Used car price prediction with gradient boosting

Bálint Hantos

Exploratory data analysis

Describing the data

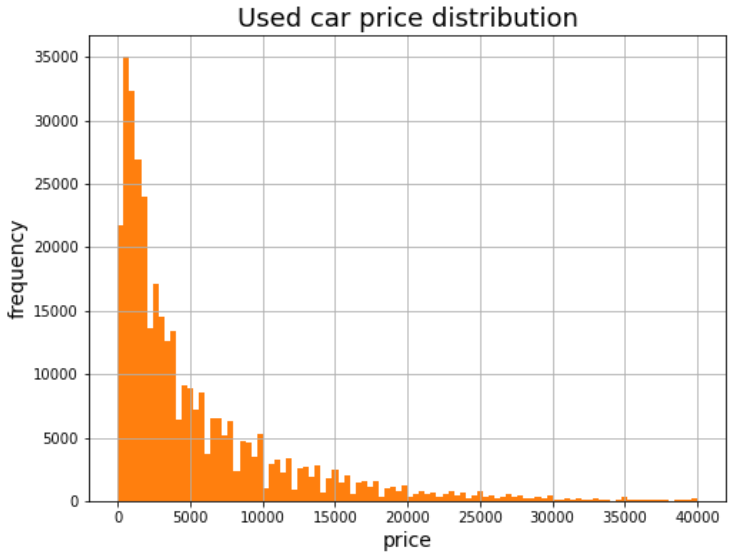
The data is originally in German, since I don't speak the language I used Google Translator to translate words and expressions I did not understand. The table consists of data on used car adverts. Each row contains information about the car, the seller and the ad itself. In the following I'm going to describe the data and give some insights about the columns.

First look at each numerical column:

- Regarding the price distribution, we can see, that the mean is much greater than any quartiles values. Based on that I claim, that the distribution is not that skewed by itself, so to have such a great mean, it must be because of some outliers.
- The year of registration column has some weird element(min and the max), but other than that, it looks okayish. The mean is okay, though the standard deviation doesn't make too much sense.
- Power has the same problems: some outliers.
- The odometer data is believable.
- The month of registration is a little off, because there is 0-12 months (=13) in this column.
- The number of pictures is a column with full of zeros. Better be removed, no predictive power at all.
- According to Wiki on German postal codes, those should be between 0-99999. Postal codes look okay. There could be some digging made, if some postal codes exist at all, but it's not worth the effort now.
- The dates on which the advert was created, and when it was last seen by the scraper.

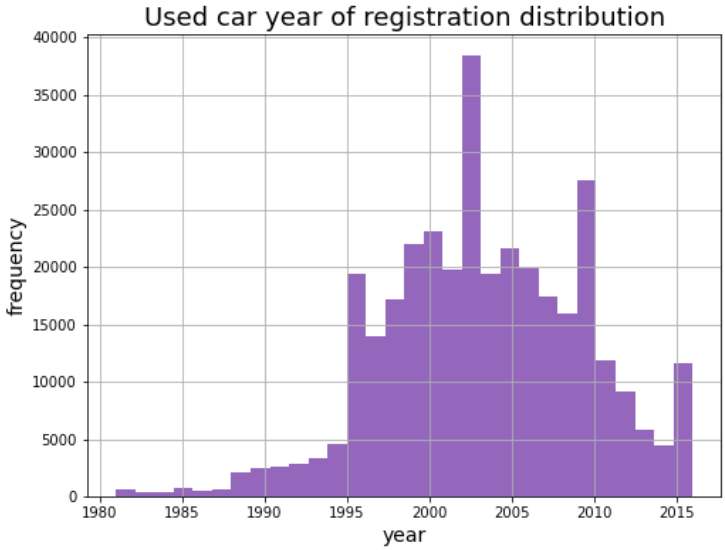
Numerical features

After looking at the price distribution, I decided to remove the too cheap or too expensive cars: plenty of €0 ads, which are misleading. I've set €40 000 (13M Ft @ 330HUF/EUR) as a maximum, because I think the average car buyer wouldn't spend more than that sum on a used car.
After filtering, the price distribution looks as follows:



1. Figure: filtered price distribution, price in EUR

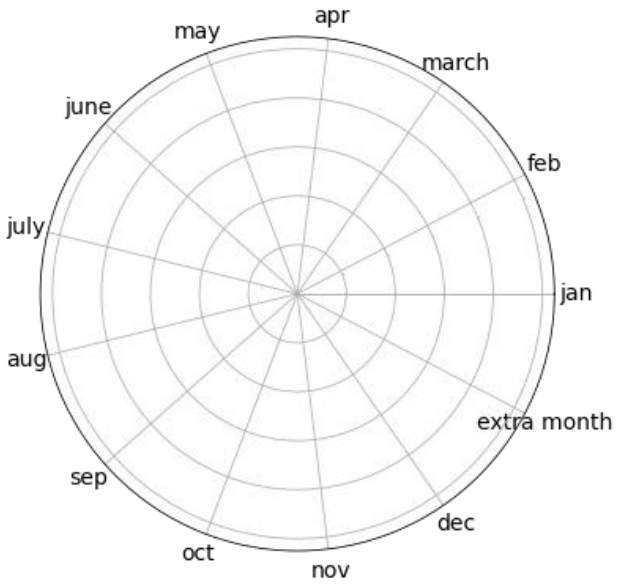
We don't want our model to predict prices of exceptionally old cars, so I limited the year of registration to 1980. Also, limit the max year at 2016, since that was the year when the data was scraped. Thus, the outliers and false data are filtered.



2. Figure: filtered year of registration distribution

Industrial vehicles and racecars were removed by setting the cap of power to be 1000. I also found, that ~10% of the power column were zeros. That might've been indication, that the data is missing. These missing values were imputed with the column median.

The month of registration column was a mysterious column, since there was 13 distinct values, even though, there are only 12 months. I didn't know what I could've done about it, since there was no pattern to recognize how there was 13 months. But this feature seemed valuable, so I didn't get rid of it. Additionally, we could introduce some periodicity. For example January and December are on the two ends of the scale, but actually very close.



3. Figure: more meaningful representation of month of registration

On Figure 3, I visualized a unit circle, to give some insight on the concept. I split this column to two features:

$$cos\ m/14\ , \qquad sin\ m/14\ ,$$

where m is the number of the month.

Two shorter notes: the number of pictures was 0 in every row, and the offer type and seller column had the same problem: almost every row were the same, so I got rid of all three columns, because they have no predictive power.

Additionally, based on the date of the ad creation and the ad's last seen date, I made a new feature: the minimum days of the ad being online.

Categorical features

Categorical features include :

- Most importantly the brand and the model of the car.
- The type of the fuel, the car uses.
- The type of the car (for eg: SUV, coupe)
- The type of the gearbox.
- The odometer status.
- Was it ever damaged, and if it was, were the damages taken care of?

There was plenty of missing values of these features. I chose the model as a key feature. Firstly, for every ad, where the model was missing I tried to scrape it from the advert's name. After this step, if there was no model name, I dropped them from the table.

In the case of other missing features (type of the car, fuel, gearbox), I grouped the rows by the model, and imputed the missing feature with the mode of the model's feature.

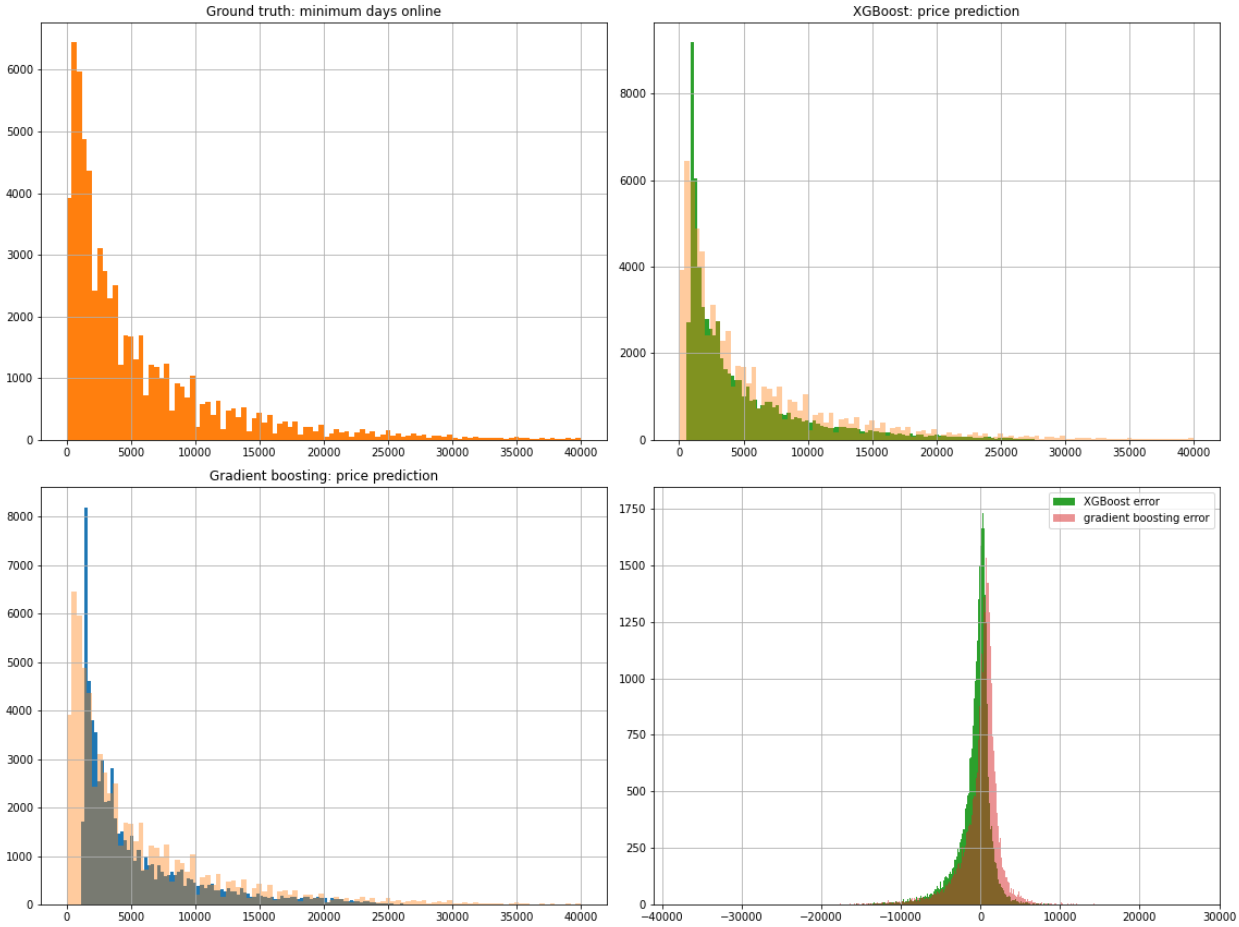
I presumed if the notRepairedDamage was missing, that'd mean, that the car wasn't even damaged at all.

Model specification

I used gradient boosting, because I wanted to try out a new model. Gradient boosting is an ensemble method like random forest: it is based on decision trees and uses bagging. But it is more than that, models are built sequentially by minimizing the errors from previous models while boosting influence of high performing models. The error is minimized by the gradient descent algorithm.

I tried to use gradient boosting from sklearn and also from the xgboost library.

Results



4. Figure: results of price prediction with gradient boosting

There was no significant difference between gradient boosting and xgboost in accuracy. Although xgboost's runtime was 5x faster. From this picture we can see that the distribution of the predicted prices is very similar to the ground truth.

Sources

[1] Wiki on German postal codes: https://en.wikipedia.org/wiki/List_of_postal_codes_in_Germany