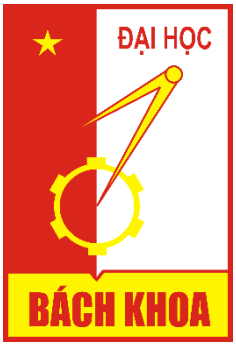
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY



IT4142E PROJECT REPORT

***Topic*: Data Science project for car price prediction**

Professor: **Dr. Than Quang Khoat**

Members of Group :

Index	Student name	Student ID
1	Nguyễn Hoàng Vũ	20190100
2	Đặng Quang Minh	20194796
3	Đào Đức Mạnh	20194794
4	Vũ Hoàng Nam	20194809

Academic Year 2021-2022

Table of contents

PART I: INTRODUCTION TO OUR PROJECT

Nowadays, a great number of customers are seeking a way to purchase the latest model of car. With a wide variety of range in features (number of seats, favorite brands, ….), the customers make a hard way to predict for the price of their favorite models of cars.

Car prediction AI model can facilitate a smarter way to predict the car price based on the features customer desires. That would be meaningful for customer to adjust their needs to make a smarter purchase on their own future car.

PART II: CRAWLING DATA

1. Tools

- **Web scraper**: can extract data from sites with multiple levels of navigation. It can navigate a website on all levels.
- **Scrapy:** An open-source and collaborative framework for extracting the data you need from websites. In a fast, simple, yet extensible way.

2. Crawling process

- Link data for crawling: https://www.cars-data.com/en/.
- Firstly, we crawl all possible car links on the website by using Web scraper. In the end, we got a total of more than 84000 available links.
- After receiving the link for car data, we use the scrapy library with Python in order to get features data of cars in each page source of the website link.

```
# get name
item['name'] = response.xpath("//meta[@property='og:title']").attrib['content']
# get data from json file
table_items = list(map(remove_tags, response.xpath("//tr/td[contains(@class, 'col-6')]").getall()))
```

** craw data by Html tag on the page source*

- Our car data have 37 features with more than 75000 lines of data (have yet preprocessed)
  - URL: link
  - name: name of the car
  - model: car model
  - brand: car brand
  - price: car price (target) ($)
  - eLabel: energy label
  - bodyType: type of car body
  - length: length of the car (mm)
  - height: height of the car (mm)
  - width: width of the car (mm)
  - weight: weight of the car (kg)

- weightTotal: total weight of the car (kg)
- emissionsCO2: the amount of emission (g/km)
- modelDate: year of producing car
- fuelType: type fuel
- numberOfAxles: number of axles
- numberOfDoors: number of doors
- numberOfForwardGear
- seatingCapacity: seating capacity in the car
- vehicleTransmission:
- cargoVolume: cargo capacity (l)
- roofLoad: roof load
- accelerationTime: acceleration time
- driveWheelConfiguration: configuration wheel
- fuelCapacity: fuel capacity
- fuelConsumption: fuel consumption (l/100km)
- weight:
- speed:
- payload:
- trailerWeight:
- vEengineType: Engine Type
- vEfuelType: Fuel Type
- vEenginePower: Engine Power (KW)
- vEengineDisplacement: Cylinder capacity
- torque:
- curbWeight: curb weight (kg)
- engineCapacity: engine capacity (cc)

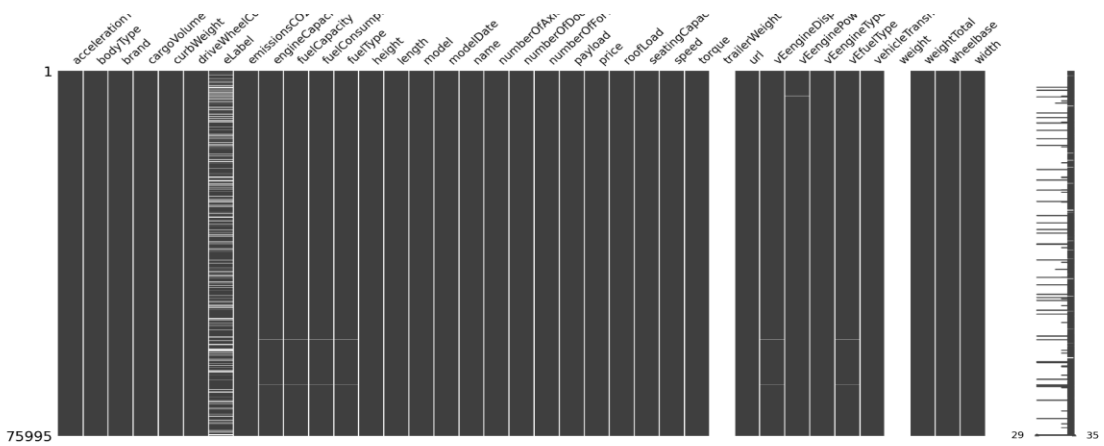
3. Difficulty

The number of lines of data is large so it takes nearly a day to crawl complete data from the website. Moreover, since the website is quite old and hasn't been updated, many link websites were empty, therefore the number of car data is less than the number of links by 5000 lines.

PART III: CLEANING DATA

1. Why need clean data

- Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset.
- In our crawled dataset, there are many features that have not been in the incorrect format or contained unnecessary information such as unit, URL, and some characters like $, /, …
- Besides redundant information, we also met many missing data or NaN values caused by objective conditions as the stale website



- Proper data cleaning will save time and make our model more efficient, therefore, the cleaning process is inevitable

2. Methods of handling missing data

- Handling the missing values is one of the greatest challenges faced by analysts because making the right decision on how to handle it generates robust data models. We will introduce some methods that we have researched.
    - **Deleting rows**: This method is commonly used to handle null values. Here, we can either delete a particular row if it has a null value for a particular feature and a particular column if it has more than 70-75% of missing values.
        - Pros:
            + Complete removal of data with missing values results in a robust and highly accurate model
            + Deleting a particular row or a column with no specific information is better since it does not have a high weightage
        - Cons:
            + Loss of information and data
            + Works poorly if the percentage of missing values is high (say 30%), compared to the whole dataset
    - **Replacing with Mean/Median/Mode:** This strategy can be applied to a feature that has numeric data like the age of a person or the ticket fare. We can calculate the mean, median, or mode of the feature and replace it with the missing values. This is an approximation that can add variance to the data set. But the loss of the data can be negated by this method which yields better results compared to the removal of rows and columns.
        - Pros:
            + This is a better approach when the data size is small
            + It can prevent data loss which results in the removal of the rows and columns
        - Cons:
            + Imputing the approximations add variance and bias
            + Works poorly compared to another multiple-imputations method
    - **Using Algorithms Which Support Missing Values:** We can use machine learning algorithms such as KNN or Random Forest based on their characteristics.
        - Pros:

        + Does not require the creation of a predictive model for each attribute with missing data in the dataset
        + Correlation of the data is neglected
    - Cons:
        + Is a very time-consuming process and it can be critical in data mining where large databases are being extracted
        + Choice of distance functions can be Euclidean, Manhattan, etc. which is do not yield a robust result

3. Cleaning process description + Encode categorical columns

- Firstly, we remove two null columns in crawled data (trailerWeight, weight) and unwanted observations from the dataset, including duplicate observations or irrelevant observations. We also remove some outlier values in each column that we found incorrect and replace them with Nan values
- The most difficult part of the cleaning process is handling missing data, which takes much time to explore and figure out the suitable method for our dataset.
- Since the null data after first cleaning account for more than 30% of the total data, the deleting rows method becomes unsuitable to build a rich dataset. With the categorical features, we realized most of them contain a value with the number of appearances overwhelming to the rest, so we decide to fill the NaN value by the most frequent value. After that, we also convert them to numeric based on their order in the alphabet. With the numerical attributes, we decide to use KNN in order to fill our missing data by taking the majority of the K nearest values. To be unbiased from other features, before filling by KNN, we normalize all features that do not have non-null values.
- The dataset description detail after the cleaning process is: 30 features and 75824 lines of data

PART IV: EXPLORATION DATA ANALYSIS

1. General analysis

After cleaning the data phase, it is necessary for analyzing the data which have already been selected. First, we find out how many columns of data each column consists of, how many distinct values, and the main type of each column.

Categorical	bodyType, brand, driveWheelConfiguration, eLabel, fuelType, model, name, url, vEengineType, vEfuelType, vehicleTransmission.
Numerical	accelerationTime, cargoVolume, curbWeight, emissionsCO2, engineCapacity, fuelCapacity, fuelConsumption, height, length, modelDate, numberOfAxles, numberOfDoors, numberOfForwardGears, payload, price, roofLoad, seatingCapacity, speed, torque, trailerWeight, vEengineDisplacement, vEenginePower, weight, weightTotal, wheelbase, width.

As we can see in the above figure, most of the columns are numerical data (type of int64 and float64). There are only 11 fields of categorical data (object type). Hence, our job is basically to find out the hidden feature of each column and the relationship between each pair of that column with the target once (the "Price" column) and the correlation between each of them.

Then we would list all the number of non-unique values of each column

```
accelerationTime            227
bodyType                     10
brand                        89
cargoVolume                1043
curbWeight                 1599
driveWheelConfiguration       3
eLabel                        7
emissionsCO2                381
engineCapacity              768
fuelCapacity                 97
fuelConsumption             194
fuelType                      9
height                      645
length                     1229
model                      1605
modelDate                    51
name                      60546
numberOfAxles                 1
numberOfDoors                 4
numberOfForwardGears          8
payload                     822
price                     26125
roofLoad                     38
seatingCapacity               8
speed                       225
torque                      472
trailerWeight                 0
url                       75995
vEengineDisplacement        768
vEenginePower               369
vEengineType                  4
vEfuelType                    9
vehicleTransmission           5
weight                        0
weightTotal                1003
wheelbase                   573
width                       424
dtype: int64
```

2. Detailed analysis

2.1. Numerical value

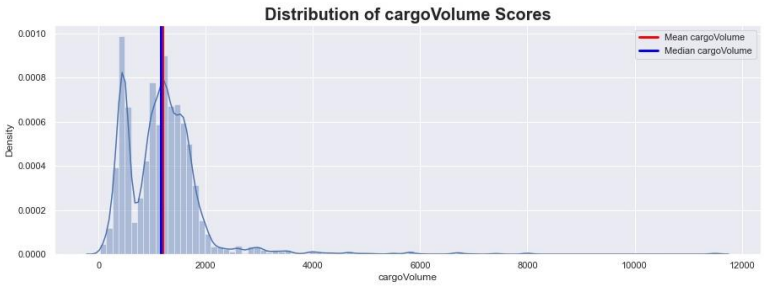
Since this dataset is mainly constructed with various numerical data, detailed analysis of this value could bring back lots of useful information to overview this data. The maximum value, minimum value, and the distribution of each column are what we are concerned about. A part of this analysis is shown below.

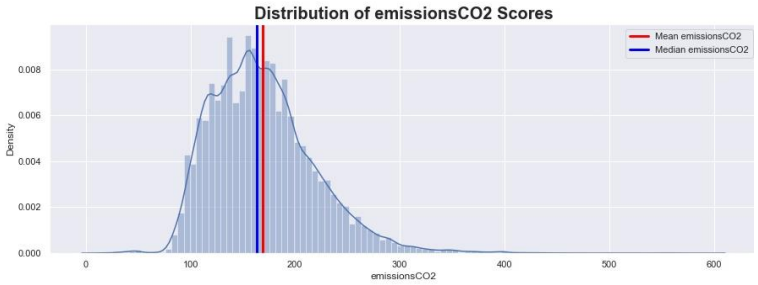
	accelerationTime	cargoVolume	curbWeight	emissionsCO2	engineCapacity	fuelCapacity	fuelConsumption	height	length	modelDate
count	73763.000000	74946.000000	75769.000000	61705.000000	75818.000000	75815.000000	74862.000000	75925.000000	75974.000000	75995.000000
mean	10.652689	1173.072786	1354.292336	163.504043	1967.117624	59.360206	7.004242	1495.500744	4436.188209	2004.557063
std	2.957012	726.498829	301.391706	49.706530	705.041881	11.835721	4.236233	138.842303	385.509466	9.329384
min	2.500000	31.000000	474.000000	12.000000	599.000000	9.000000	0.600000	1012.000000	339.000000	1969.000000
25%	8.700000	570.000000	1150.000000	127.000000	1581.000000	51.000000	5.400000	1421.000000	4222.000000	1999.000000
50%	10.500000	1165.000000	1342.000000	155.000000	1948.000000	60.000000	6.700000	1458.000000	4483.000000	2007.000000
75%	12.300000	1510.000000	1515.000000	190.000000	1999.000000	66.000000	8.200000	1512.000000	4702.000000	2012.000000
max	35.000000	11500.000000	2921.000000	595.000000	8277.000000	208.000000	999.900000	2815.000000	6967.000000	2019.000000

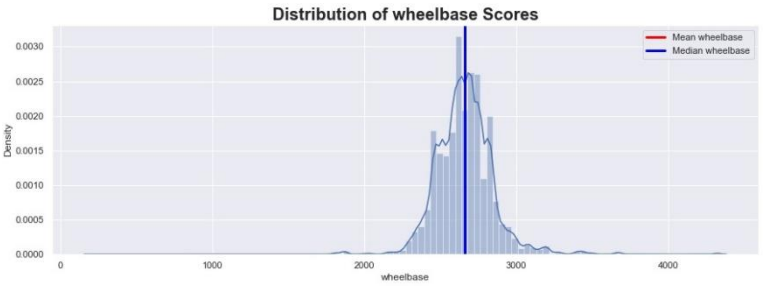
A large value with various means and standard deviation with each column is shown as one of the typical features. To explore the covariance between price and other numerical attributes, we can use a scatter plot. As can be seen from the following graphs, for each feature, the relationship to price is distinct. To determine the relationship towards price, we should see the following joint plot. Except for torque, vEengineDisplacement, and vEenginePower, the other features have a weak relation towards the price. In other words, the torque, vEengineDisplacement, and vEenginePower can be used for evaluating with higher coefficients to predict the price.

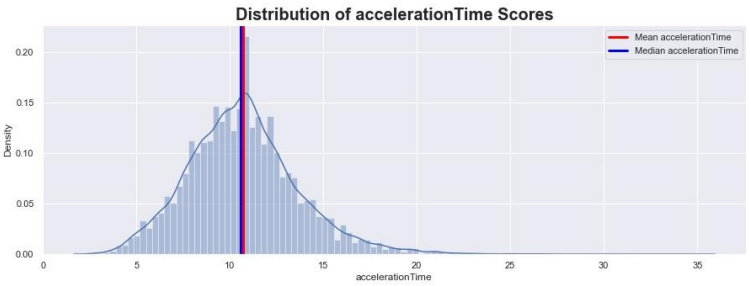
2.1.1. Explore Skewness of Data

First of all, we should explore the skewness of numerical features in our data. Most of the data in each feature have its skewness shifted to the left side means it is positive skewness. They are all features except for features in black_attr, which are denoted for categorical data. Overall, the distribution of these data is approximately symmetric, which infers that there are only a small number of outliers in these features. Outliers are distributed in some features such as height, emissionsCO2, roofLoad, speed, torque, vEenginePower. Some figures have their distribution in Gauss form such as cargoVolume, engineCapacity, fuelCapacity that also required careful processing before input to the model (binormal distribution). In short, numerical attributes in this data set can be used efficiently. There are some sample images for exploring skewness of data. You can see all images in EDA/Image directory.





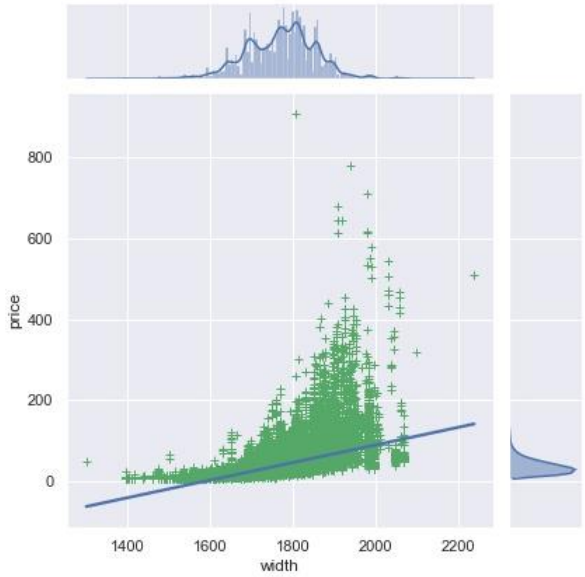


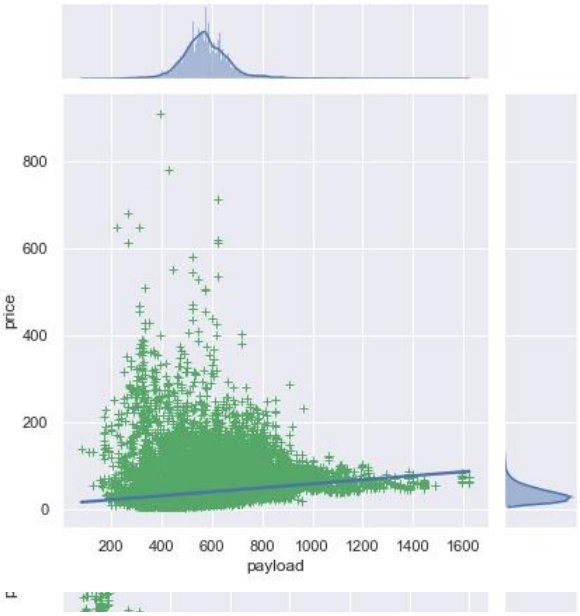


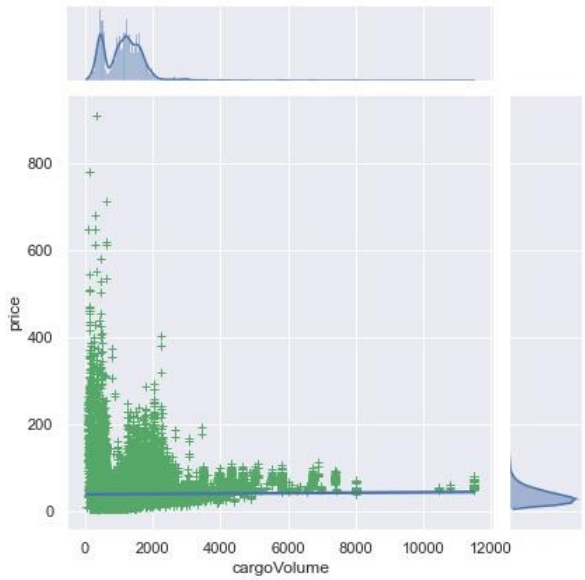
2.1.2. Relationship with target output

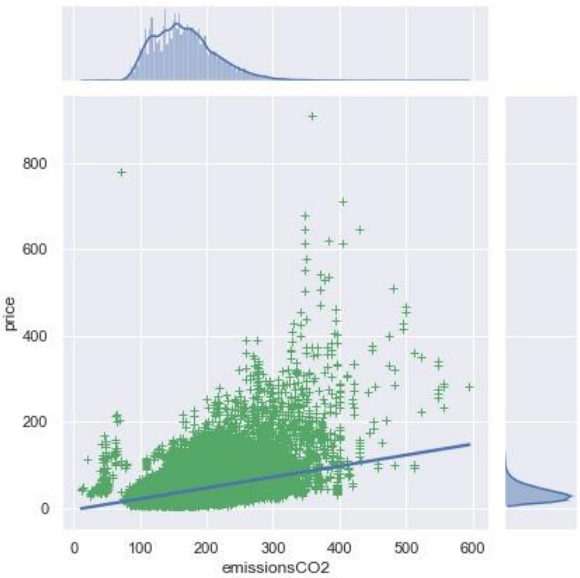
To explore the covariance between price and other numerical attributes, we can use a scatter plot. As can be seen from the following graphs, for each feature, the relationship to price is distinct. Moreover, some of the features show a good distribution towards prices, such as engineCapacity, payload, torque, vEenginePower, and weightTotal.

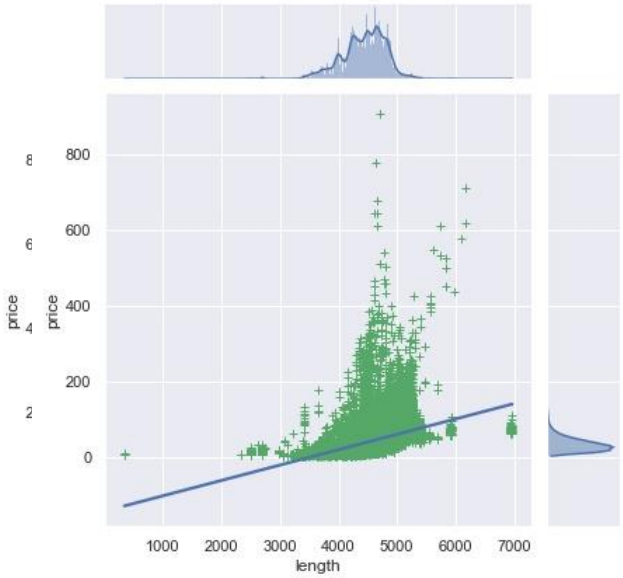
To determine the relationship towards price, we should see the following joint plots. Except for engineCapacity, payload, torque, and vEenginePower, the other features have a weak relation towards price feature. In other words, the torque, vEngineDisplacement, and vEenginePower can be used for evaluating with higher coefficients to predict the price. Moreover, some features can be eliminated due to bad distribution towards price such as fuelComsumption, height, length, and width.

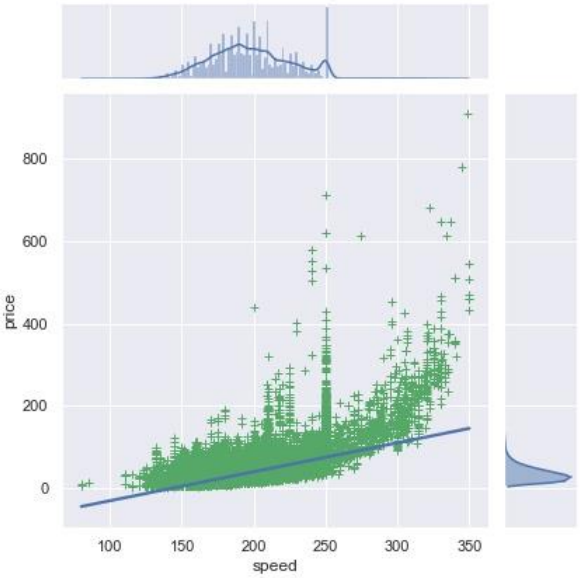

price
width


price
payload


price
cargoVolume


price
emissionsCO2


price
length


price
speed

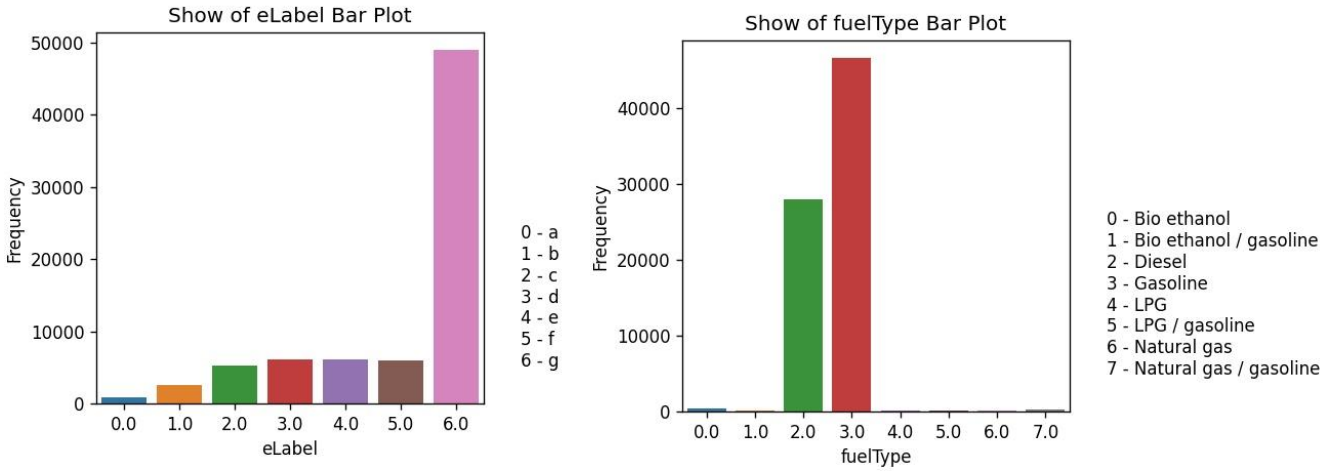
2.2. Categorical value

2.2.1. Barplot

Next, we will explore the other categorical features in our dataset. We only show some sample images in EDA/Image.

First of all, we can see from the count plot for each categorical feature. For intuitive visualization, all the bar plots are represented with the top 10 most significant values of that feature. If the number of values is less than 10 features, all the values will be represented on the barplot. We can see that some features have a high bias on one or two values of that feature. For more detail,

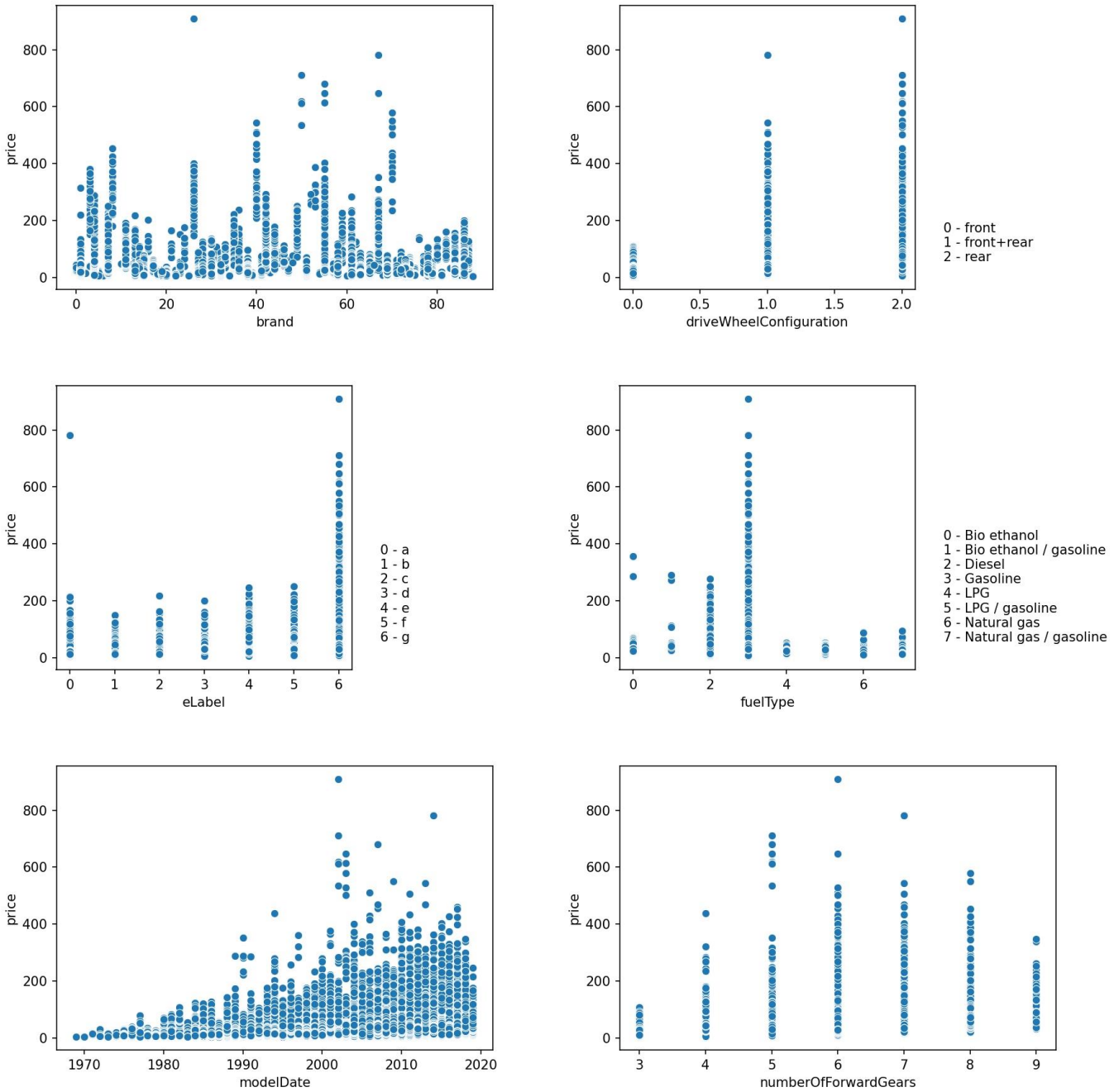
- driveWheelConfiguration is mostly front type
- eLabel is mostly type g
- seatingCapacity is mostly 5 and 4 seats
- fuelType is mostly Diesel and Gasoline
- roofLoad is mostly 75 and 100



2.2.2. Scatter plot

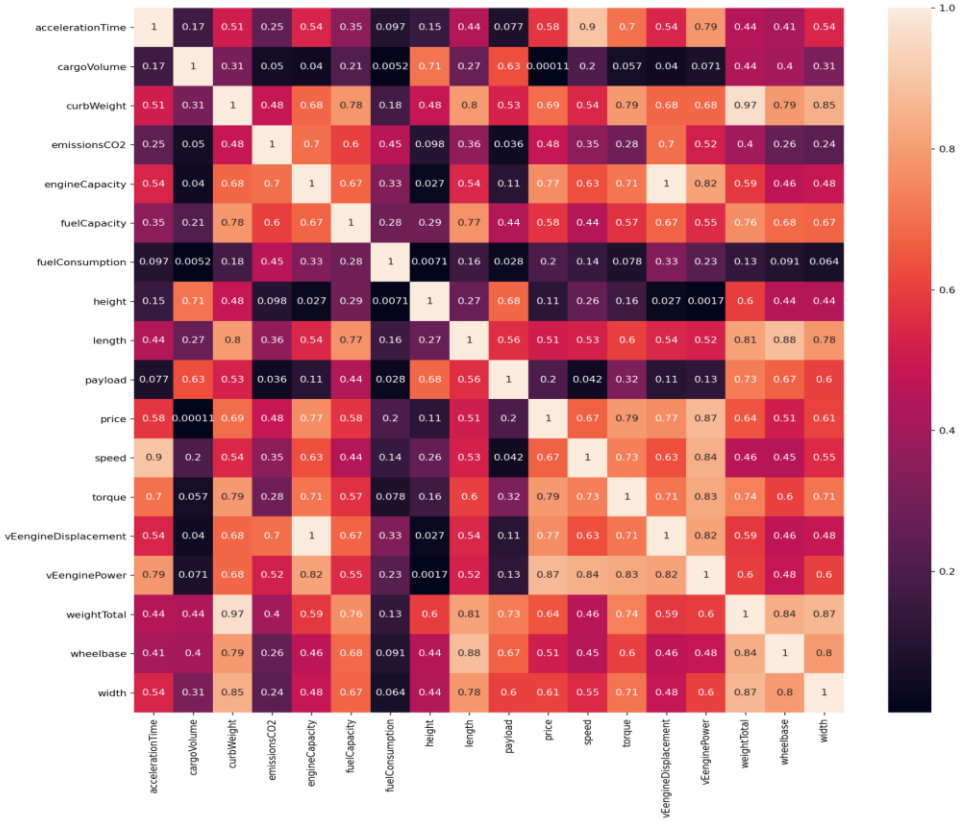
Finally, to prove that conclusion, we will use scatter plot to show the relationship between categorical features and price. Except for the following features, all the other features are suitable for evaluating the price of the car, which poses a considerable effect on the target output.

- driveWheelConfiguration: Front and front-and-rear cover all the prices. The mutual part lies in the price range of less than $200.
- eLabel: Engine type g covers all the high price ranges except for the low range price of less than $200.
- fuelType: most cars are Gasoline, particularly those cars with a price higher than $300.



2.2.3. Heatmap plot

From the above heatmap, it can be seen evidently that some correlation values between some features of this dataset are very high. To our knowledge, if some features having the correlation value between them are big, they can be considered as the same in the dataset. Those same attributes may make the dataset harder for the model to learn.



PART V: SOME ALGORITHM FOR BENCHMARK

In this part, we present some algorithms to benchmark our car dataset. All algorithms in this report are implemented in sklearn, xgboost and catboost libraries.

1. The basic idea of each algorithm.

1.1. Linear Regression

This is a simple approach for the regression problem. The main idea of this algorithm is to fit the observations of all variables into the hyperplane relationship between them.

1.2. KNN Regression

K nearest neighbor is a lazy algorithm that is used for handling the classification problem. However, we also can use the KNN algorithm in the regression problem. The basic idea of this algorithm for the regression problem is to take the mean or median of the targets of K nearest neighbor of the considered point to be the predicted target of that point.

1.3. Random Forest algorithm

This is an ensemble machine learning algorithm that can be used for both classification and regression. In the regression problem, Random Forest will take the mean value of multiple decision trees to get the final prediction.

1.4. Some other algorithms

Besides using Linear Regression, KNN, Random Forest, we also try to use some Boosting algorithms such as XGB boost, LGBM boost, Cat boost algorithm. However, we do not go too deep in those algorithms. We just read the simple idea of the Boosting algorithm. To our knowledge, the basic idea of the Boosting algorithm is to use the combination of multiple weak models. Then, training weak learners sequentially, each will try to correct its previous model.

2. Data preparing for training process

2.1. Drop some similar features by using the high correlation filter

As we said in the data exploring part, there are some big correlation values in the correlation matrix. This means that there are some similar features in our dataset. Those features may make the dataset harder for our benchmark model to learn. With the goal to drop some same attributes, we have created a drop_column function with a threshold value.

```
def drop_column(data, cut = 0.9) :
    # Get correlation matrix and upper triangle
    corr_mtx = data.corr().abs()
    avg_corr = corr_mtx.mean(axis = 1)
    # up is a upper triangle
    up = corr_mtx.where(np.triu(np.ones(corr_mtx.shape), k=1).astype(np.bool))
    # list of drop col
    drop = list()
    for row in range(len(up)-1):
        col_idx = row + 1
        # row != col
        for col in range (col_idx, len(up)):
            if corr_mtx.iloc[row, col] > cut:
                # We choose attribute have its mean of corr larger
                if avg_corr.iloc[row] > avg_corr.iloc[col]:
                    drop.append(row)
                else:
                    drop.append(col)
    # Make unique for element in drop
    drop_set = list(set(drop))
    # dropcols_idx = drop_set
    dropcols_names = list(data.columns[[item for item in drop_set]])
    return dropcols_names
```

In order to find the best threshold value, we will try to run the threshold from 0,5 to 1 with the value of step which is equal to 0,01. Then, we run the Linear Regression for finding the best threshold. Finally, the best threshold value is 0.96 which will drop two columns whose names

are "vEengineDisplacement" and "curbWeight". After dropping those features, our dataset contains 27 fields which consist of 26 features and 1 target.

2.2. Splitting our dataset into train, test sets

We use a random split method for splitting our dataset into train and test sets with the ratio of 70:30. Moreover, in the train process, we will use the random split method 10 times and take the average results.
Finally, we have 53076 car records for the train set and 22748 for the test set.

3. Training process

After preparing data, we will use those above sets to train our above algorithm.

3.1. Normalization

As we said in the data cleaning part, we have converted all categorical columns into numerical columns. We use StandardScaler which is already in the sklearn library. This normalization will help to convert the distribution of our dataset to normal distribution.

```
# feature scale the X_train and X_test values
norm = StandardScaler().fit(X_train)
# transform training data
X_train = norm.transform(X_train)
# transform testing data
X_test = norm.transform(X_test)
```

For the label of each car records, we use log transform to make the true value smooth.

```
# creating X and y variables
X = df.drop('price', axis=1)
# price column
y = np.log(df['price'])
# y = df['price']
```

3.2. Training part

After normalizing data, we will train our model with train and test sets. Each model will be trained 10 times with the random split method for splitting our dataset. Finally, we will average the results to get the final result of each model.

3.3. Loss function

We use root mean squared error as the loss function in order to evaluate the efficiency of our model and r2 score.

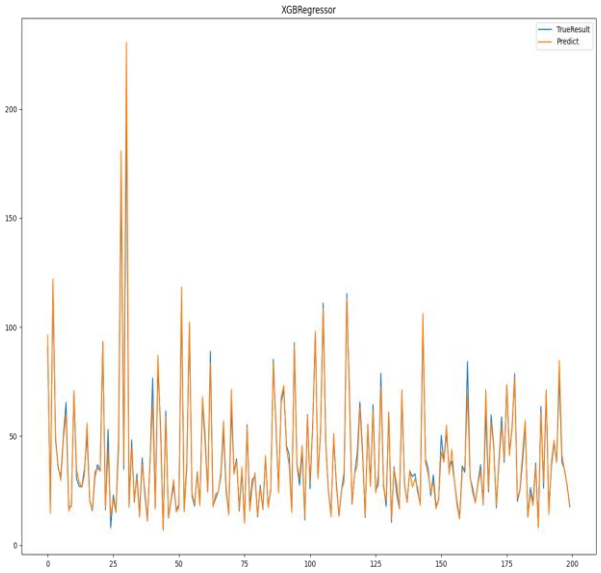
$$\text{RMSE loss} = \sqrt{\frac{1}{N} \times \sum_{i=1}^{N} \left(y_{i\_true} - y_{i\_pred}\right)^2}$$

$$R2 = 1 - \frac{\sum_{i=1}^{N}(y_{i\_true} - y_{i\_pred})^2}{\sum_{I}^{N}(y_{i\_true} - \bar{y})^2}$$

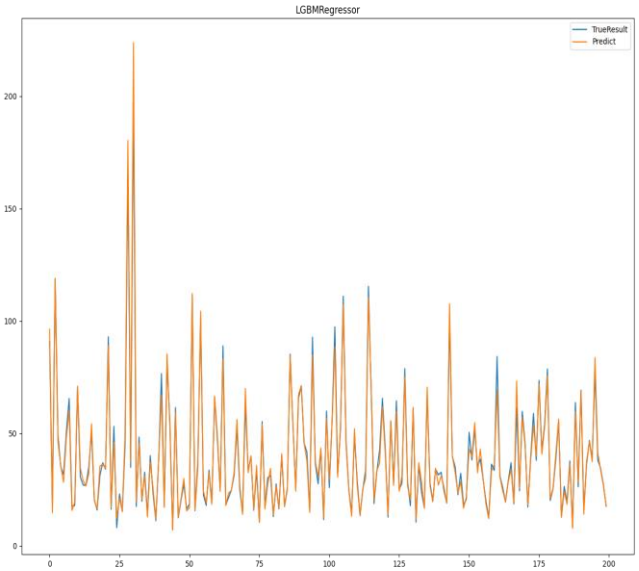
4. Results

Model	RMSE	R2_SCORE
LinearRegression	11.31	0.87
KNeighborsRegressor	7.90	0.93
LGBMRegressor	6.01	0.96
XGBRegressor	6.05	0.96
RandomForestRegressor	6.19	0.96
CatBoostRegressor	7.19	0.94

XGB Boosting



LGBM Boosting



Test result on samples with size 200 on two best models (XGB, LGBM). Blue is for the true value and orange is for the prediction.

PART VI: MEMBER CONTRIBUTION

Member name	Task	Review by
Đào Đức Mạnh	• Clean data • Explore data • Report, slide	All member

Đặng Quang Minh	• Crawl data • Algorithms • Report, slide	All member
Nguyễn Hoàng Vũ	• Crawl data • Algorithms • Report, slide	All member
Vũ Hoàng Nam	• Clean data • Explore data • Report, slide	All member

PART VII: REFERENCES

This is some references which we use in this project

- https://www.jigsawacademy.com/popular-regression-algorithms-ml/
- https://levelup.gitconnected.com/random-forest-regression-209c0f354c84
- https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/
- https://viblo.asia/p/cac-ky-thuat-dimensionality-reduction-OeVKB98A5kW
- https://towardsdatascience.com/boosting-algorithms-explained-d38f56ef3f30