy tuyến tính**

"Hồi quy tuyến tính" là một phương pháp thống kê để hồi quy dữ liệu với biến phụ thuộc có giá trị liên tục trong khi các biến độc lập có thể có một trong hai giá trị liên tục hoặc là giá trị phân loại. Nói cách khác "Hồi quy tuyến tính" là một phương pháp để dự đoán biến phụ thuộc (Y) dựa trên giá trị của biến độc lập (X). Nó có thể được sử dụng cho các trường hợp chúng ta muốn dự đoán một số lượng liên tục.

Cụ thể API của mô hình trong thư viện sklearn có dạng:

class sklearn.linear_model.LinearRegression(*, fit_intercept=True,

normalize='deprecated', copy_X=True, n_jobs=None, positive=False)

Với các tham số được cấu hình trên việc ta xây dựng một mô hình Linear Regression tương đối dễ dàng.

- **Random Forest Regressor:**

Random Forest n là một công cụ ước tính meta phù hợp với một số cây quyết định phân loại trên các mẫu con khác nhau của tập dữ liệu và sử dụng giá trị trung bình để cải thiện độ chính xác của dự đoán và kiểm soát việc phù hợp quá mức. Kích thước mẫu con được kiểm soát bằng max_samplestham số if bootstrap=True(mặc định), nếu không thì toàn bộ tập dữ liệu được sử dụng để xây dựng từng cây.

Cụ thể API của mô hình trong thư viện sklearn có dạng:

Class sklearn.ensemble.RandomForestRegressor(n_estimators=100,*, criterion='squared_error',max_depth=None,min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0,max_features=1.0,max_leaf_nodes=None,min_impurit

y_decrease=0.0,bootstrap=True,oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None)

Với các tham số thường dùng:

+ n_estimators int, default = 100

Số lượng cây trong rừng.

+ max_features*{"sqrt", "log2", None}*

số lượng tính năng được xem xét ở mỗi lần phân chia

+ max_depth: int, default=None

số mức tối đa được phép trong mỗi cây quyết định

+ min_samples_split int or float, default=2

số mẫu tối thiểu để tách một nút

+min_samples_leafint or float, default=1

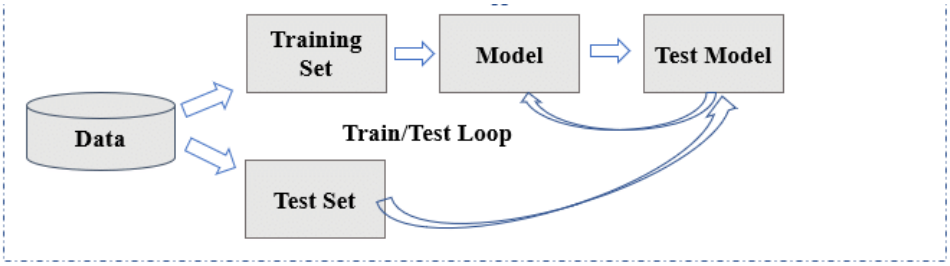
số mẫu tối thiểu có thể được lưu trữ trong một nút lá

+ bootstrapbool, default=True

phương pháp được sử dụng để lấy mẫu các điểm dữ liệu

4.2. Giải quyết bài toán

4.2.1 Quá trình phân chia và sử dụng dữ liệu



Hình 22 :quá trình xây dựng một model

Như ảnh trên ta thấy:

B1:Chia tập Data thành 2 phần là Tranning Set và Test Set với tỉ lệ 0.7 và 0.3

B2: Lấy phần Trainning Set vào Xây dựn model

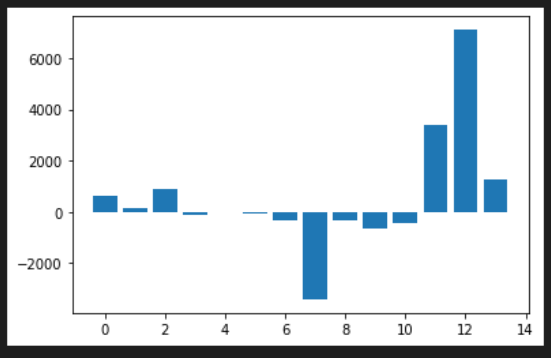
B3: sử dụng Test Set để đánh giá model

4.2.2. Thực hiện dự đoán với mô hình Linear Regression

- Tìm các đặc trưng quan trọng đối với mô hình Linear Regression

```
# define the model
model = LinearRegression()
# fit the model
model.fit(X_train, y_train)
# get importance
importance = model.coef_
# summarize feature importance
# plot feature importance
plt.bar([x for x in range(len(importance))], importance)
plt.show()
```

✓ 0.2s



Hình 23: Đồ thị biểu diễn các đặc trưng quan trọng

Như vậy chúng ta sẽ lấy các đặc trưng 'origin', 'condition', 'carType', 'capacity', 'carEngine', 'yearsManufacturing' để dự đoán mô hình.

Với Linear Regression (Sử dụng với tham số mặc định) và sử dụng metric : RMSE, MAE để đánh giá quá trình train mà ta thu được

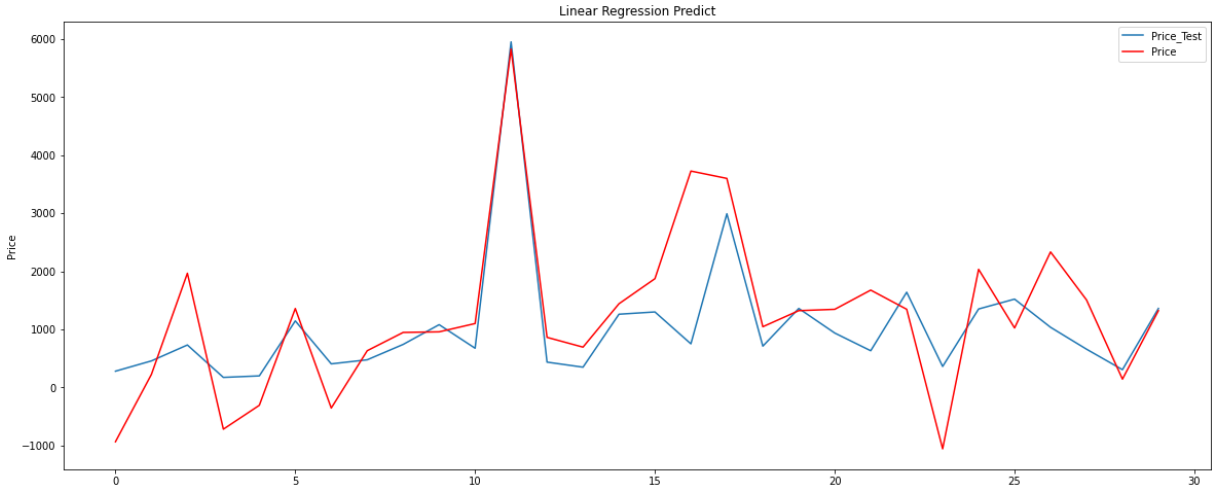
```
# Dùng LinerRegression
LR = LinearRegression()
LR.fit(X_train_LR, y_train)
# Dự đoán
predict_train = LR.predict(X_train_LR)
predict_test  = LR.predict(X_test_LR)
rs = {
  "Price_Test": y_test[:10],
  "Price": predict_test[:10]
}
df = pd.DataFrame(rs)

print("Mean Y_test: {0}".format(np.mean(y_test)))
print("rmse of test set : {0}".format(mean_squared_error(y_test,predict_test)**0.5))
print("rmse of test set : {0}".format(mean_absolute_error(y_true = y_test,y_pred = predict_test)))
```

✓ 0.1s

```
Mean Y_test: 1330.8705035971223
rmse of test set : 1408.3258088231755
rmse of test set : 786.3476432701431
```

Hình 24: Đồ thị tương quan giữa các đặc trưng

Kết quả áp dụng mô hình trên cho dữ liệu test để xem độ chính xác của dự đoán của bài toán Linear Regression



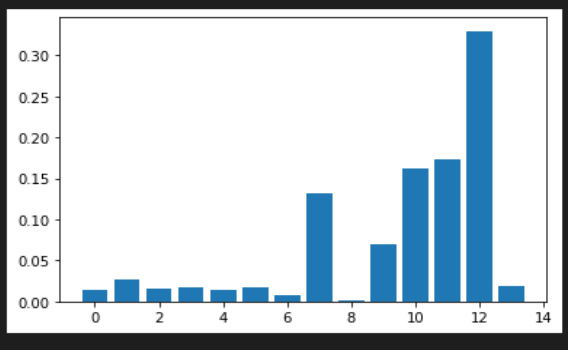
Hình 25: Kết quả dự đoán với 30 nhãn đầu tiên

4.2.3. Thực hiện dự đoán với mô hình Random Forest Regressor:

- **Tìm các đặc trưng quan trọng đối với mô hình Random Forest Regressor**

```
# define the model
model = RandomForestRegressor()
# fit the model
model.fit(X_train, y_train)
# get importance
importance = model.feature_importances_
# summarize feature importance
# for i,v in enumerate(importance):
#   print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
plt.bar([x for x in range(len(importance))], importance)
plt.show()
```

[56] ✓ 11.2s



Hình 26:Biểu dồ biểu thị các đặc trưng quan trọng

Như vậy dựa vào đồ thị cho ta thấy mức độ quan trọng của các đặc trứng đối với mô hình Ramdom Forest Regressor đều quan trọng, nên đều chọn hết tất cả các đặc trưng để xây dựng

- **Chọn lọc các siêu tham số cho Random Forest Regressor**

Sử dụng các tham số thường dung là n_estimators, max_features, max_depth, min_samples_split, min_samples_leafint, bootstrapbool

- Chú thích các tham số được giải thích ở phần cơ sở lý thuyết

```
# Các siêu tham số
n_estimators = [5,20,50,100]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(10, 120, num = 12)]
min_samples_split = [2, 6, 10]
min_samples_leaf = [1, 3, 4]
bootstrap = [True, False]

Parameters = {  'n_estimators': n_estimators,
                'max_features': max_features,
                'max_depth': max_depth,
                'min_samples_split': min_samples_split,
                'min_samples_leaf': min_samples_leaf,
                'bootstrap': bootstrap}
```

Hình 27: các tham số và giá trị dùng để chọn siêu tham số

-Sử dụng RandomizedSearchCV để tìm ra bộ tham số tốt nhất

```
print ('Best Parameters: ', model_random.best_params_, ' \n')
```

Hình 28: Sử dụng RandomizedSearchCV

Kết quả:

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
Best Parameters:  {'n_estimators': 50, 'min_samples_split': 6, 'min_samples_leaf': 1, 'max_features': 'sqrt', 'max_depth': 40, 'bootstrap': False}
```

Hình 29: kết quả các siêu tham số

Sau khi tìm đc bộ tham số tốt nhất ta xây dựng model với bộ tham số trên

- **Đánh giá mô hình:**

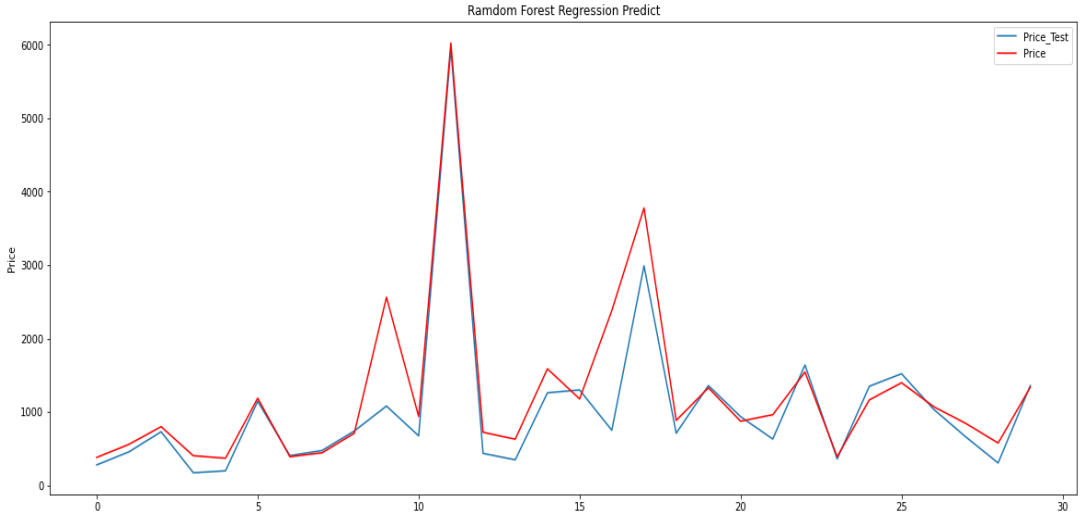
Với Random Forest Regressor ta sủ dụng metric MAE và RMSE để đánh giá mô hình sau khi sử dụng tập kiểm thử

```
# Dự đoán
predict_train = model.predict(X_train)
predict_test  = model.predict(X_test)
rs = {
  "Price_Test": y_test[:30],
  "Price": predict_test[:30]
}
df = pd.DataFrame(rs)

print("Mean Y_test: {0}".format(np.mean(y_test)))
print("rmse of test set : {0}".format(mean_squared_error(y_test,predict_test)**0.5))
print("mae of test set : {0}".format(mean_absolute_error(y_true = y_test,y_pred = predict_test)))
```

```
Mean Y_test: 1330.8705035971223
rmse of test set : 960.4565730835867
mae of test set : 382.091830133296
```

Hình 30: Kết quả sau khi sử dụng MAE và RMSE

Kết quả áp dụng mô hình trên cho dữ liệu test để xem độ chính xác của dự đoán của bài toán Random Forest Regressor



Hình 31: Kết quả dự đoán với 30 nhãn đầu tiên

4.2.4. So sánh kết quả 2 model

Thực hiện so sánh 2 model bằng metric sử dụng so sánh:

	Model	Linear Regression	Random Forest Regressor
0	RMSE	1408.325809	991.787327
1	MAE	786.347643	355.516676

Hình 32: So sánh metric của 2 model

- Random Forest Regressor tạo ra kết quả tốt hơn, hoạt động tốt trên tập dữ liệu hiện tại so với Linear Regression

- Hồi quy tuyến tính là một mô hình tuyến tính, vì thế với tập dữ liệu phi tuyến tính như dữ liệu này thì thuật toán Linear Regression cho ra kết quả xấu hơn

5. Kết luận

Tổng kết:

- Nhóm đã thực hiện và hoàn thiện tất cả các yêu cầu tổng quan của bài tập thông qua các cộng đoạn crawl dữ liệu từ nguồn cụ thể, trực quan hóa, làm sạch dữ liệu và xây dựng mô hình và thực hiện dự đoán dữ liệu (dự đoán giá ô tô).

- Kết quả các quá trình trong công việc của các thành viên trong nhóm đóng góp và phù hợp đáng kể cho các quá trình tiếp theo cho các thành viên còn lại.
- Thông qua việc sử dụng 2 mô hình hồi quy để đánh giá, 2 mô hình đều cho ra kết quả khác nhau, nên dễ dàng nhận xét mô hình nào có khả năng ảnh hưởng cao hơn so với mô hình còn lại
- Với 2 mô hình mà nhóm sử dụng là Linear Regression và Random Forest Regressor thì kết quả tốt hơn nghiên về Random Forest Regressor, đối với Linear Regression độ chính xác không được tốt do bị ảnh hưởng bởi sự tuyến tính của dữ liệu

Hướng phát triển:

- Xây dựng thêm các mô hình dự đoán và thử nghiệm thêm các kỹ thuật xử lý dữ liệu khác để giảm sai số.
- Tăng kích thước dataset bằng cách crawl thêm dữ liệu.

6. Tài liệu tham khảo

[1] Linear_regression, https://en.wikipedia.org/wiki/Linear_regression

[2] Feature importance, http://surl.li/chrif

[3] Seaborn, https://seaborn.pydata.org/generated/seaborn.lineplot.html

[4] Random Forest Regression, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[5] BeautifulSoup, Beautiful Soup Documentation — Beautiful Soup 4.4.0 documentation (beautiful-soup-4.readthedocs.io)

[6] Metric MAE, https://en.wikipedia.org/wiki/Mean_absolute_error

[7] Requests_html, Python requests-html - Learn Web scraping (alixaprodev.com)