



# Khoa học dữ liệu

## Đồ án cuối kỳ

# Dự đoán trên tập dữ liệu cars

GV: thầy Trần Trung Kiên

1612855 - Hồng Thanh Hoài  
1612858 - Huỳnh Minh Huân



# Nội dung

- Giới thiệu đồ án
- Thu thập dữ liệu
- Chuyển đổi tập dữ liệu
- Tiền xử lý dữ liệu
- Xây dựng mô hình
- Đánh giá kết quả
- Tổng kết
- Tham khảo



# Giới thiệu đề án

Câu hỏi: dự đoán về giá xe dựa trên các thuộc tính đặc trưng của xe?

- Input: các đặc trưng của xe
- Output: Giá xe dự đoán.

Lợi ích: đem lại các thông tin cần thiết cho người muốn mua xe, muốn tìm hiểu về xe.

Nguồn gốc: nhóm tự nghĩ ra.

# Thu thập dữ liệu







Dữ liệu thu thập trên trang: <https://www.cars-data.com/en/>.

Let`s find your car

Select make (e.g. Ford) ▼

Select model (e.g. Focus) ▼

se by car body

 <u>SUV cars</u>	 <u>convertible cars</u>	 <u>coupe cars</u>	<u>wagon cars</u> > 1.0 L
			<u>Minivan cars</u> 1.0 L
			<u>MPV cars</u> 2.0 L
			<u>Diesel cars</u> 3.0 L
			<u>Hybrid/Electric</u> 4.0 L
			<u>Luxury cars</u> 5.0 L
 <u>hatchback cars</u>	 <u>sedan cars</u>	 <u>Pick-up</u> <i>(coming soon)</i>	<u>all cars</u> 6.0 L



# Thu thập dữ liệu

Dữ liệu thu thập trên trang này hợp pháp (nhóm đã check trước khi crawl)

```
rp = urllib.robotparser.RobotFileParser()
rp.set_url('https://www.cars-data.com/robots.txt')
rp.read()
rp.can_fetch('*', 'https://www.cars-data.com/en/all-cars.html')
```

True

Kiểm tra trên một trang cần crawl:

```
rp.can_fetch('*', 'https://www.cars-data.com/en/audi-rs4-avant-2.9-tfsi-quattro-specs/80342')
```

True



# Thu thập dữ liệu - nội dung dữ liệu

Dữ liệu bao gồm hơn 80000 dòng. (Dữ liệu có tổng cộng hơn 200 loại thuộc tính (liên quan đến bảo hiểm, phụ tùng, ...)).

Có 34 cột (chưa tiền xử lý dữ liệu):

- url: link trên web của xe.
- name: tên của xe.
- 'model': mẫu xe.
- 'brand': thương hiệu xe.
- 'price:' giá xe (\*\*)



# Thu thập dữ liệu - nội dung dữ liệu

Dữ liệu bao gồm hơn 80000 dòng

Có 34 cột (chưa tiền xử lý dữ liệu):

- 'eLabel': energy label.
- 'bodyType': loại thân xe.
- 'length': chiều dài.
- 'height': chiều cao.
- 'width': chiều rộng.
- 'weight': trọng lượng xe.



# Thu thập dữ liệu - nội dung dữ liệu

Dữ liệu bao gồm hơn 80000 dòng

Có 34 cột (chưa tiền xử lý dữ liệu):

- 'weightTotal': tổng trọng lượng.
- 'emissionsCO2': lượng khí thải.
- 'modelDate': năm sản xuất mẫu.
- 'fuelType': loại nhiên liệu
- 'numberOfAxles': số trục.
- 'numberOfDoors': số lượng cửa.





# Thu thập dữ liệu - nội dung dữ liệu

Dữ liệu bao gồm hơn 80000 dòng

Có 34 cột (chưa tiền xử lý dữ liệu):

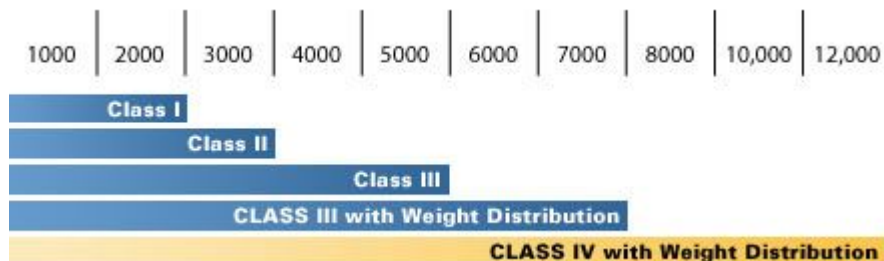
- 'numberOfForwardGears': số gears.
- 'seatingCapacity': không gian chỗ ngồi (số ghế).
- 'vehicleTransmission'
- 'cargoVolume': dung lượng hành hóa.
- 'roofLoad': tải trọng roof.
- 'accelerationTime': thời gian tăng tốc.

# Thu thập dữ liệu - nội dung dữ liệu

Dữ liệu bao gồm hơn 80000 dòng

Có 34 cột (chưa tiền xử lý dữ liệu):

- 'driveWheelConfiguration': cấu hình wheel.
- 'fuelCapacity': dung tích nhiên liệu.
- 'fuelConsumption': độ tiêu hao nhiên liệu.
- 'speed': tốc độ.
- 'payload'
- 'trailerWeight': khối lượng móc nối.

[illegible]

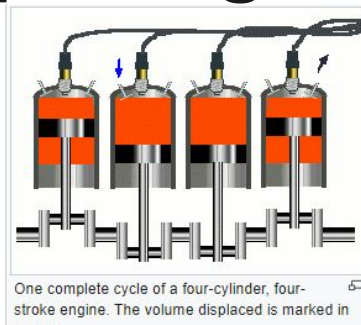
# Thu thập dữ liệu - nội dung dữ liệu

Dữ liệu bao gồm hơn 80000 dòng

Có 34 cột (chưa tiền xử lý dữ liệu):

Nhóm liên quan đến động cơ xe

- 'vEngineType': Loại máy
- 'vFuelType': loại nhiên liệu
- 'vEngineDisplacement':
- 'vEnginePower': công suất
- 'torque': mô men-xoắn



$$\text{Công thức: } T = P * 9.55 / n$$

- Trong đó T là mô-men xoắn trên trục động cơ (Nm)
- P là công suất động cơ điện (kW)
- n là tốc độ động cơ (vòng/phút)



# Thu thập dữ liệu - vấn đề

Dữ liệu có các vấn đề như:

- Dữ liệu vẫn chứa các giá trị thiếu. (xử lý ở bước **tiền xử lý dữ liệu**)
- Một số cột có thể không mang lại ý nghĩa cho việc dự đoán. (xử lý ở bước **tiền xử lý dữ liệu**)
- Phần lớn các cột chưa được định dạng đúng kiểu dữ liệu (xử lý ở bước **chuyển đổi tập dữ liệu**).

# Chuyển đổi tập dữ liệu thô

Tập dữ liệu thô ban đầu các dòng chỉ chứa các giá trị chuỗi

magnum	year	model	length	height	width	weight	weightTotal	emissionsCO2	numberOfAxles	numberOfDoors	numberOfForwardGears	seatingCapacity	cargoVolume	roofLoad	accelerationTime	fuelCapacity	fuelConsumption	speed	payload	trailerWeight	vEngineDisplacement	vEnginePower	torque
42.247	G	hatchback	4159	1415 mm	1763 mm	1375 kg	1375 kg	218 g/km	1999	petrol	2	3	6	5	gearbox	270-1020	1						

Nhóm chia ra 2 loại dữ liệu là **numeration** và **categorization**

- Các cột num:

```
num_cols = ['price', 'length', 'height', 'width', 'weight', 'weightTotal', 'emissionsCO2', 'numberOfAxles',  
            'numberOfDoors', 'numberOfForwardGears', 'seatingCapacity', 'cargoVolume', 'roofLoad',  
            'accelerationTime', 'fuelCapacity', 'fuelConsumption', 'speed', 'payload', 'trailerWeight',  
            'vEngineDisplacement', 'vEnginePower', 'torque']
```



# Chuyển đổi tập dữ liệu thô

Các cột dữ liệu số (ngoài cột **cargoVolume**) có định dạng 'xxx\_unit' (số <khoảng trắng> đơn vị).

- Hàm `cvtFloat` để tách lấy ra được các giá trị số.
- `pandas df` và `series` hỗ trợ `apply function` để gọi hàm trên.

Đối với cột **cargoVolume** có định dạng 'xx - yy unit', nhóm mặc định lấy giá trị yy



# Chuyển đổi tập dữ liệu thô

Các cột categorization:

- Cột `driveWheelConfiguration` không có giá trị lỗi ('N.A.', '-', ...)
- Các cột `bodyType`, `vEngineType`, `vFuelType` có chứa nan (đã được xử lý).
- Cột `eLabel` có chứa các giá trị lỗi, cần được chuẩn hóa. Sau khi chuẩn hóa, dòng thiếu dữ liệu quá nhiều nên cần loại bỏ khi qua bước xử lý.

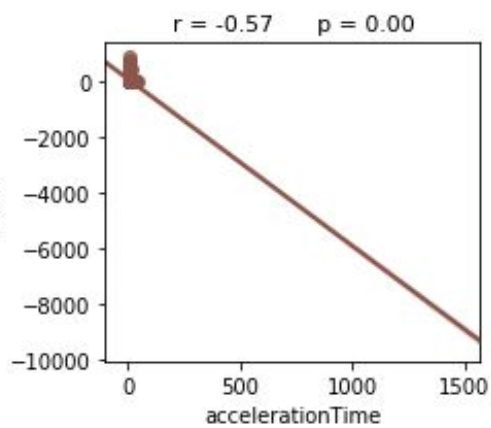
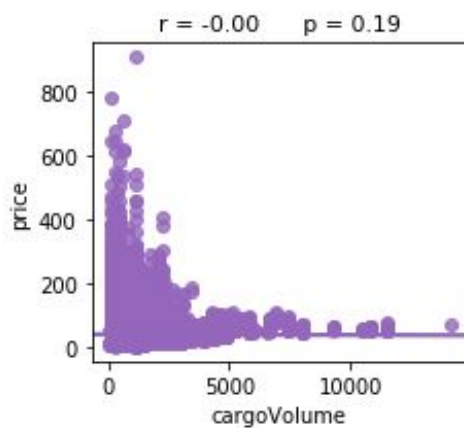
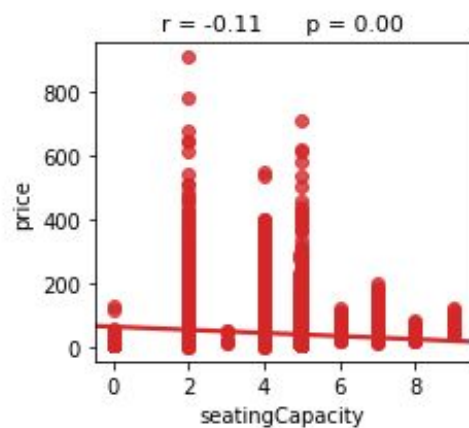
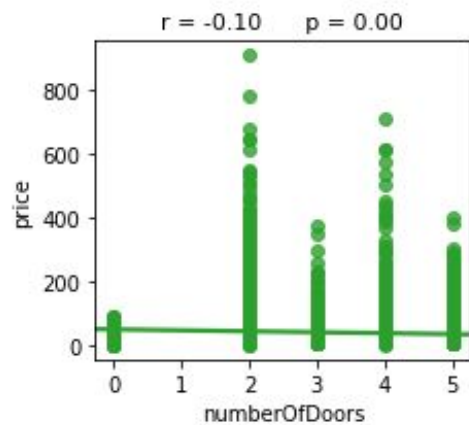
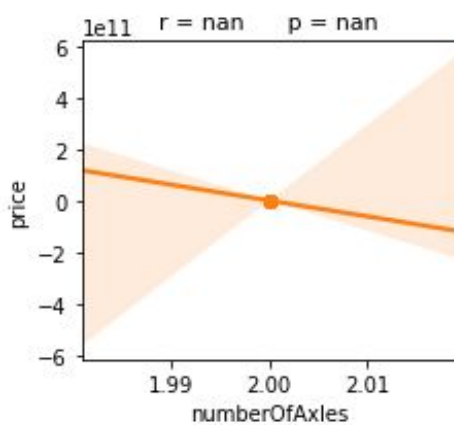
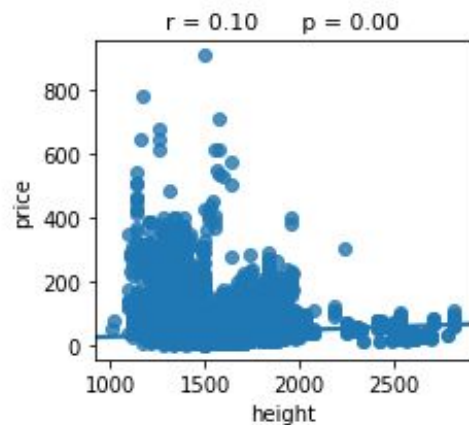


# Tiền xử lý dữ liệu

Xóa bỏ các thuộc tính:

- 'url', 'name', 'model': 'brand' đã thể hiện đủ
- 'weightTotal': trùng với 'weight'
- 'fuelType': trùng với 'vEfuelType'
- 'vehicleTransmission': chỉ có một giá trị duy nhất
- 'modelDate': ảnh hưởng đến kết quả, và với tập test có thể không có giá trị







# Tiền xử lý dữ liệu

Class Multilabel Encoding: Để encode hai thuộc tính 'vEfuelType', 'driveWheelConfiguration'.

```
train.vEfuelType.unique()
```

```
array(['petrol', 'diesel', 'aardgas', nan, 'benzine', 'bio-ethanol',  
      'natural gas', 'LPG', 'aardgas / petrol', 'LPG / petrol',  
      'petrol / bio-ethanol'], dtype=object)
```



# Tiền xử lý dữ liệu

Class ColAdderDropper: Để xóa thuộc tính và gán lại giá trị cho thuộc tính 'branch' dựa vào 'num\_top\_brands'



# Tiền xử lý dữ liệu

- Numerical: Điền giá trị thiếu bằng giá trị trung bình của thuộc tính
- Categorical (single):
  - + Điền giá trị thiếu bằng giá trị phổ biến nhất của thuộc tính
  - + Sử dụng One Hot Encoder
- Categorical (multi): Sử dụng Multi-label Encoder.



# Xây dựng mô hình

- Quá trình
- Các mô hình đã chọn:
  - Neural networks
  - RandomForest



# Xây dựng mô hình - quá trình

Nhóm chọn 2 mô hình được sklearn hỗ trợ là MLPRegressor và RandomForestRegressor.

- Với MLP sau khi chọn được mô hình tốt nhất sẽ dùng mô hình đó để chọn best\_num\_brands.
- Với RFR chọn mô hình tốt nhất dựa trên n\_estimator (“The number of trees in the forest.”).



## Xây dựng mô hình - quá trình

- Ban đầu, nhóm chỉ sử dụng các record không chứa null, kích thước tập dữ liệu khi đó khoảng ~50000 dòng (train: ~40000, validation ~10000).
- Khi đó sử dụng mô hình để huấn luyện đều cho kết quả không tốt (score train ~0.7, score val ~0.6).
- Sau đó, nhóm đã sử dụng nguyên tập dữ liệu ~84000 dòng (train ~65000, validation ~15000), kết hợp tiền xử lý dữ liệu -> huấn luyện hiệu quả hơn.



# Xây dựng mô hình - mô hình neural net

- Sử dụng **MLPRegressor** trong module `neural_network` của `sklearn`.
- Theo nhóm tìm hiểu, mô hình `MLPRegressor` sẽ tối ưu hóa hàm lỗi bình phương (squared-error).
- Các thiết lập của nhóm:
  - `activation`: 'relu'
  - `solver`: 'adam' (theo doc thì với lượng dữ liệu lớn thì nên dùng. Còn tập dữ liệu nhỏ thì dùng 'lbfgs').
  - `learning_rate`: 'adaptive'





# Xây dựng mô hình - mô hình neural net

- Mô hình 2 hidden layers (256, 256, )
- Mô hình 2 hidden layers (512, 512, )

0.9771158431945284 0.749004253783355  
MSE = 227.407  
MAE = 2.91

- Mô hình 3 hidden layers (256, 256, 256, )

0.9642044573025929 0.8159559148084718  
MSE = 166.747  
MAE = 3.092

- Mô hình 3 hidden layers (256, 512, 256, )

0.9753737214003044 0.878517636724576  
MSE = 110.065  
MAE = 2.886



# Xây dựng mô hình - mô hình neural net

- Mô hình 3 hidden layers (512, 256, 512, )

0.981888215514595 0.7352570716057742

MSE = 239.862

MAE = 2.776

- Mô hình 4 hidden layers (256, 512, 512, 256, )

0.9469180582378245 0.8917373042694016

MSE = 98.088

MAE = 3.41



# Xây dựng mô hình - mô hình neural net

- Mô hình 5 hidden layers (256, 512, 512, 128, )

0.968874482026066 0.8829205161438752

MSE = 106.076

MAE = 3.27

- Mô hình 5 hidden layers (256, 512, 512, 512, 256, )

0.9739361561697789 0.8771331410776515

MSE = 111.32

MAE = 2.858



# Xây dựng mô hình - mô hình neural net

- Mô hình 4 hidden layers (256, 512, 512, 256, ) cho kết quả trên tập validation tốt nhất.
- Nhóm dùng mô hình này cố định lại để chọn 'num\_brands' (vì cột

```
full_pipeline.set_params(mlpregressor__verbose=0, mlpregressor__hidden_layer_sizes=(256, 512, 512, 256, ))
nn_train_scores, nn_val_scores = [], []
best_n_brands, nn_best_val_score = 10, -float('inf')
for n_brands in range(5, 95, 5):
    print('...')
    full_pipeline.set_params(coladderdropper__num_top_brands=n_brands)
    train_score, val_score = train_and_val(full_pipeline, train_X, train_y, val_X, val_y)
    if (nn_best_val_score < val_score):
        nn_best_val_score = val_score
        best_n_brands = n_brands
    nn_train_scores.append(train_score)
    nn_val_scores.append(val_score)
    print()
'Finish!'
```



# Xây dựng mô hình - mô hình neural net

- Cho num\_brands với các giá trị 5, 10, 15, ..., 90 (đã có xử lý trong class **coladderdropper**).

```
...  
0.9791494784842625 0.43344014041802026  
MSE = 513.313  
MAE = 2.988
```

```
...  
0.9731883254778363 0.8046131333876603  
MSE = 177.024  
MAE = 2.988
```

```
...  
0.9829765185771713 0.8876575305881531  
MSE = 101.784  
MAE = 2.753
```

```
...
```

- Kết quả:

```
[64]: best_n_brands, nn_best_val_score
```

```
[64]: (80, 0.9084258826395483)
```



# Xây dựng mô hình - mô hình RandomForest

n\_estimators: 1  
0.9718247797364523 0.9478718245603407  
MSE = 47.229  
MAE = 3.163

n\_estimators: 2  
0.9828131031479419 0.9606942242194864  
MSE = 35.612  
MAE = 2.875

n\_estimators: 4  
0.986878391887526 0.9665858439464541  
MSE = 30.274  
MAE = 2.673

n\_estimators: 8  
0.9884801786779545 0.9730351534046627  
MSE = 24.431  
MAE = 2.554

n\_estimators: 16  
0.9903936076513717 0.9762188046877665  
MSE = 21.546  
MAE = 2.502

n\_estimators: 32  
0.9913698175139225 0.9761451846033995  
MSE = 21.613  
MAE = 2.471

n\_estimators: 64  
0.9916120780081737 0.9762780659135604  
MSE = 21.492  
MAE = 2.455

n\_estimators: 128  
0.9919707938386294 0.9762435383099555  
MSE = 21.524  
MAE = 2.449

n\_estimators: 256  
0.9920670454581194 0.9765652539792711  
MSE = 21.232  
MAE = 2.444

n\_estimators: 512  
0.9920278565547928 0.9765225206626313  
MSE = 21.271  
MAE = 2.445



## Đánh giá kết quả

Kết quả model Neural net.

```
validation score did not improve more than 0.0100  
0.9846814518215354 0.9758296985939665  
MSE = 22.37  
MAE = 2.472
```

```
Out[58]: (0.9846814518215354, 0.9758296985939665)
```



# Tổng kết

- Thực hành lại quy trình một bài toán Khoa học dữ liệu.
- Biết được cách sử dụng phương pháp regression cho bài toán dự đoán.
- Biết được MLPRegressor, RandomForestRegressor, ...





# Tham khảo

[RandomForestRegressor — scikit-learn 0.22.1 documentation](#)

[MLPRegressor — scikit-learn 0.22.1 documentation](#)

file BT03-TienXuLy\_ChongOverfit.



**Cảm ơn thầy và các bạn đã theo dõi**