



# Market Prediction of Used Car on eBAY

0656033 許金賢 資科工所

0656620 盧俊言 多工所

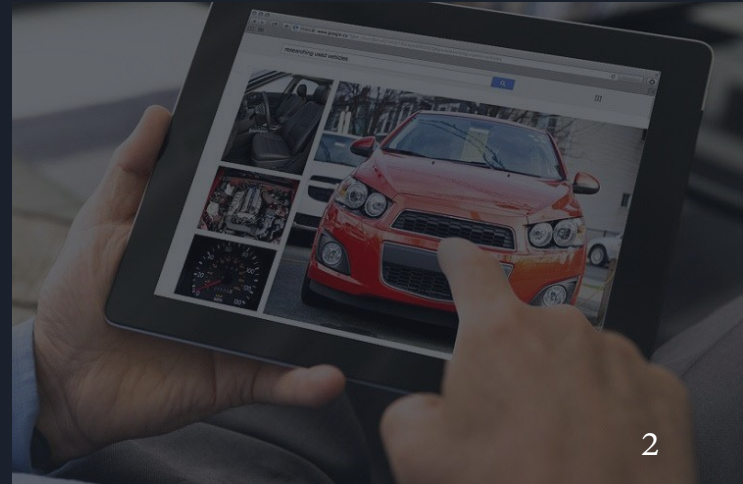
0656622 林子皓 多工所

# Outline

- ❖ Introduction
- ❖ Problem definition
- ❖ Related works
- ❖ Challenges or importance of this work
- ❖ Dataset description
- ❖ Methods
- ❖ Experimental results
- ❖ Issues
- ❖ Other efforts on this project
- ❖ Conclusions and future works
- ❖ Job description
- ❖ References

# Introduction

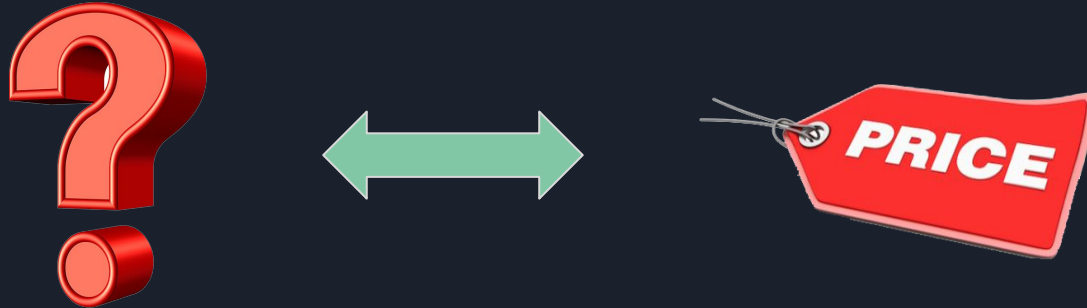
- More and more people tend to get information of products from online e-commerce platforms
- Same phenomenon applies to the used car market, in which potential customers first search online for prices and other information & discussion of the car they have interest.





# Introduction

Our purpose: Help people get a clearer knowledge and understanding of the used car market in their hands.





# Problem definition

## 1. Data Observation

Before doing predictions, we want to observe the visualization outputs to find some trend & relation between a specific attribute and *price*.

## 2. Model Prediction

Find  $f()$  that minimize  $E((price\_true - f(attr))^2)$  for each column.

- For real-valued fields: find first-order approximation ( $y=ax+b$ ) .  
i.e. Find the Correlation coefficient and mean between *price* and specific attributes (real-valued fields).
- For discrete-valued fields: we only use mean.



## Related works

Rayid Ghani (2005) [1]

- Collecting data from online auctions and use several classification algorithms to predict the probable-end prices of online auction items.
- Describing the feature extraction and selection process, and several machine learning formulations of the price prediction problem.



## Related works

Pudaruth (2014) [2]

- Use different techniques like multiple linear regression analysis, K-nearest neighbors, Naïve Bayes and decision trees to investigate the application of supervised machine learning techniques to predict the price of used cars.
- Main limitation: low number of records have been used.



# Related works

Chuanacan, Lulu, and Cong (2017) [3]

- Results:
  - Random forest has a stable but not ideal effect in price evaluation model for a certain car make
  - It shows great advantage in the universal model compared with linear regression.
  - Random forest is an optimal algorithm when handling complex models with a large number of variables and samples
  - Random forest shows no obvious advantage when coping with simple models with less variables.





# Challenges of this work

1. Data are chaos
2. There are many kinds of value in some discrete features. We use one hot encoding to solve it. However, one hot encoder expands dimension too much, so it slows down the speed of model training.



# Importance of this work

*Information asymmetry* is a common problem encountered by consumers browsing online trading platforms, and same problem occurs on online used cars platforms.

Through our work, we want to let consumers know what is the reasonable price for a used car with certain conditions.



# Dataset description

The fields used in our project are below:

- ❖ **price** - The price on the ad to sell the car
- ❖ **vehicleType** - SUV, Coupe...
- ❖ **yearOfRegistration** - At which year the car was first registered
- ❖ **gearbox** - Auto / Manual
- ❖ **powerPS** - Vehicle power in PS (no need to convert to Horse Power because 1HP=1.015PS)
- ❖ **model** - Similar to name, but without brand and engine displacement
- ❖ **monthOfRegistration** - At which month the car was first registered
- ❖ **fuelType** - Diesel / Benzin(gasoline)
- ❖ **brand** - Volkswagen, Audi, BMW...
- ❖ **notRepairedDamage** - If the car has a damage which is not repaired yet

Source: <https://www.kaggle.com/orgesleka/used-cars-database> [4]



# Data Preprocessing

1. First, inspect each column for itself and remove undesired rows.
2. And then remove the rows that have missing value.

# Data Observation

1.



2.



3.



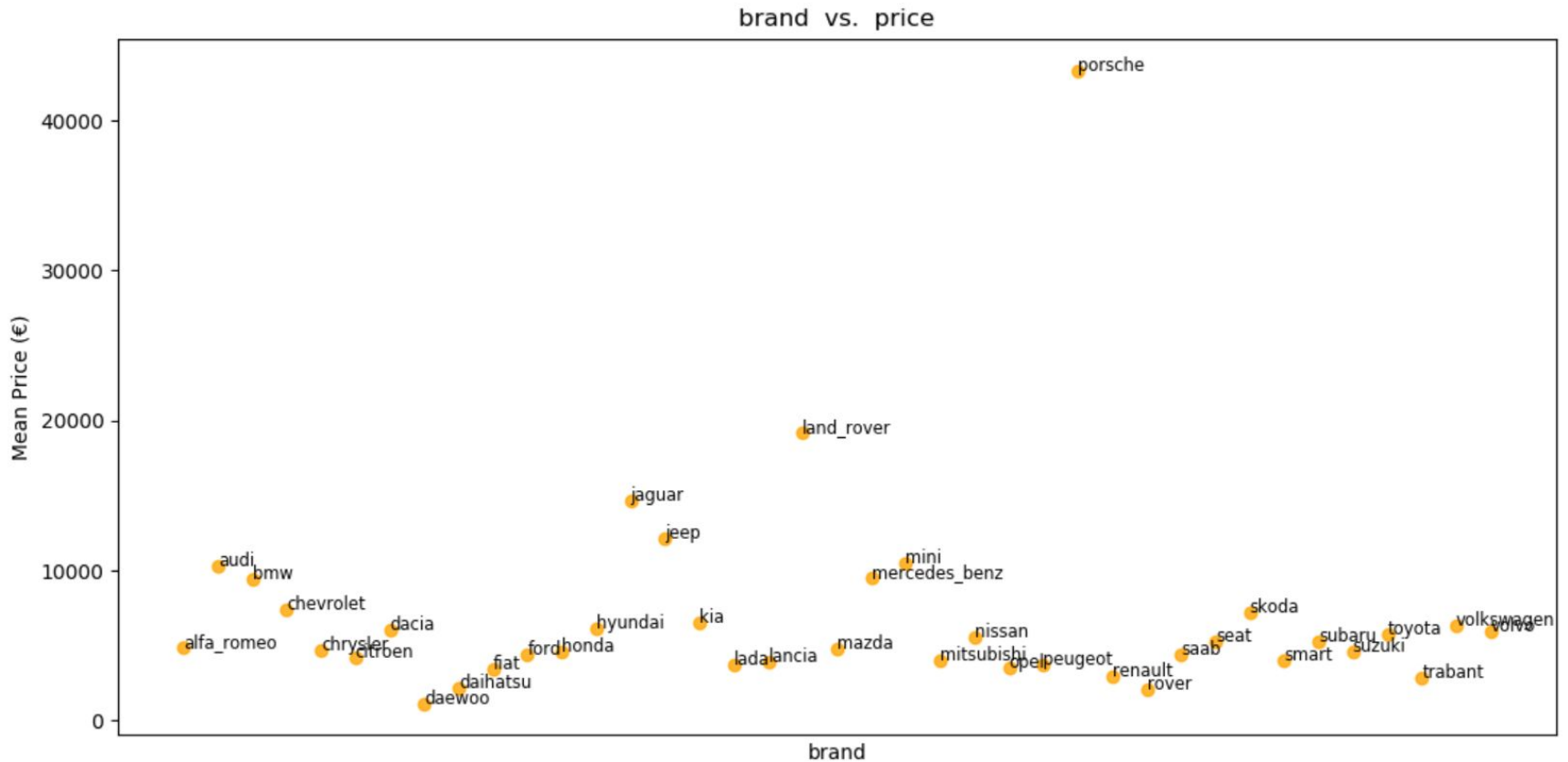
4.



VS.



# Data Observation - *brand* vs. *price*

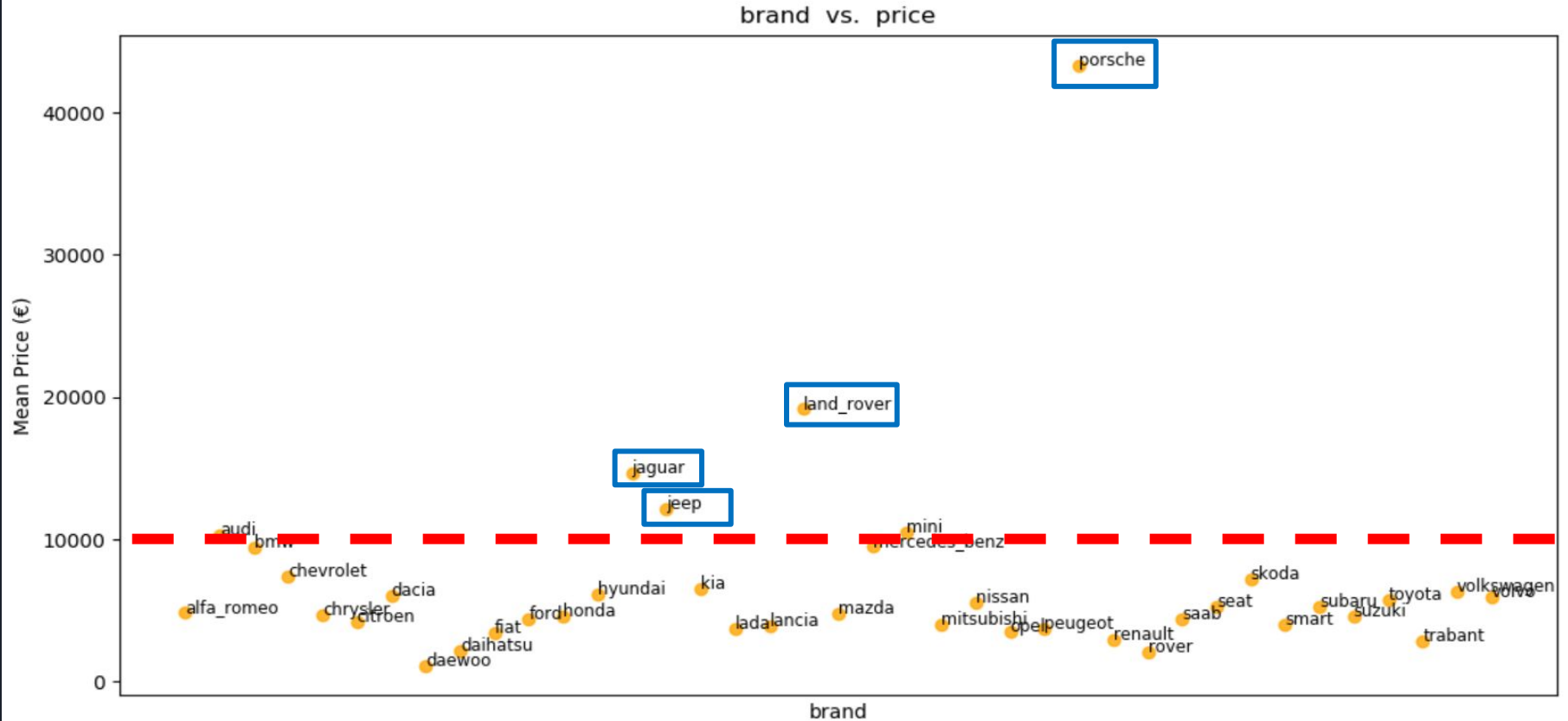




## Data Observation - *brand* vs. *price*

- 39 brands in 247367 data
- Porsche has much higher price than other brands.
- Most of used cars are lower than 10000
  - the others are famous for Sports car and Sport Utility Vehicle.

# Data Observation - *brand* vs. *price*



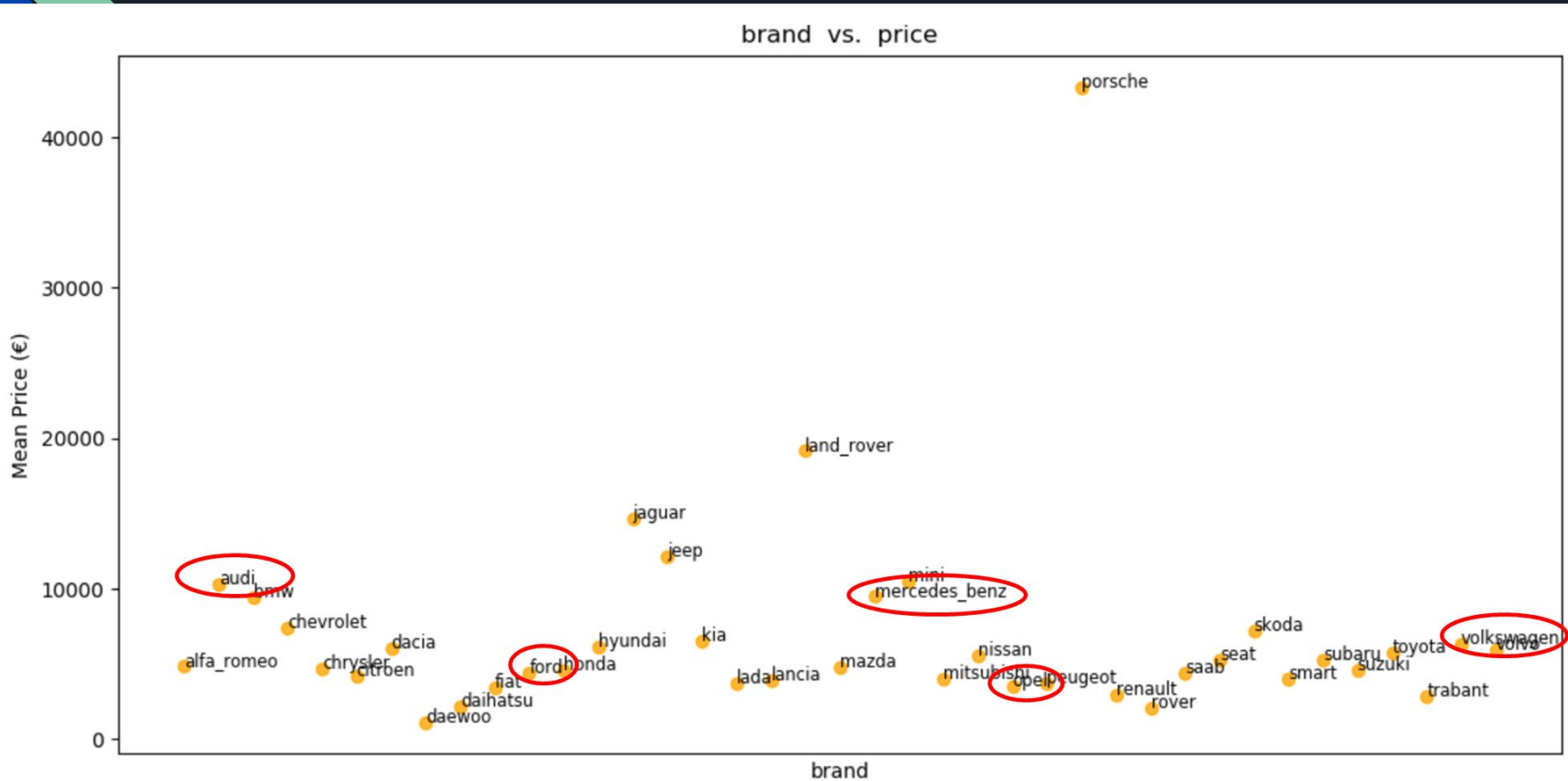


# Data Observation - *model* (in the same brand) vs. *price*

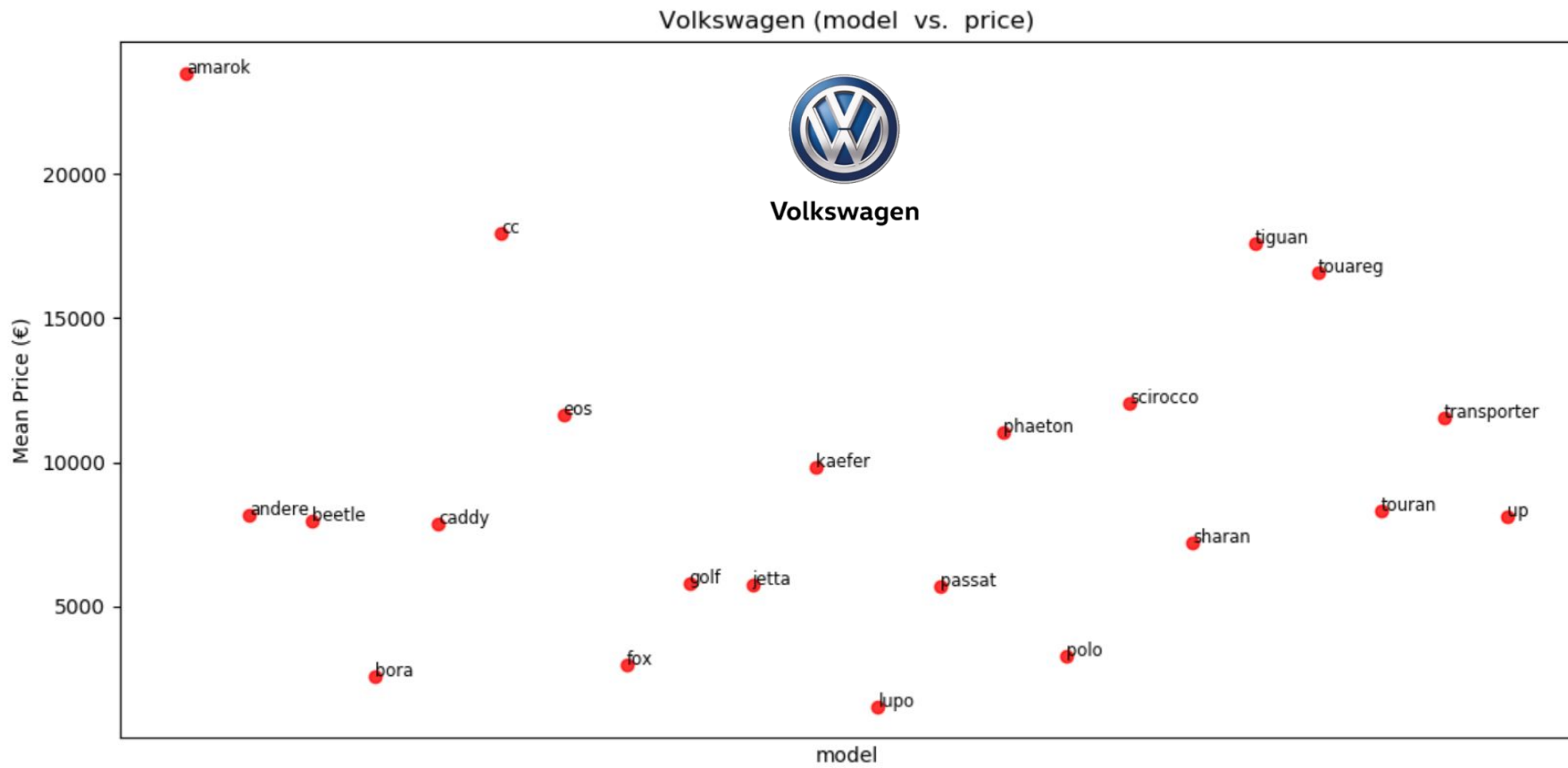
- 250 models in 36 brands
- We choose the brand which has larger than 10 models.
  - Volkswagen: 22
  - Ford: 14,
  - Mercedes Benz: 18
  - Audi: 16
  - Opel: 16



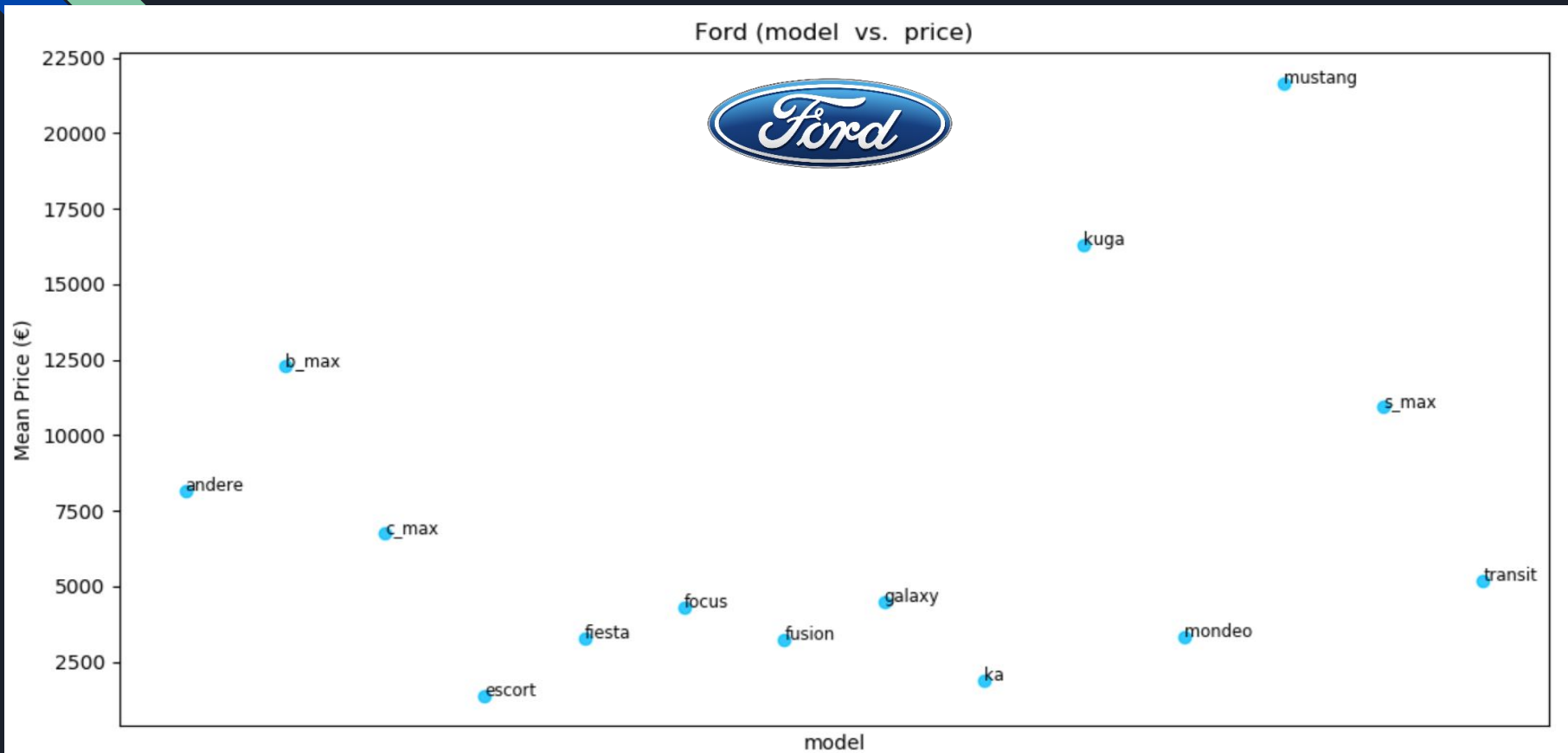
# Data Observation - *model* (in the same brand) vs. *price*



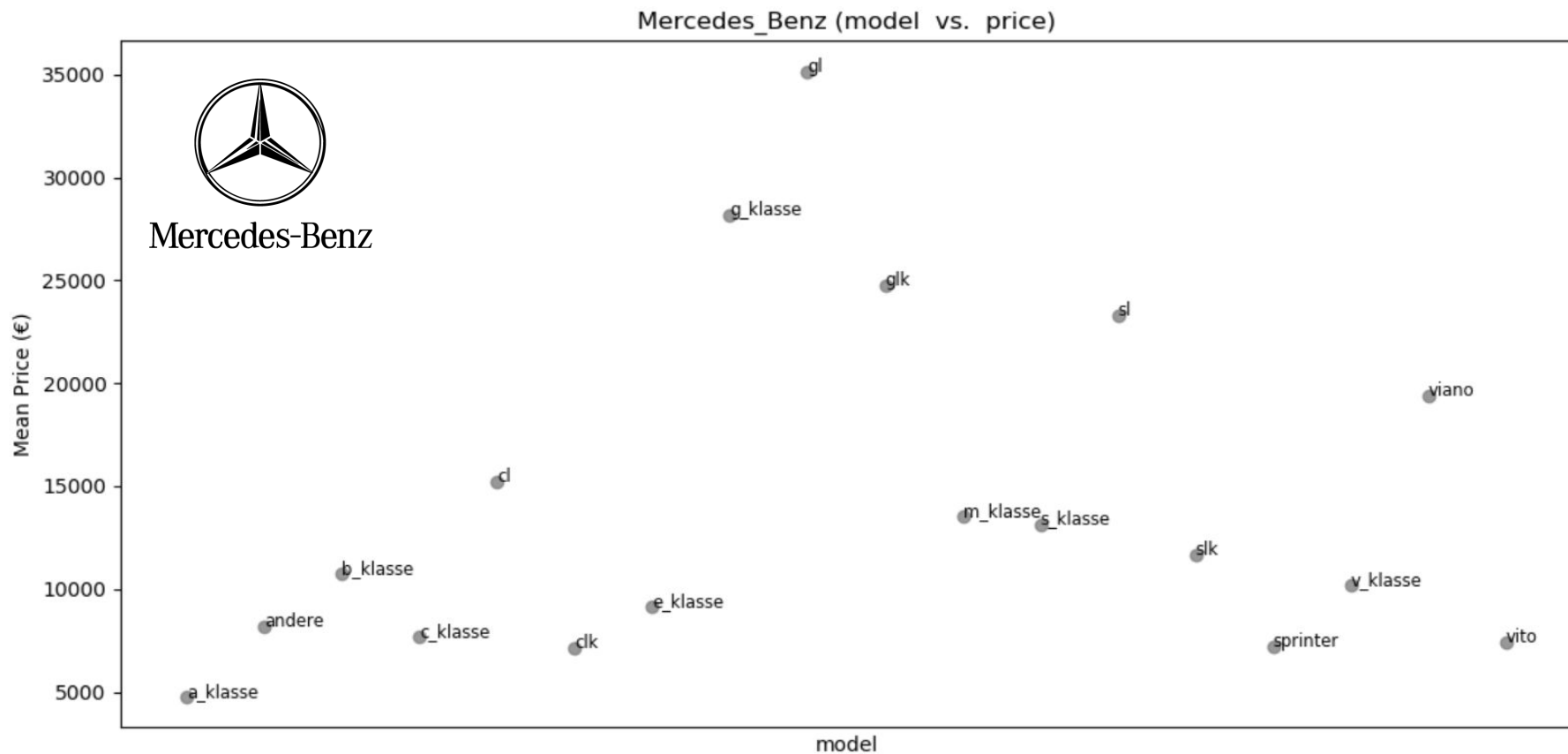
# Data Observation - *model* (in the same brand) vs. *price*



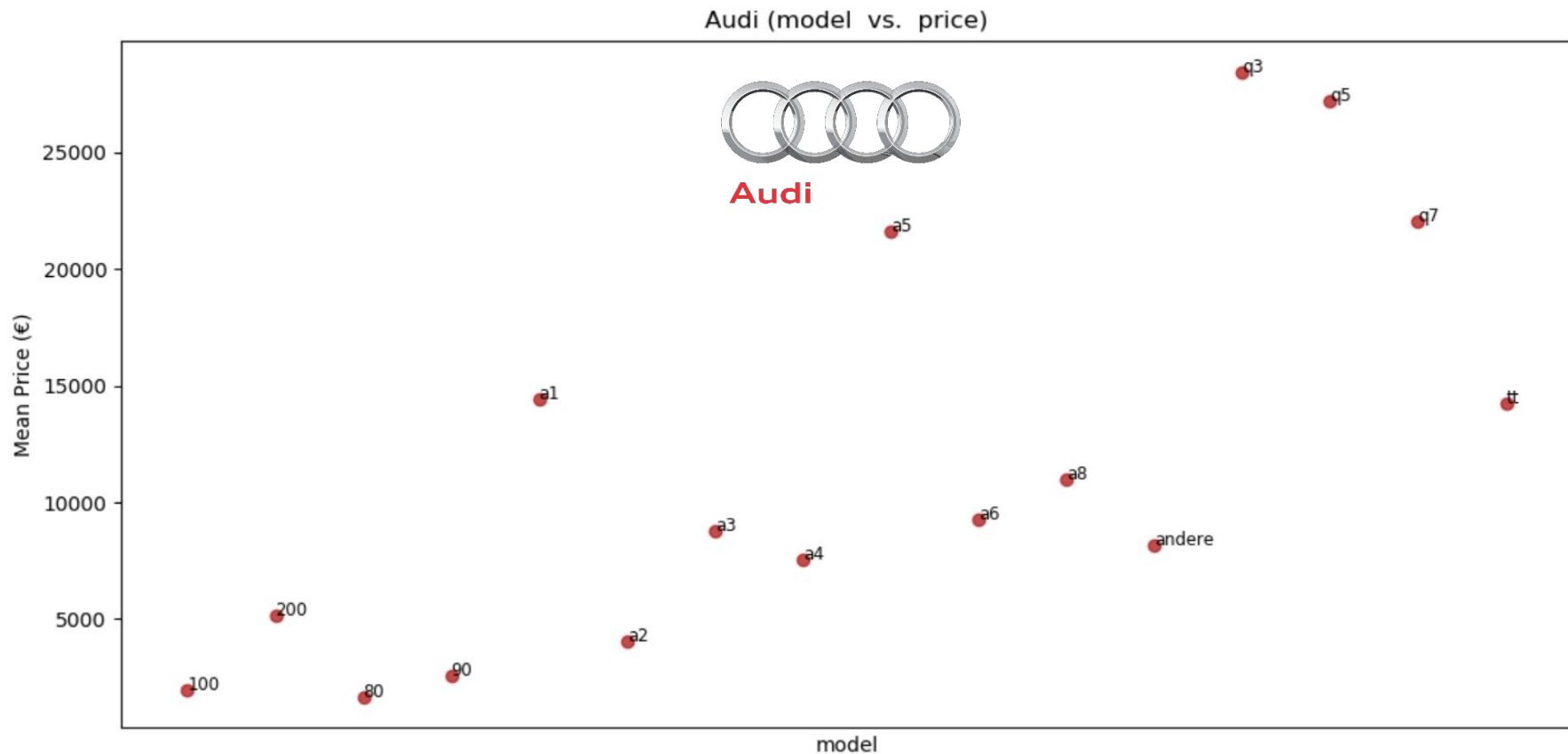
# Data Observation - *model* (in the same brand) vs. *price*



# Data Observation - *model* (in the same brand) vs. *price*



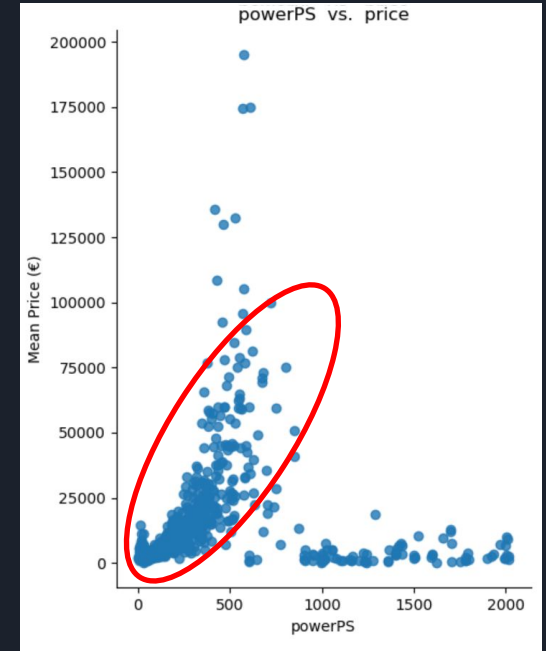
# Data Observation - *model* (in the same brand) vs. *price*



# Data Observation - *powerPS* vs. *price*

Using all data

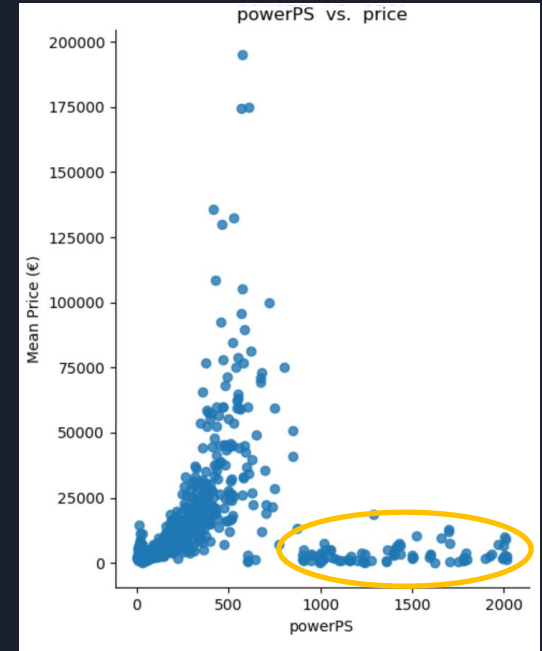
- We can obviously see the price increment following by PowerPS.



# Data Observation - *powerPS* vs. *price*

## Using all data

- We can obviously see the price increment following by PowerPS.
- Because there are different brands and models, some irregular plots would be produced.





# Data Observation - *notRepairedDamage* vs. *price*

We take

Porsche



: which has the highest mean price

PORSCHE

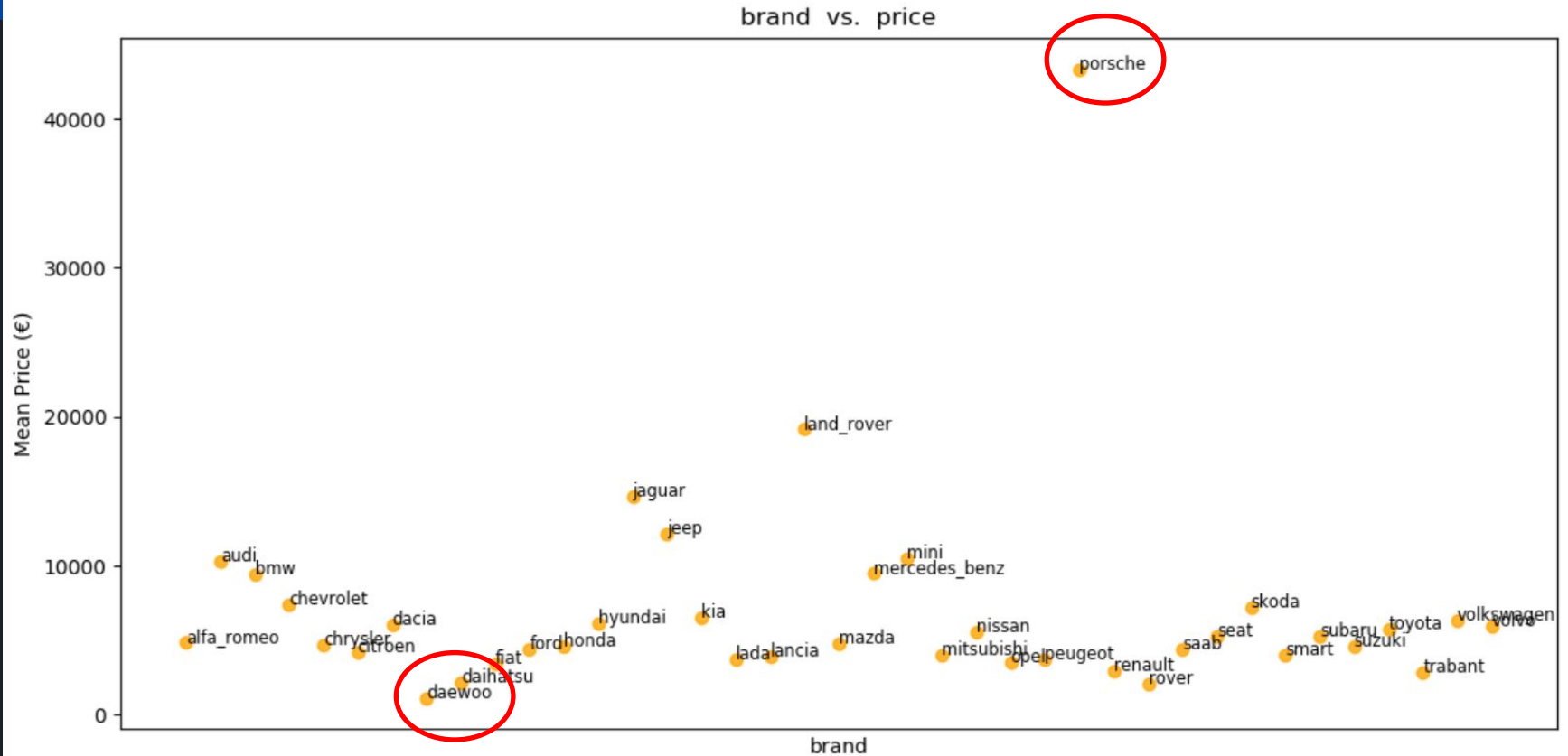
Daewoo



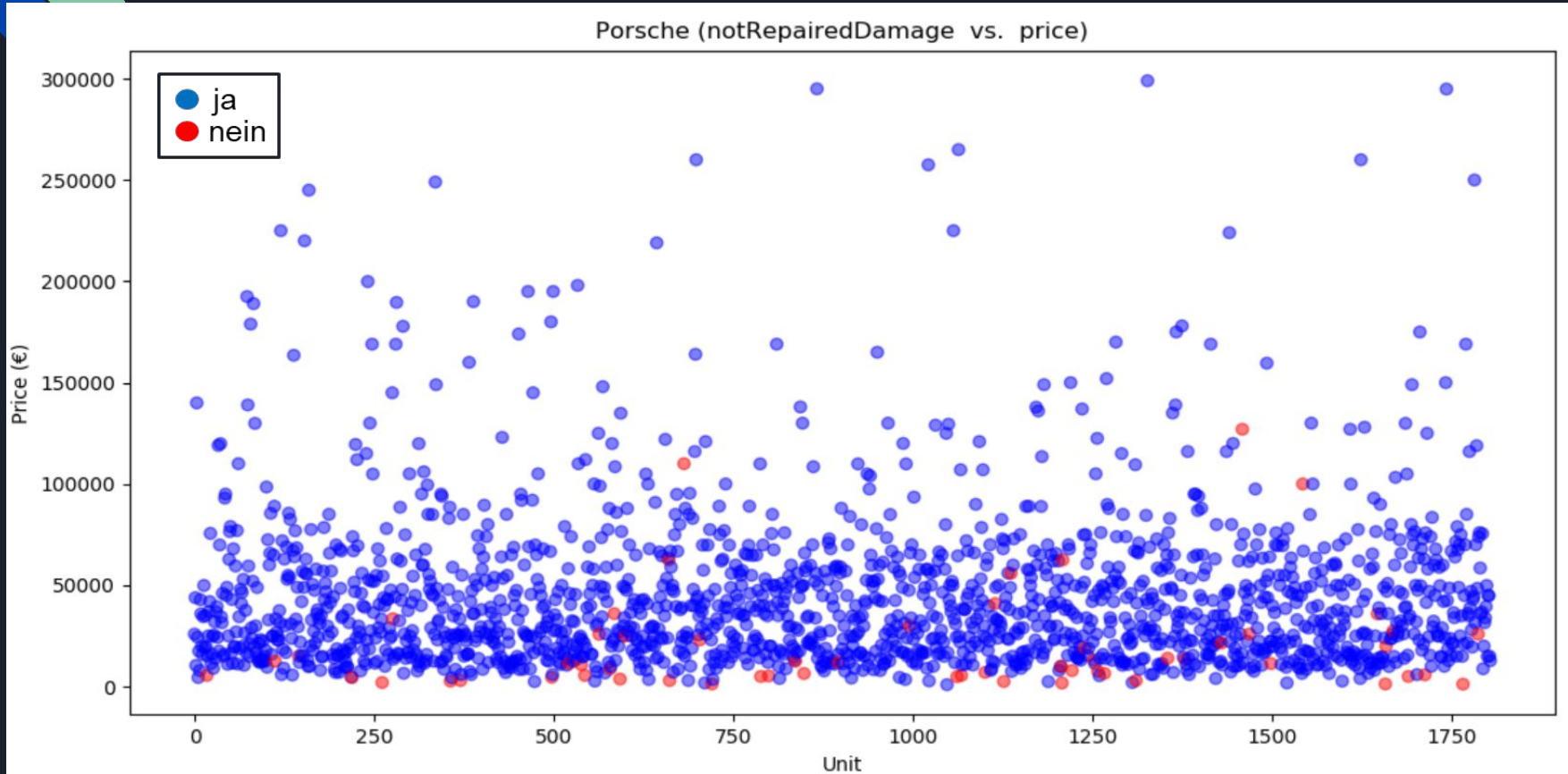
: which has the lowest mean price

to do comparison in *notRepairedDamage* attribute.

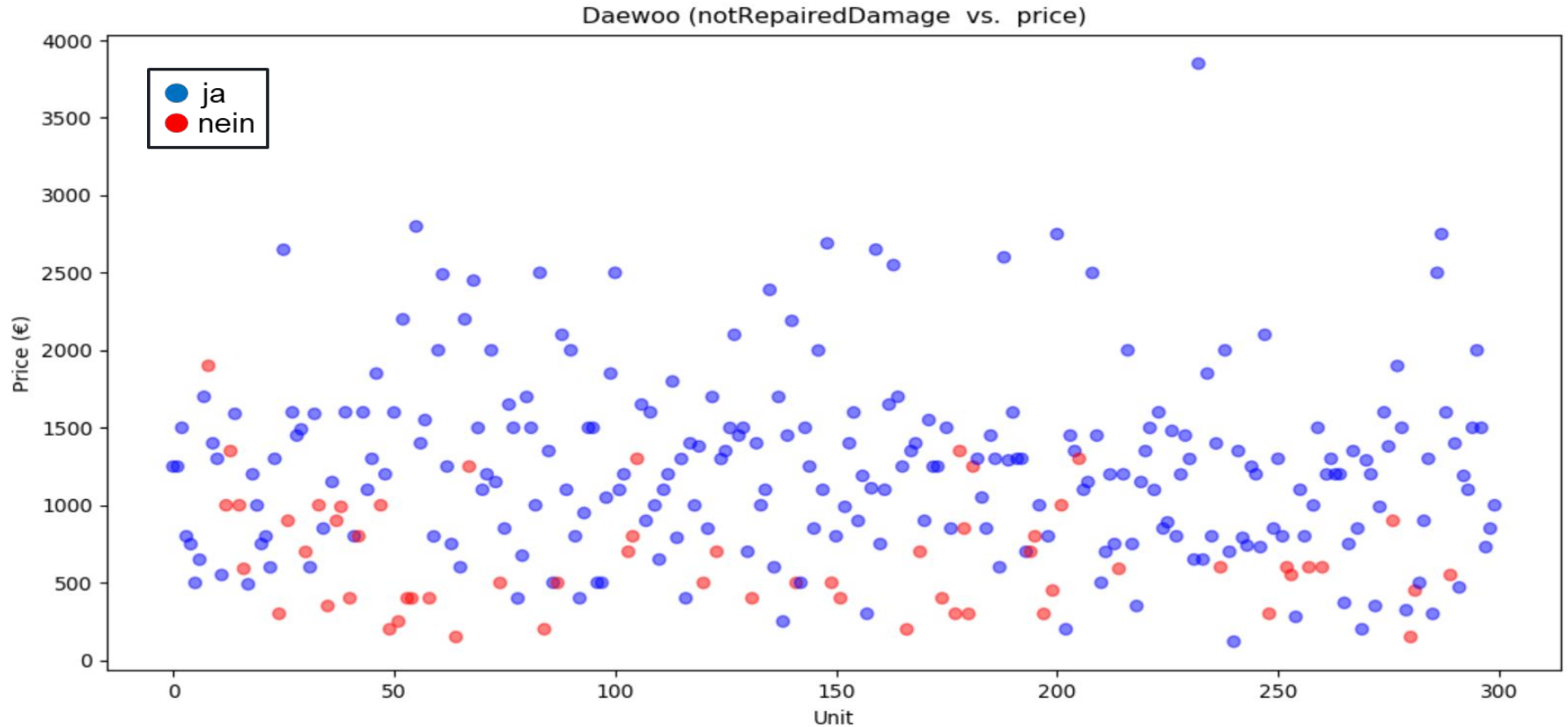
# Data Observation - *notRepairedDamage* vs. *price*



# Data Observation - *notRepairedDamage* vs. *price*



# Data Observation - *notRepairedDamage* vs. *price*





## Data Observation - *notRepairedDamage* vs. *price*

- We can see that blue points are more than red points.  
→ It means used cars are usually repaired damage by the seller.
- Looking at the scatter plot, we can know that  
... cars which are repaired have higher price than those are not.
- Consumers also can evaluate whether repairing themselves can cost less money than buying a already repaired car.
- Again, some repaired cars may seem like having the same price as not repaired cars because of different models.



# Methods

- Model Prediction
  1. SGD Classifier (Linear SVM with Stochastic Gradient Descent)
  2. Linear Regression
  3. Naive Bayes Classifier
  4. Random Forest



# Methods - SGD Classifier

Why using SGD classifier (Linear SVM with SGD)?

- Initially, I chose SVC in sklearn, however, it run endlessly.
- *SGDClassifier* in sklearn can be a linear SVM model by default. Moreover, it has a *regularizer* that shrinks model parameters towards the zero vector using either L2 or L1 norm as a penalty.



# Methods - Linear Regression

## Why using linear regression?

- Referred to the related work, we also want to know the predictive effect of linear regression.
- Also, we compare it to other regression models: *Ridge*, *Lasso*.





# Methods - Naive Bayes Classifier

Why choosing Naive Bayes in sklearn?

- Make comparison with other models
- Almost no parameters can be adjust. convenient!



# Methods - Random Forest

Why using random forest in sklearn?

- Make comparison with other models
- Being able to play with parameters

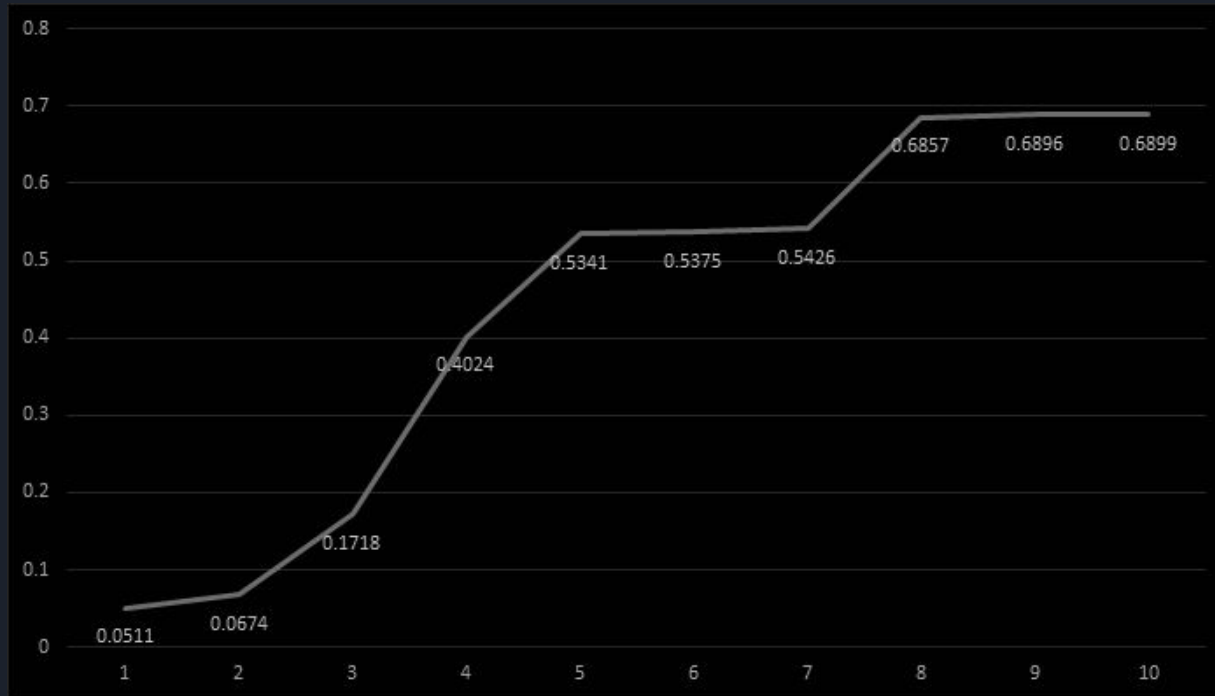


# Experimental results

- Model Prediction
  1. SGD Classifier (Linear SVM with Stochastic Gradient Descent)
  2. Linear Regression
  3. Naive Bayes Classifier
  4. Random Forest

# Experimental results - SGD Classifier

After modifying several parameters for many times, the accuracy gradually increased:





# Experimental results - SGD Classifier

Settings with the highest accuracy: 0.6898

- Parameters: `max_iter=200`, `n_jobs=4`, `learning_rate='optimal'`
- Discretization of training data:
  - 'powerPS': divided by 100
  - 'price': divided by 5000
- Training data being encoded by *oneHotEncoder*



# Experimental results - SGD Classifier

Findings from the parameter modification results:

- Settings that **INCREASE** the accuracy:
  1. Larger interval of discretization of 'price' attribute.
  2. Training data encoded by *oneHotEncoder*.
- Settings that **DECREASE** the accuracy:
  1. Training data without 'model' attribute (Initially, I assumed this column may decrease the accuracy, so I dropped it. However, the result is opposite).
  2. Usage of discretization of 'yearOfRegistration' attribute.
  3. Set `random_state` (to 42) while using *train\_test\_split*.



# Experimental results - SGD Classifier

## Accuracy comparisons between different settings:

- Using *train\_test\_split*: with vs. without setting *random\_state* (to 42)  
(Fixed: max\_iter=200, 'price' divided by 1000)  
➡ 0.0674 vs. 0.1718
- Discretization of 'price': divided by 1000 vs. 3000 vs. 5000  
(Fixed: max\_iter=200, without setting random\_state, Encoded by oneHotEncoder)  
➡ 0.2465 vs. 0.5341 vs. 0.6886 (all with oneHotEncoder)
- OneHotEncoder : not using vs. using  
(Fixed: max\_iter=200, 'price' divided by 3000, 'without setting random\_state, Encoded by oneHotEncoder)  
➡ 0.4023 vs. 0.5341



# Experimental results - Linear Regression

Metrics values of the prediction results in different settings

			alpha			
			0.3	0.5	0.7	1.0
normalize	True	MSE	20054674.666	20054674.666	17666759.722	17666759.722
		MAE	2040.1839	2040.1839	2050.8953	2050.8953
		Explained Variance Score	0.7555	0.7555	0.7807	0.7807
	False	MSE	17564999.8526	20013780.2219	17564999.8526	20013780.2219
		MAE	2050.96271034	2040.68972552	2050.96271034	2040.68972552
		Explained Variance Score	0.7820	0.7559	0.7820	0.7559





# Experimental results - Linear Regression

Findings from the parameter modification results:

- *normalize*: True vs. False (Fixed *alpha*)

Ridge and Lasso are affected more by *normalize* in our dataset.  
With higher error (MAE) while *normalize=True*.

- *alpha*: 0.3 vs. 0.5 vs. 0.7 (Fixed *normalize*)

Lasso is affected more by *alpha* in our dataset.  
With higher error (MAE) while *alpha* increases.



# Experimental results - Linear Regression

Comparison between linear regression and other regression models.

- *normalize*: True vs. False ( $\alpha=1.0$ )

alpha=1.0			Linear Regression	Ridge	Lasso
normalize	True	MSE	17666759.722	24831971.3828	30235969.0826
		MAE	2050.8953	2447.3858	2970.0680
		Explained Variance Score	0.7807	0.6918	0.6248
	False	MSE	20013780.2219	19773120.6169	21322752.2178
		MAE	2040.6897	2042.6073	2115.5061
		Explained Variance Score	0.7559	0.7589	0.7400

# Experimental results - Linear Regression

Comparison between linear regression and other regression models.

- *alpha*: 0.3 vs. 0.5 vs. 0.7 (*normalize=False*)

normalize=False			Linear Regression	Ridge	Lasso
alpha	0.3	MSE	17564999.8526	17481014.2717	17937955.124
		MAE	2050.9627	2050.7540	2063.6833
		Explained Variance Score	0.7820	0.7830	0.7774
	0.5	MSE	20013780.2219	19808194.4451	20599599.2321
		MAE	2040.6897	2041.0819	2071.2331
		Explained Variance Score	0.7559	0.7585	0.7488
	0.7	MSE	17564999.8526	17480313.018	18546463.7075
		MAE	2050.9627	2051.5244	2096.2860
		Explained Variance Score	0.7820	0.7830	0.7698


# Experimental results - Naive Bayes Classifier

From the result, we find that the accuracy is better if we encode the features (clustered first if they're real number) with one hot encoding.

			price=floor(price/x) , x=						uniform group size			
			1000	2000	4000	6000	8000	10000	#group=2	#group=3	#group=4	#group=5
model			time(sec)									
	data processing		accuracy(*100%)									
MultinomialNB			1.15	1.07	0.84	0.70	0.66	0.72	0.64	0.65	0.60	0.59
	one hot encoding		0.17	0.31	0.46	0.56	0.62	0.67	0.70	0.53	0.45	0.38
MultinomialNB			0.44	0.29	0.20	0.15	0.13	0.11	0.04	0.04	0.04	0.05
	string to number		0.11	0.21	0.45	0.55	0.62	0.66	0.69	0.52	0.44	0.36
BernoulliNB			0.97	0.75	0.62	0.58	0.55	0.54	0.45	0.46	0.46	0.47
	one hot encoding all		0.26	0.41	0.58	0.68	0.75	0.79	0.84	0.71	0.61	0.54

# Experimental results - Random Forest

discretizing		price=floor(price/x) , x=						uniform group size			
		1000	2000	4000	6000	8000	10000	#group=2	#group=3	#group=4	#group=5
model		time(sec)									
	data processing	accuracy(*100%)									
RandomForestClassifier		1.68	1.55	1.54	1.29	1.12	1.10	1.18	1.15	0.85	0.80
	one hot encoding	0.17	0.30	0.46	0.56	0.62	0.67	0.70	0.53	0.45	0.37
RandomForestClassifier		0.61	0.51	0.35	0.20	0.19	0.13	0.06	0.04	0.05	0.07
	string to number	0.15	0.29	0.45	0.55	0.62	0.66	0.69	0.52	0.43	0.36
model		time(sec)									
	data processing	mse									
RandomForestRegressor											
	one hot encoding	28.27		7,786,897.							
	string to number	3.09		8,367,024.							
RandomForestRegressor - parameters tune by EC (training data: "string to number") and a person											
	one hot encoding	5.07		8,949,910.							
	string to number	1.33		8,085,984.							



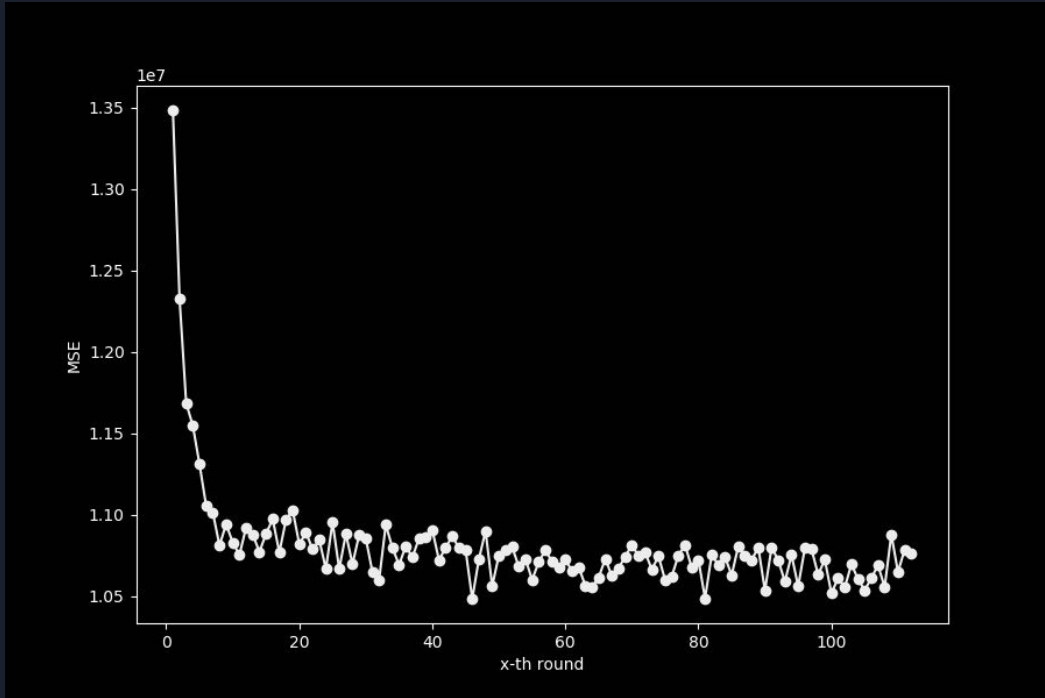
# Experimental results - Random Forest

- Random forest can take continuous features. Thus, there's no experience of all feature one hot encoded.
- Compared with NB, almost same accuracy at each condition, but slower.
- For regressor version, MSE are very large. It's probably that some values are too big, and mean is not enough to lower down the squared error as others.
- There are some interesting results below. As we don't know what parameters can be the best, let computer decide them for us.

\* random search via Genetic Algorithm, searching values of parameters of { max\_features, bootstrap, min\_samples\_split, min\_samples\_leaf, min\_weight\_fraction\_leaf, min\_impurity\_decrease, max\_depth }. "use 'string to number' to train"

# Experimental results - Random Forest

MSE decreases while being trained more rounds, but converges around  $1e7$ .





# Issues

- CPU overloaded (i7-7700K..)
  - Restart unexpectedly while running random forest and regressions simultaneously.
  - Solutions: use GPU computing, or... Google TPU



## Other efforts on this project





# Conclusions and future works

- Conclusion
  - One hot encoding can drastically increase the accuracy if the dataset has some discrete attributes.
  - Multinomial Naive Bayes & Random Forest have better accuracy
  - For dealing with attributes that has quite a lot data being keyed mistakenly (ex: 2147483647 & 12345678 in the 'price' attribute), it's hard to drop them if they're not simply out of reasonable range.
- Future works
  - Use rule-based or even neural network to solve the reasonableness problem of price, in order to decrease the interval of discretization of 'price' to reach the same accuracy.



# Job description

- 盧俊言 35% - Data observation & visaulization
- 林子皓 35% - Model Prediction - SGD Classifier & Linear Regression
- 許金賢 30% - Model Prediction - Naive Bayes & Random Forest



# Reference

- [1] Rayid Ghani, Price prediction and insurance for online auctions, Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, August 21-24, 2005, Chicago, Illinois, USA
- [2] Pudaruth,S. 2014. “Predicting the Price of Used Cars Using Machine Learning Techniques”, International Journal of information & Computation Technology,4(7)
- [3] C Chen, L Hao, C Xu., Comparative analysis of used car price evaluation models - AIP Conference Proceedings, 2017



# Q & A