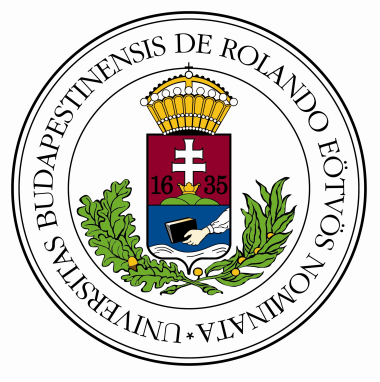
Datamining project

An analysis of used car prices

Domonkos Haffner

Date: 10/27/2020



Contents

1 Introduction

The aim of this project is to create a regressor model which accurately predicts the prices of used cars. I'm working with a used car dataset, which is described in section 2. The regressor I'm using in this model is the $ExtraTreeRegressor$.

2 The dataset

The dataset I'm using in this project consists of different features of used cars from the year 2016, originating from Germany.

There are more than 350.000 cars in the dataset, from which not all of the data be used. There are missing values, which must be filtered. After doing so, the remaining dataset consists of around 200.000 data. This however is not resulting in a perfect dataset, because we still need to clean the values. I checked the prices of the cars, where I found false values, for example: 12345678 or 99999999. Fortunately, there were not many non-realistic prices but just enough to ruin the prediction of the regressor. Not only did I remove these values, but also everything above \$80.000 and below \$1000. The *year of registration* column also contained some values, which needed to be removed, for example: 1800 or 9999.

The last column which contained lots of false data, was the horsepower column. After checking the way above average values, I found some horsepowers around $10.000 - 20.000$. These are clearly unrealistic values, since the highest hoursepower car in 2016 was around 500, according to the following website: [1]. Since these were luxury cars, which are most likely not sold on ebay, I made a cut-off at 300 horsepower. When chosing the lower limit, I considered the information on the following website: [2]. According to this website, the lowest horsepower car from 2016 which is still sold today, is around 66 horsepower. Thus I only considered the horsepowers above 50.

The original dataset had 19 columns in total. Not all of these were important to create

the models and predictions from the dataset, so I simply removed them. After this, the remaining dataset consisted of 8 columns, which were the following:

- price - price of the car
- vehicleType
- yearOfRegistration
- horsepower - power of the car in horsepower
- model - type of car
- kilometer - kilometers driven
- brand - brand of the car
- notRepairedDamage - if there is some damage which was not repaired

The head of the dataset - bofe removing the NaN values - can be seen in figure 1.

	price	vehicleType	yearOfRegistration	powerPS	model	kilometer	brand	notRepairedDamage
0	480	NaN	1993	0	golf	150000	volkswagen	NaN
1	18300	coupe	2011	190	NaN	125000	audi	ja
2	9800	suv	2004	163	grand	125000	jeep	NaN
3	1500	kleinwagen	2001	75	golf	150000	volkswagen	nein
4	3600	kleinwagen	2008	69	fabia	90000	skoda	nein

Figure 1: The head of the data

After selecting the important features of the data, one has to decide what to predict. Naturally, I chose the prices to be the predicted values. The dataset can be found and downloaded from the following website: [3].

3 The model

Before I started working on the regression model itself, I plotted the correlation matrix of the dataset. This joint probability describes, if greater values of one variable correspond to greater or smaller values of another variable. The correlation matrix can be seen in figure 2.

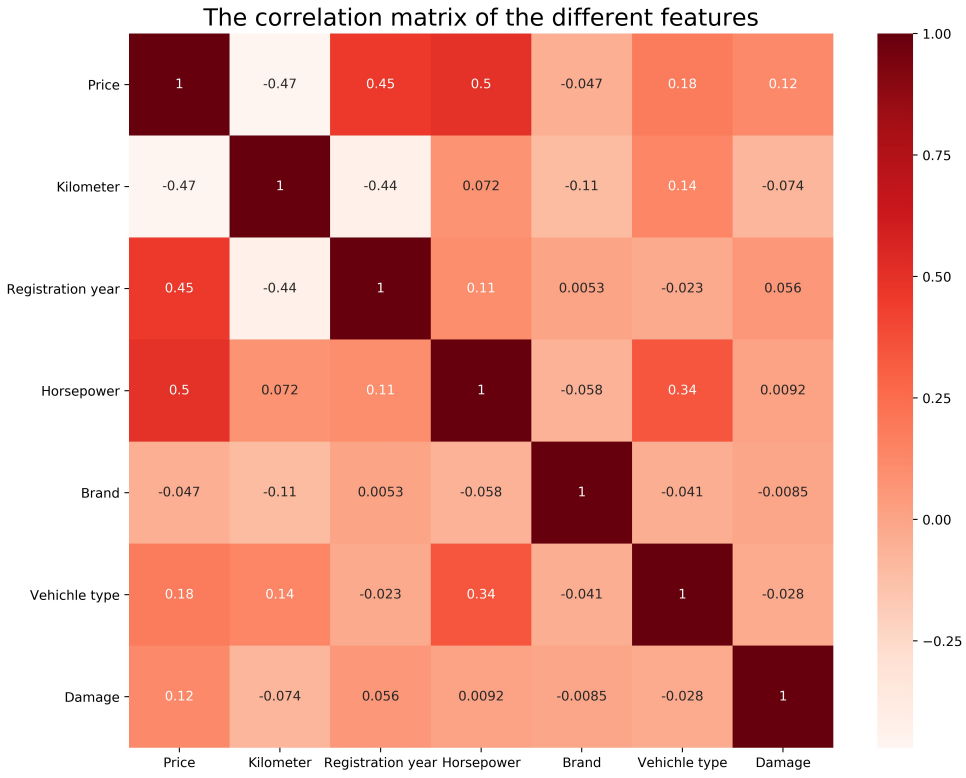


Figure 2

The highest and smallest correlations can be seen between the following variables:

- price - registration year: the newer the car is, the more it costs
- price - horsepower: the more power the car has, the more it costs
- vehicle type - horsepower: some vehicle types (coupé, bus) have more horsepower
- kilometer - price: the more kilomters the car is driven for, the less it costs

Interestingly, there was no significant correlation between the brand and the price. In my humble opinion, this might be due to the fact that people do not tend to sell luxury cars on ebay.

After filtering and cleaning the dataset, scaling was still required. Before this, I split the data into training and testing data. I used the training data's mean value (μ) and standard deviation (σ) to scale the data in the following way:

$$x_{scaled} = \frac{x_{raw} - \mu}{\sigma} \tag{1}$$

After scaling the data, I had to come up with a regression model for the dataset. After some thinking, I ended up using the $skelarn's\ ExtraTreesRegressor$ implementation [4], [5]. This regressor splits the data into various sub-samples, fits multiple randomised decision trees on them and averages out the solutions, so an accurate prediction is achieved. Since the default parameters of controlling the sizes of the trees lead to fully grown trees, one has to be careful that growing these trees do not consume such amount of memory that results in the collapse of the kernel. These can be controlled by the arguments of the function. There are two main differences between these extremely randomised trees and ordinary random forests. First, the top-down splitting in the tree is randomised and second, the whole learning sample is used to train each tree. One can select bootstrapping in $sklearn's$ implementation, which I ended up choosing instead of training the model on the whole sample.

After getting familiar with the $ExtraTreesRegressor$, I fitted the model on the training dataset. I measured the R^2 metric as the accuracy of the trained model. This measure represents a proportion of the variance, which is explained by the regression model. This means that (in our case) since the R^2 score with the $ExtraTreesRegressor$ model is 0.88, approximately 88% of the observed outputs can be explained by the model's inputs [6]. After fitting the model, I generated a predicted dataset according to the regression. The results of this predection - the predicted data in the function of the real data - can be seen in figure 3.

As it's seen, despite the fact that the relative error is comparatively high, the R^2 score

is very close to one. This means that the *ExtraTreesRegressor* is az excellent regressor to describe the data with. In the plot, the distance between each point and the slope is the relative distance between the predicted and actual values. The closer the points are, the higher the accuracy of the model is. As it's seen, most of the values were in a close range of the slope, which indicates the model's accuracy once again. There are relatively not too many outliers, compared to the fact that we considered approximately 200.000 cars.

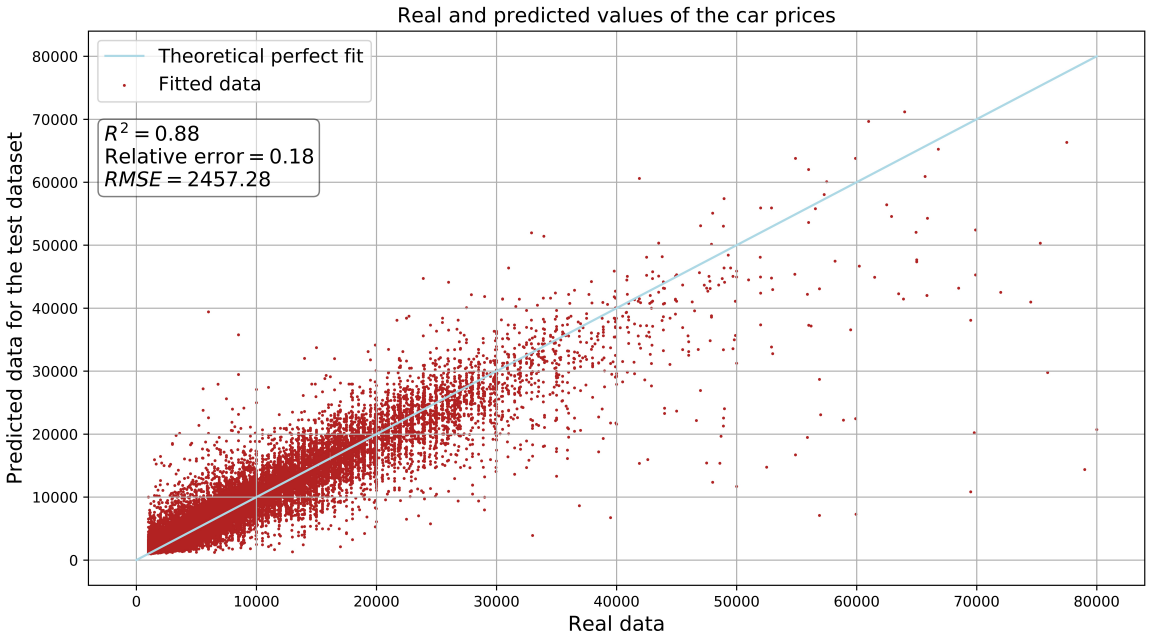


Figure 3

As it's seen, the ExtraTreesRegressor successfully and quite accurately predicted the test dataset. We can plot the residual plot now, which is the absolute difference between the predicted and actual y values. This can be seen in figure 4. As we would expect, the distance is the smallest in the beginning, since the car prices are the lowest then.

Whenever working with real-life - not toy - data, one has to set a limit what to accept as the outcome of the regression model. Since these are car prices, I considered the following: would I be happy about the predicted car price, if it was 30% more/less than the actual price? This is not ideal, however if I imagine a car being $1000 and I have a model which

predicts it's price between $700 and $1300, I'd be quite satisfied with that model! Thus I decided to add a 30% error line to figure 4, so it represents all values which are acceptable by my standards. One might ask the question, what the corresponding data percentages are to the different error thresholds. This can be seen in figure 5 as a barchart.

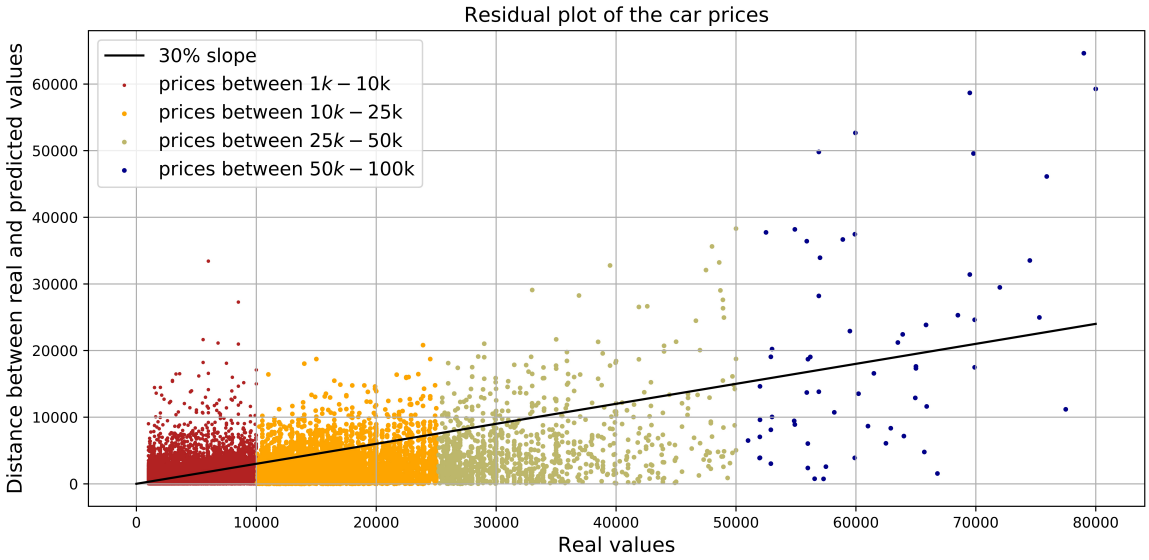


Figure 4

According to figure 5, only around 20% of the data is predicted with 5% accuracy. This might seem like a very low number, but let's not forget how dispersal the data is! At 15% error almost half of the training dataset is predicted with high accuracy. I didn't think this was sufficient enough for predicting the car prices, so I chose the 30% error threshold, where around 75% of the data is predicted with acceptable accuracy. I also calculated the fraction of values predicted with 50% accuracy, however this is way over the threshold we are looking for.

Figure 5

4 Conclusion

In conclusion, it can be stated that the *ExtraTreesRegressor* proved to be an excellent regressor to predict the car prices from the cleaned and scaled dataset. If we chose 30% as an acceptable error threshold, almost 80% of the data was predicted accordingly.

There might be some problems with the complexity of the trees generated by the *skelarn's ExtraTreesRegressor* implementation, that's why it's extremely important to read the documentation before starting to work with a new estimator.

References

[1] https://www.autoblog.com/photos/the-22-most-powerful-cars-we-drove-in-2016/slide-1384809

[2] https://www.cnet.com/roadshow/pictures/least-powerful-lowest-horsepower-cars-you-can-buy-pictures/23/

[3] https://data.world/data-society/used-cars-data

[4] $https://en.wikipedia.org/wiki/Random\_forest$

[5] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html

[6] https://www.investopedia.com/terms/r/r-squared.asp