Report

Capstone Project
Determine the Price of Used Cars

Udacity Machine Learning Engineer Nanodegree

Ute Blume

January 2020

1 Definition

1.1 Project Overview

The target of the project is to predict the price of used cars. Vehicle experts expect times that are difficult to predict for the used car market even an existencial crisis is not in sight. The reasons given for this are the use of diesel technology and the discussion about driving bans on older diesel models, which have a direct impact on the residual values of the vehicles. There are also other factors, such as the current climate debate and the increasing use of Car-Sharing models, that can have an impact on used car sales.

1.2 Problem Statement

This is a regression problem where the model expects the price as output variable. Several input variables that describe the characteristics of a car are examined to see whether they influence the price. These input variables are e.g. the brand, the kilometers, the age of the car, the gearbox or the fuel type. Another interesting aspect is to find out if there are regional differences in Germany.

1.3 Metrics

To compare the performance of the different algorithms, I will use the coefficient of determination R^2, Mean Squared Error (MSE) and Mean Absolute Error (MAE) as provided by scikit learn. These metrics are commonly used to compare regression results:

- **coefficient of determination** R^2 measures the size of the variance in the target vector, which is explained by the model. The closer R^2 is to 1, the better the model.
- **the mean square error (MSE)** is the sum of all squared distances (errors) between predicted and true values. The higher the value of MSE, the worse the model is.
- **the mean absolute error (MAE)** is the sum of all absolute distances (errors) between predicted and true values. The higher the value of MAE, the worse the model is. Due to the squaring, large errors are significantly more important for the MSE than for the MAE. Therefore, it is sometimes beneficial to look at both errors, even if they develop similarly.

2 Analysis

2.1 Data Exploration

2.1.1 The datasets

For my analysis I will use the „Used cars database“provided by Kaggle[1]. The dataset contains 371,528 observations and 20 features, so there is a wide range to explore how they influence the price. The data was scraped in 2016 with Scrapy from Ebay-Kleinanzeigen, the leading online market in Germany and contains the following columns:

1. dateCrawled: when this ad was first crawled, all field-values are taken from this date
2. name: „name “of the car
3. seller: private or dealer
4. offerType
5. price: the price on the ad to sell the car $\rightarrow$ **the target variable**
6. abtest
7. vehicleType
8. yearOfRegistration: at which year the car was first registered
9. gearbox
10. powerPS: power of the car in PS
11. model
12. kilometer: how many kilometers the car has driven
13. monthOfRegistration: at which month the car was first registered
14. fuelType
15. brand
16. notRepairedDamage: if the car has a damage which is not repaired yet
17. dateCreated: the date for which the ad at ebay was created
18. nrOfPictures: number of pictures in the ad (unfortunately this field contains everywhere a 0 and is thus useless (bug in crawler!))
19. postal_Code
20. lastSeenOnline: when the crawler saw this ad last online

[1] https://www.kaggle.com/orgesleka/used-cars-database

In my analysis I will use the price as target variable. While inspecting the dataset I found out, that I can drop the following features:

- dateCrawled: this is metadata
- dateCreated: this is metadata
- lastSeen: this is metadata
- name: Free text field that is filled freely without any specifications. This feature contains too many different entries and that it will not help us
- monthOfRegistration: In opposite to the year of registration the month of registration is not suitable as input variable
- seller/nrOfPictures and offerType: These features have been removed because they consist almost entirely of identical entries and do not provide needful information
- abtest: No A/B-Test is planned for this analysis
- model: This feature contains too many different entries and that it will not help us

The feature „postalCode “contains 8080 different categorical numbers. In addition to the used car dataset, I have used a look-up-table to transform the feature to a federal state. All columns with missing values have been removed. After cleaning the data, the dataset consisted of 248,528 observations and 10 columns:

1. price
2. vehicleType
3. yearOfRegistration
4. gearbox
5. powerPS
6. kilometer
7. fuelType
8. brand
9. notRepairedDamage
10. federal_state

2.1.2 Descriptive Statistics of the data set after cleaning

There was a lot of cleaning that needed to be done. The dataset after cleaning has 251,555 observations and 10 columns:

- price
- vehicleType
- yearOfRegistration
- gearbox

- powerPS
- kilometer
- fuelType
- brand
- notRepairedDamage
- federal_state

The price has a range between 101 and 100,000 Euro, the yearOfRegistration goes back to 1970, the powerPS ranges between 31 and 589 and the kilometer between 5000 and 150000.

	price	yearOfRegistration	powerPS	kilometer
count	251555.000000	251555.000000	251555.000000	251555.000000
mean	6782.782584	2003.547300	129.714333	123896.901274
std	8046.121542	6.137569	61.866259	39757.475292
min	101.000000	1970.000000	31.000000	5000.000000
25%	1690.000000	2000.000000	86.000000	100000.000000
50%	3999.000000	2004.000000	116.000000	150000.000000
75%	8900.000000	2008.000000	160.000000	150000.000000
max	100000.000000	2016.000000	589.000000	150000.000000

Abbildung 2.1: Description of the dataset after cleaning

2.1.3 Exploratory Visualization

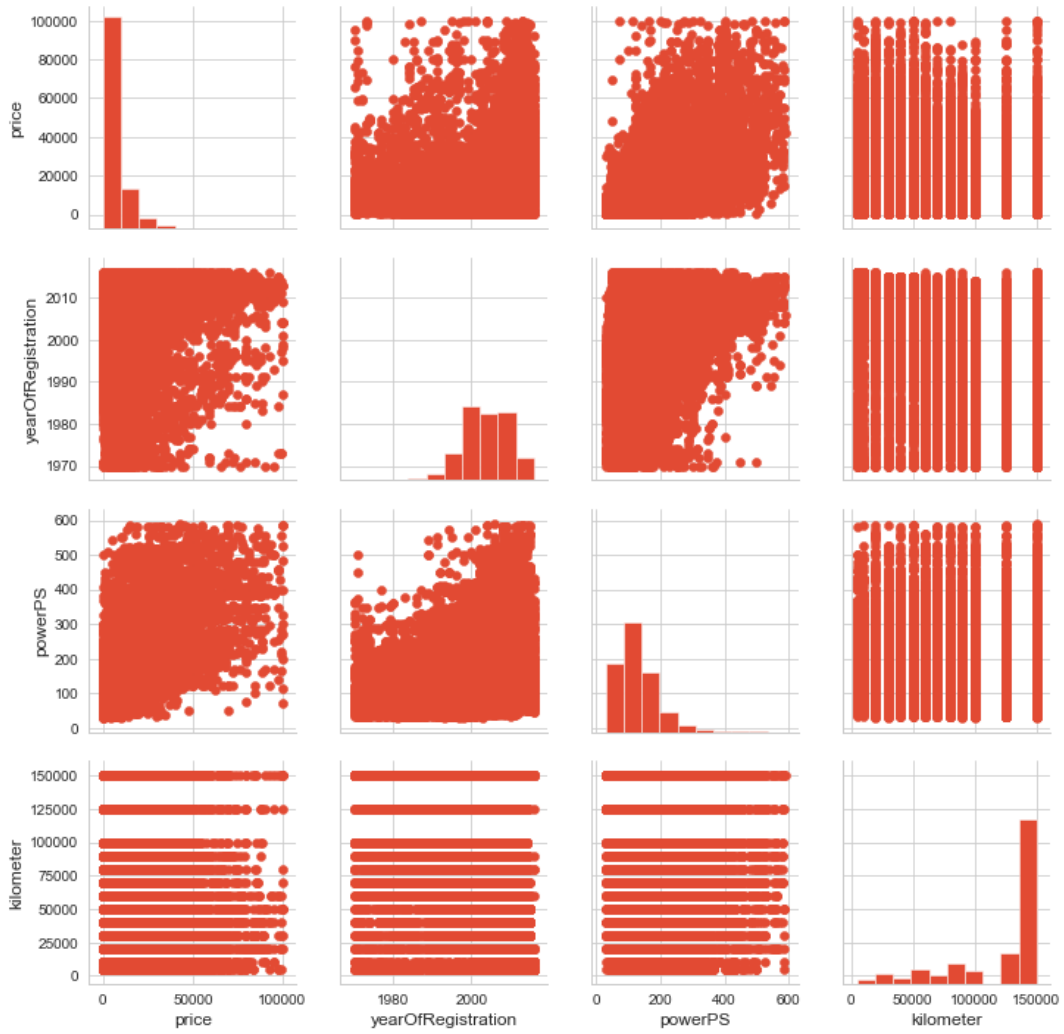


Abbildung 2.2: PairPlot of the data

First of all let's have a look at the histograms of the pair plot:

1. the price histogram is right skewed and has a long tail. Most of the prices are between 0 and 50000 EUR, so we can make the tail shorter
2. the year of registration is in most cases between 1980 und 2016
3. PS is in most cases smaller than 400 PS
4. we have a large number of cars having around 150000 kilometers. Since this is a used car market this seems to be fine for me.

	price
price	1.000000
yearOfRegistration	0.607277
powerPS	0.572826
kilometer	-0.484803

Abbildung 2.3: Correlation between the numerical features of the dataset and the price

We can see by the correlation table that price is influenced by the year of registration, PS of the car and finally by kilometer. In general we can say **a car is more expensive when it is younger, has more horsepower and fewer kilometers.**

Now we need to have a look at the categorical features:

- **gearbox:** Cars with an automatic gearbox seem to be more expensive and have more PS than cars with a manual gearbox. So gearbox seems to be an important feature.

Abbildung 2.4: PairGrid of Dataset (sample) highlighted by the gearbox feature

- **notRepairedDamage:** Cars having a not repaired damage are much cheaper than other ones and have on average more kilometers. It seems to be an important feature.

Abbildung 2.5: PairGrid of Dataset (sample) highlighted by the notRepairedDamage feature

- **fuelType:** fuelType is a very interesting feature. The market is dominated by gasoline (benzin) and diesel drives, alternative drives like hybrid or electro are still a niche product. Diesel drives are more expensive than gasoline and younger cars, but they have the most kilometers.

Abbildung 2.6: PairGrid of Dataset (sample) highlighted by the fuelType feature

- **federal_state:** The federal state seems to have an impact on the price-niveau. When we have a look at richer federal states like Bayern or Baden-W¸rttemberg we can see that the median price is very high just like the 75% percentil. That's maybe because these states have on average younger vehicles in their offers with more PS.

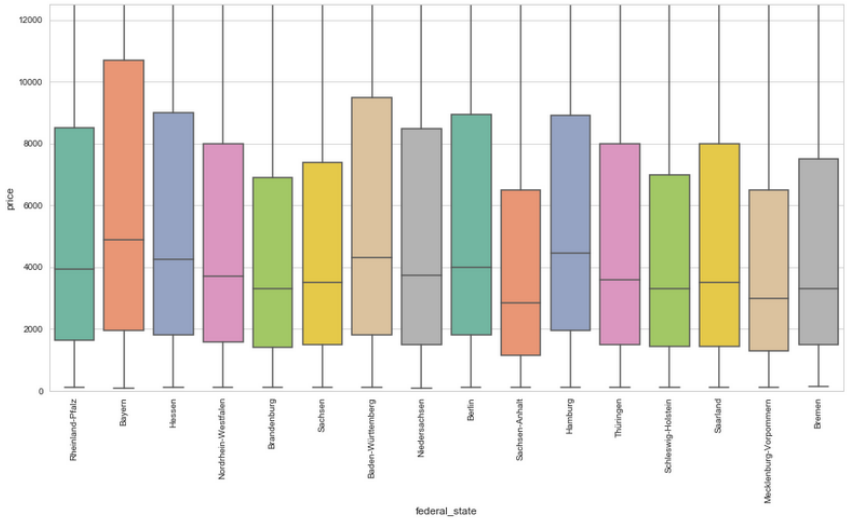


Abbildung 2.7: Boxplot of the price divided by the federal_state feature

- **brand:** There are car brands in very different price segments. This can also be seen very precisely on the box plot. Expensive brands such as Porsche have a much higher

average price than a cheaper brand such as Kia

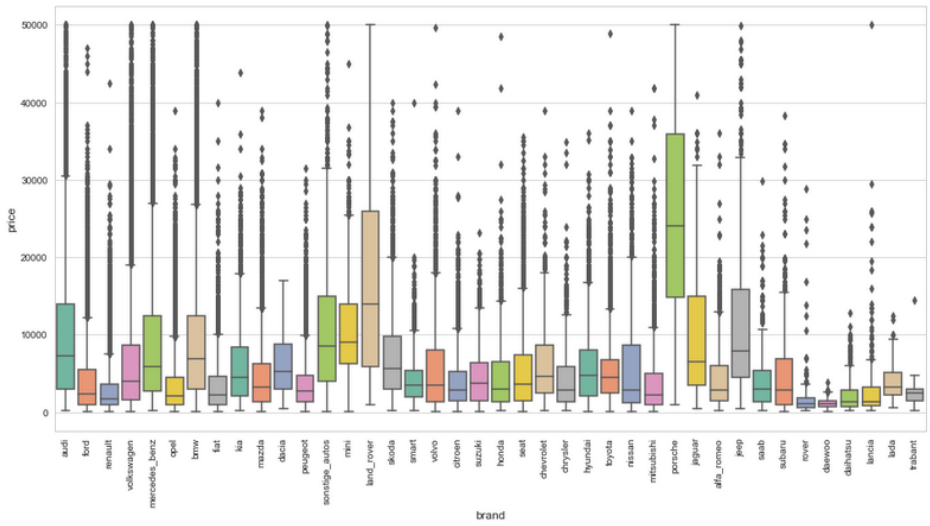


Abbildung 2.8: Boxplot of the price divided by the brand feature

- **vehicleType:** Well, I'm not quiete sure if the vehicleType has a big impact on the price, because some feature types are very similar to each other like e.g. the bus,station wagon (kombi) and limousine. But you can clearly see that the small car (Kleinwagen) is cheaper than an SUV or coupe.

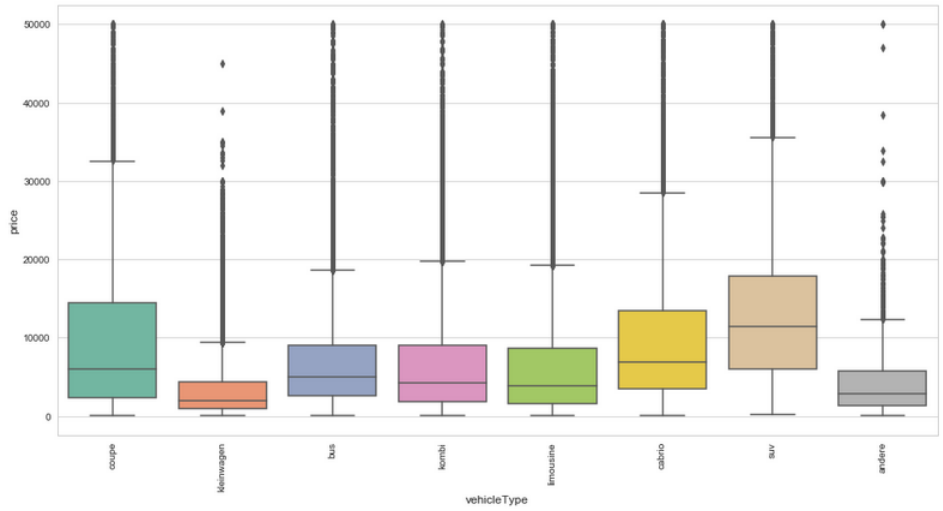


Abbildung 2.9: Boxplot of the price divided by the vehicleType feature

2.2 Algorithms and Techniques

This is a regression analysis and I want to try to build a model using the following algorithms:

- **Linear regression** is the most widely used algorithm when it comes to predicting quantitative values. In the present case, I want to use the model to find out whether

there is a linear relationship between the feature and the target vector. LinearRegression fits a linear model with coefficients to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation. In my analysis I want to use the scikit-learn library linear_model for linear regression[2].

- Decision trees often have the problem that they adapt too much to the data (overfitting). To compensate this disadvantage, the ensemble learning method **Random Forest** is used in this analysis. It creates a forest of decision trees. In this forest, each tree uses a subset of observations made using the bootstrapping technique. At each node, the decision rule looks at a subset of the characteristics. In my analysis I want to use the scikit-learn library ensemble for the random forest regressor[3].
- Finally I want to try the **GradientBoosting-Regressor**, because it can be used to improve the results of the RandomForest-Regressor. The GradientBoosting-Regressor builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function. In my analysis I want to use the scikit-learn library ensemble for the random forest regressor[4].

If I get a very good score using linear regression, I can end this analysis, because we have a linear problem and no more algorithms will be needed. But if not I will use the other algorithms to create a model for non-linear data.

2.3 Benchmark

The benchmark model is the **DummyRegressor** as it's provided scikit-learn[5]. The DummyRegressor is a regressor that makes predictions using simple rules. It can be useful as a simple baseline to compare with other (real) regressors, but it should not be use for real problems. In this case it always predicts the mean of the training set.

[2] https://scikit-learn.org/stable/modules/linear_model.html
[3] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html#sklearn.ensemble.RandomForestRegressor
[4] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html#sklearn.ensemble.GradientBoostingRegressor
[5] https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyRegressor.html

3 Methodology

3.1 Data Preprocessing

Before I can start to train the different algorithms I need to finalize the dataset. Therefore I first need to onehotencode the categorical features in my dataset. The function is provided by pandas and can convert series to dummy code consisting of one or zeros[1]. I need it for the following features: vehicleType, gearbox, fuelType, brand, notRepairedDamage and federal_state. With the onehotencoded data included the dataset now has 79 columns. This step is necessary because the algorithms cannot handle categorical variables

In the next step I need to split the dataset into the feature variables and the target variable. My target variable is the price and will be stored in a dataframe called y. The feature variables will be stored in a dataframe called x.

Then I will split the dataset into a training and a testing set. This step is necessary because the algorithm will learn on the training set. To check what and how good it has learned, I have to use unknown data for the algorithm. I keep the test data for this. To split the dataset I will use the train_test_split-function as it's provided in scikit-learns model_selection[2]. I will use a test size of 0.3 and a random state of 1809.

In the last step, the data is standardized because the data areas of the individual features are different. However, these differences should not affect the algorithms. I will use the standard scaler as it's provided in scikit-learn preprocessing[3].

3.2 Implementation

Using the final data set I can start to train the different algorithms. The process will be used as follows:

- import the regressor
- create the model (variable name: model)
- fit the model using the training set
- make predictions on the test set (variable name: y_pred)
- compute the scores using y_pred and the test data y_test

My aim is to get a R^2-Value that is close to one while MSE and MAE should be as low as possible.

[1] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html
[2] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
[3] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

3.3 Refinement

After using this process for the Dummy-Regressor, the LinearRegression and the RandomForest-Regressor I compared the results to find out, which model fits best. That is the RandomForest-Regressor. I used the grid search cv[4] to improve the results. The grid search tries to improve the result by specifying different parameters. It works according to the trial and error principle and you simply have to run several tests. Finally I tested the parameters n_estimator = [100, 150], max_depth = 20 and min_samples_split = [7, 9,11] and got max_depth = 20, min_samples_split = 7 and n_estimators = 100 as final result. Using these parameters I computed the RandomForest-Regressor again to run it on the full trainings set. Then I was able to compute the scores for the test set.

I also tried to improve the results using the GradientBoosting-Regressor. It's being used to improve the results of the RandomForest-Regressor. I used the same parameters as mentioned above added by the loss = lad parameter.

[4] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

4 Results

4.1 Model Evaluation und Validation

In the following table you will find the results of the different algorithms. The results were computed on the testing set and can be compared with each other:

	R^2	MSE	MAE
Dummy-Regressor	-2.7260	65.3 Mio.	5443
LinearRegression	0.6887	20.3 Mio.	2814
RandomForest-Regressor	0.8919	7.1 Mio.	2165
RandomForest-Regressor and grid search	0.8934	7.0 Mio.	1288
GradientBoosting-Regressor	0.8948	6.9 Mio.	1219

The results of the dummy regressor are very bad, because R^2 is smaller than zero und the MSE and MAE are very high. Using the mean of the training set for prediction leads to an error an average of 5443 Euro per car.

The results of the linear regression are getting better. The variance in the target vector, which is explained by the characteristics, is 68.9%, I was able to predict the price of a car with an error of 2814 Euro.

Again I could improve the results using the RandomForest-Regressor. The variance in the target vector, which is explained by the characteristics, is 89.2%, predicting the price of a car has on average an error of 2165 Euro. We can therefore state that this is not a linear problem at all.

Using grid search to improve the results of the RandomForest-Regressor or using the GradientBoosting-Regressor could improve the results just a little bit in the decimal range of R^2. GradientBoosting-Regressor was better than RandomForest-Regressor using grid search. Finally I got an R^2 of 89.5% and could predict the price with an error on average of 1219 Euro.

4.2 Justification

In the following table you will see the results of final model using the GradientBoosting-Regressor in comparison to those from the dummy-regressor:

	Final Model	Benchmark Model	difference
R^2	0.8948	-2.7260	3.6208
MSE	6,870,469	65,334,894	58,464,425
MAE	1219	5443	4224

The results could be improved significantly so I think I have found satisfactory solution for

the model.

5 Conclusion

5.1 Free-Form Visualization

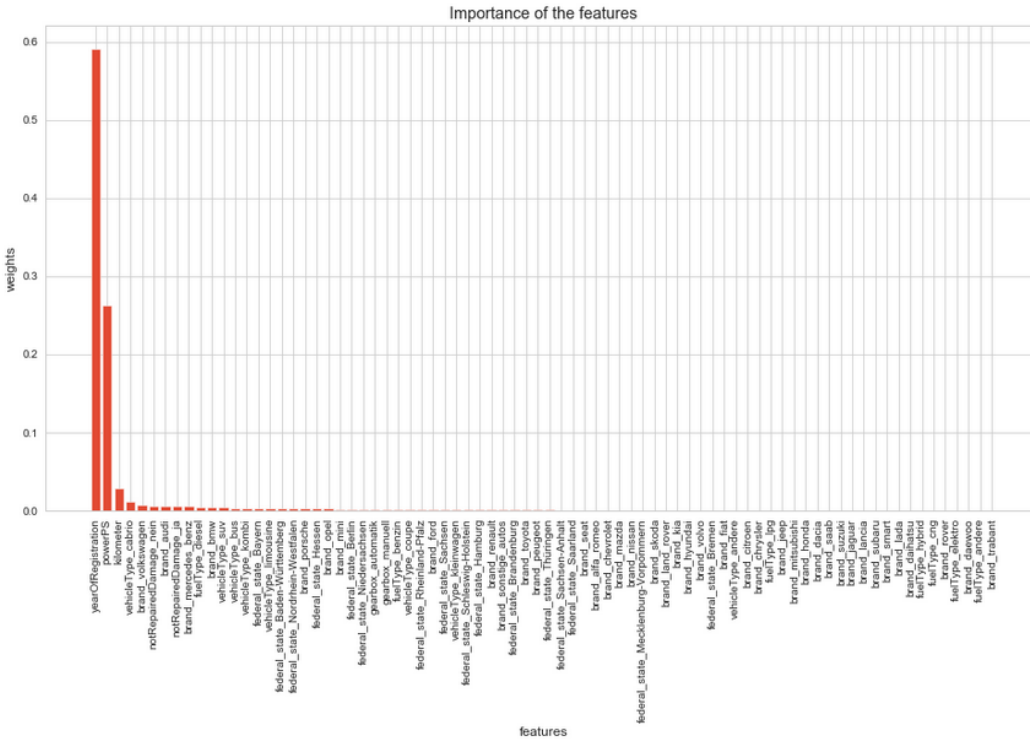
The final model is able to predict 89.5% of the variance of the price making an error on average of 1219 Euro per car. But I'm also interested to find out, which are the most important features to predict the price. These are my Top 5:

1. yearOfRegistration
2. powerPS
3. kilometer
4. vehicleType_cabrio
5. brand_volkswagen

And my Flop 5:

1. brand_smart
2. brand_lada
3. brand_daihatsu
4. fuelType_hybrid
5. fuelType_cng

Here is a visualization as bar chart:



The age of the (yearOfRegistration) is the most important feature followed by the PS and the kilometer. Then we have the vehicle Type cabrio and the brand volkswagen. I'm not very surprised of the first three features because they are very important to characterize a car. But I am surprised that the vehicle type cabrio is at number four. Maybe that's why the data was taken in summer of 2016.

The list of the flop 5 does not surprise me much. The brands smart, lada and daihatsu are not high quality cars when they are older and have a lot of kilometers. Alternative drives like hybrid and cng are niche products with a small number in the data set.

But I also have to notice that the regional differences as used as federal states do not have a high impact on the price. The richer states like Bayern oder Baden-Wuerttemberg are in 15th and 17th place and do not influence the price much.

5.2 Reflection

Working on the capstone project can be summarized by the following steps:

1. I wanted to work on a regression problem, because I have to deal with regression problems in my daily work. So I first wanted to work with a data set of train delays and created a first proposal. But my reviewer mentioned that it might does not have enough features and so I designed a new project. I found the dataset to predict used car prices on kaggle and wrote a new proposal.
2. I began with my analysis by importing the data
3. then I wrangled the data, droped features and narrowed the range of values ??if

necessary

4. I added the federal state based on the postal code
5. starting the exploratory data analysis
6. creating the model by preparing the data
7. training the different algorithm
8. choosing and improving the best model
9. feature engineering to find out which are the most important features

It was very interesting for me to work on a project from scratch. But I'm doing this for the first time so it was therefore very difficult for me to estimate how long it would take to proceed the individual steps, especially when improving the algorithms and writing this report. So everything took longer than I thought and I was really pressed for time because I want to work as well as possible.

It was a lot of fun for me to create the visuals for the EDA and for the most important features. Because they can give a good intuition on the data set.

5.3 Improvement

Here are my suggestions to improve the solution:

- collect more current data to find out, if there are any changes in the results
- I won't use the MSE error again, because R^2 and MAE are enough for my analysis
- using the LinearRegressor-Algorithm as benchmark algorithm and trying more other algorithms