Predicting Used Cars Prices

Cristian Baldi, Fabrizio Olivadese, Simone Vitali
University Of Milano Bicocca - Department of Computer Sciences, Systems and Communications

Abstract
The aim of this paper is to propose a machine learning model to predict the sale price of used cars. To train and validate the model a dataset obtained from scraping Ebay is used, consisting of user-submitted listings of various car models in various conditions. A crucial part of the problem was dealing with large amount of missing values, false and placeholder listings and overall outlier detection. After cleaning the dataset various machine learning models are compared to figure out the best one for the task. An ensemble-like model is also developed which proved a small improvement in overall performance. The performance of both solutions are then evaluated, resulting in good results, especially considering the training data in use, but with room for improvements.

I. Introduction

Used car sales accounts of about 33% of all the car sales in Europe and the used-car market share is expected to rise in the next years by 7% in terms of value for each year [Technavio 2017]. Buying and selling new cars is easier than dealing with used ones: manufacturers provide pricing sheets to the final customer that tell how much a certain car with a certain configuration will cost.
Dealing with the used market instead, it is much more complex: an inexperienced driver, looking to purchase its first vehicle, might find very difficult to tell if the price he has to pay for an used car is too high or right for the status of the car. At the same time, someone looking to sell an used car might be interested in knowing how much it can be sold for, without spending days in researching other listings for the same model and conditions.
In the used-car market, kilometers, age, general condition of the car, but also the maker and the model of the car, are all things to keep in mind when trying to estimate how much a car is worth. It is clear how an **automated tool to predict the value of a car** starting from some of its details, could be very useful for the car market as a whole.
The aim of this work is to build such a tool using machine-learning techniques, mainly implementing **regression models for the price** of a certain car.
First of all, different models that can be applied directly to the car characteristics to predict the price are compared, then it is shown an ensemble-like model that predicts the price by first trying to categorize the car in a general price-range direction (cheap, medium, expensive) and later using a specific machine learning model for the predicted price-range.
To train and validate the various models a **dataset consisting of around 370,000 uses-submitted listings** to Ebay Germany is used. The dataset contains a lot of dirty, missing and malformed data, consisting of typos, fake listings and wrong user-submitted data in some attributes of the car details.
The proposed problem is not an easy task to complete, firstly because second-hand prices still depends a lot on the seller feelings and estimations on what its car is worth (one might even put a price without comparing it to the other prices on the market) and secondly because the dataset in use for this problem, even after a preprocessing phase, still contains malformed data that could alter the predictions.

II. Related Work

Research work in this field is present but limited and only spanning in the last few years. [Noor and Jan 2017] proposes a valid regression model for predicting prices of used cars based on listings on a Pakistani website: the dataset is very different from ours, having a much smaller size and an higher number of characteristics available for each listing, as well as less missing values.
[Pudaruth 2014] compares different machine learning models for the prediction of used cars in Mauritius, the research is limited to 4 mayor makers and less than 400 listings and thus does not really represents the real market distribution; it uses both regression and classification (by binning the price variable) to estimate performances. Binning the price variable is an interesting approach but at the same time it transforms the problem from regression to classification and thus it doesn't allow anymore to use the classical performance measures of regression that are more interesting in this case, such as R^2, and MAE, thus it is decided to not follow this choice.
[Gegic et al. 2019] proposes an ensemble machine learning model for categorical classification of the binned price of used cars, dealing with a small dataset with more characteristics available for each car, than the one in the dataset presented here.
There is also work done on the same dataset that is used for this paper. On [Kaggle 2019] many users attempted to predict the prices of the used cars, but in different (and definitely more ideal situations than ours) conditions. Some filtered certain makers, while others only considered listings without missing values (or applied missing imputations in very naive ways) and some others only consider certain price ranges. In research, [Pal et al. 2017] use a *Random Forest Model* for predicting the price on a non-missing value subset of our same dataset, achieving a R^2 score of 0.84. This is the most similar result that our work can compare

to, even if it uses a different subset of data and does not apply missing imputation or any domain knowledge.
The work presented in this paper would generalize on all of the previous cited papers, by applying novel concepts both in the data preprocessing step and in the machine learning step of the problem, by dealing with a much bigger dataset (in terms of raw numbers and car models) and also by working with more real-world like data.

III. Dataset

The dataset used for the task consists of **370,000 user-submitted used car listings** to Ebay.de, collected from 10 March 2015 to 16 April 2016. The main language of the dataset is German, this is especially important during the preprocessing phase.

The dataset is composed of **20 attributes**, the following:

- **name**: title of the listing on the website;
- **brand**: manufacturer of the car, such as *"Fiat"*;
- **model**: model of the car, such as *"Punto"*;
- **type**: type of the car, such as *"coupe"*, *"sedan"*, ...;
- **gearbox**: gearbox, such as automatic or manual;
- **powerPS**: horsepower of the car;
- **kilometer**: kilometers of the car;
- **yearOfRegistration**: year the car was registered;
- **monthOfRegistration**: month the car was registered;
- **fuelType**: fuel of the car, such as *"diesel"*, *petrol*, ...;
- **price**: price of the car as proposed by the seller;
- **seller**: type of the seller, either private or agency;
- **offerType**: type of the listing, sale or request of purchase;
- **nrOfPictures**: number of pictures in the listings;
- **postalCode**: postal code where the car resides;
- **dateCreated**: date of when the listing was created;
- **dateCrawled**: date of when the listing was first scraped;
- **lastSeen**: date of when the listing was last seen online;
- **abTest**: unspecified data related to the scraper.

IV. Dataset Exploration

In this section some **insights on the dataset** are presented. First we will take a look at **general statistics** for the numeric attributes and for the nominal ones.

Attribute	Min	Max	Mean	Std. Dev.
price	0	2,147,483,647	17,286	3,586,525
yOfReg	1000	9999	2004	92.83
mOfReg	0	12	5.75	3.71
powerPS	0	20,000	115.54	192.07
kilometer	5,000	150,000	125,618	40,111

Table 1: Numerical Attributes Statistics

For the **numerical attribute statistics** it is clear that some of the attributes **contains malformed data**, for example just by looking at the extreme values (and to the standard deviation) of the attribute *"price"*it is possible to guess that there is definitely something wrong in the data, in this case outliers. It is even more clear with the attribute *"powerPS"*, since a car can't have 0 or 20000 horsepower, or with *"year"*, there are no cars from the year 1000 and definitely not from the future.

Attribute	Unique Values	Examples
name	233531	*New Mercedes A140*
vechicleType	9	*Coupe, Suv, Cabrio*
gearbox	3	*Manual, Automatik, ?*
model	252	*Focus, Micra, Golf*
fuelType	8	*Benzin, Diesel, Elektro*
brand	40	*Audi, Ford, Mitsubishi*
notRepDmg	3	*Ya, Nein, ?*

Table 2: Nominal Attributes Statistics

Taking a look at the nominal attributes table it is possible to see that the dataset covers 40 brands of cars and a total of 252 different car models. This is interesting to give a scope on the problem that will be solved.
There are also **attributes with missing values** in the dataset, distributed as presented here:

Attribute	Missing Values
price	0
name	0
type	37869
yearOfRegistration	0
monthOfRegistration	0
powerPS	0
fuelType	33386
brand	0
model	20498
kilometer	0
notRepairedDamage	72060

Table 3: Attributes Missing Values

All the missing values appear in the nominal features. The attribute that has the most missing values is *"notRepaired-Damage"*, probably because owners of repaired cars don't bother filling the repaired status. Even if there are no missing values for the *"powerPs"* or the *price field*, some are set to 0 (inferred from Table 1) and thus will need to be treated as missing values.

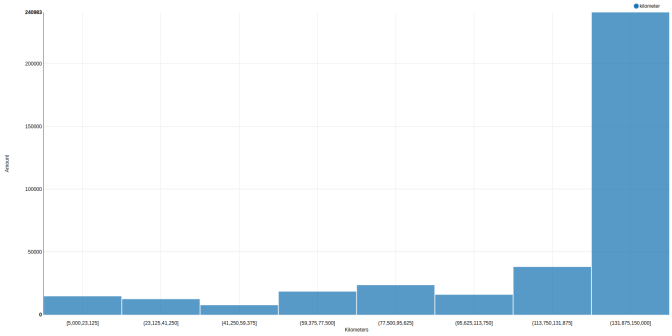

Figure 1: Kilometers Distribution

Analyzing the distribution of *"kilometers"* it is possible to notice that almost all cars have between 130.000 km and

150.000 km, as can seen in Figure 1; this tells that usually the proposed models will be dealing with very used cars. At the same time, it can be seen a potential issue with the dataset, no cars have more than exactly 150.000 km specified. This is probably due to a limitation of the software used to create the listings. Unfortunately nothing can be done about this.

Moreover, **some attributes are very dirty** and should be managed in the preprocessing phase. For example, the *year of registration* of a car presents unrealistic maximum and minimum values, as shown in Figure 2. The figure, even if not pretty, is really interesting to see the various outlier values for the *year of registration*, these rows will need to be fixed before moving on to the machine learning phase.

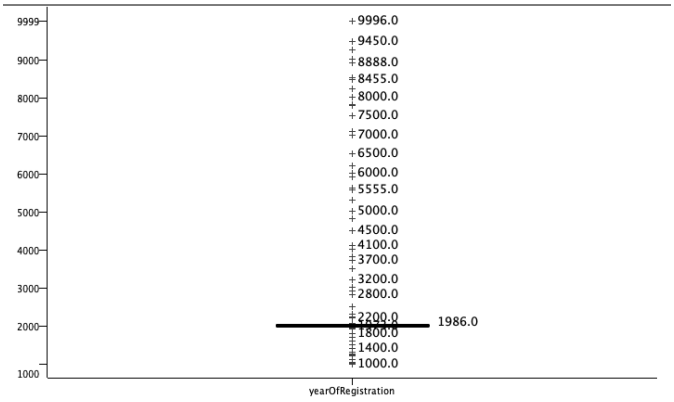

Figure 2: Year of Registration Box Plot

Cars in listings are distributed in 7 main different types, but, about 40000 listings, present missing or specified as "other" (*"andere"* in German) values in the *"vehicleType"* attribute, as shown in Figure 3. It will be necessary to handle these values in the preprocessing phase.

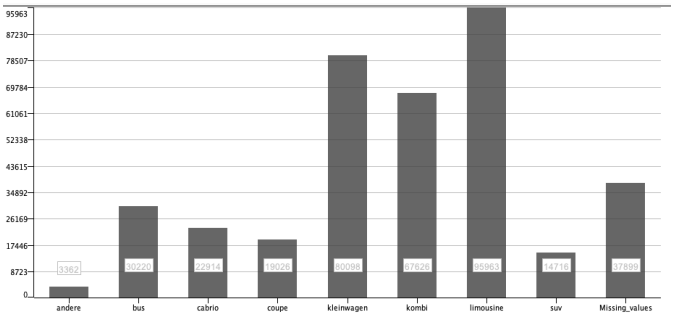

Figure 3: Cars Type Distribution

Regarding the cars fuel distribution, shown in Figure 4, *"gasoline"* is the most frequent type of power supply whereas *"hybrid"* or *"electric"* cars are very rare, probably due to the age of the dataset which does not include the latest generation cars (which are often hybrids).

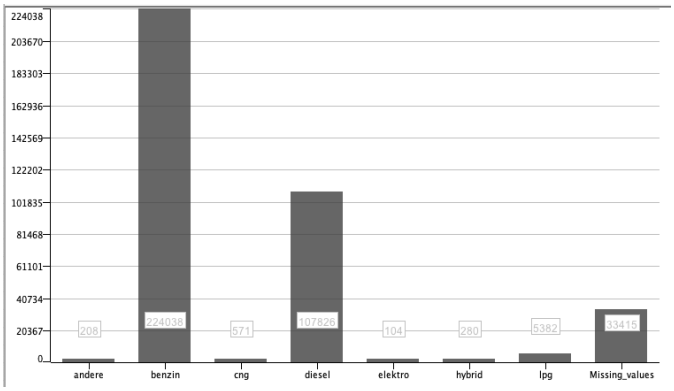

Figure 4: Cars Fuel Distribution

The most popular brand of cars in the dataset is *"Volkswagen"*, as shown in Figure 5, and the top 5 is all occupied by german-made cars, this is expected since the dataset is from a german website.

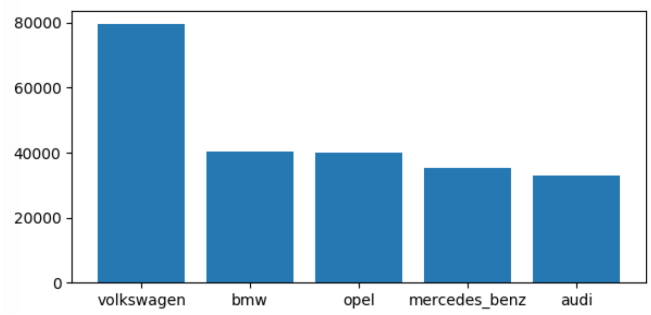

Figure 5: Top Cars Brand Distribution

Taking a look at the *"price"* attribute, shown in Figure 6, the target of our task, it is possible to see that there is a clear relation between *"price"* and *"year of registration"* of the car; this is shown in the next figure. It is clear how newer cars tends to cost more than used ones.

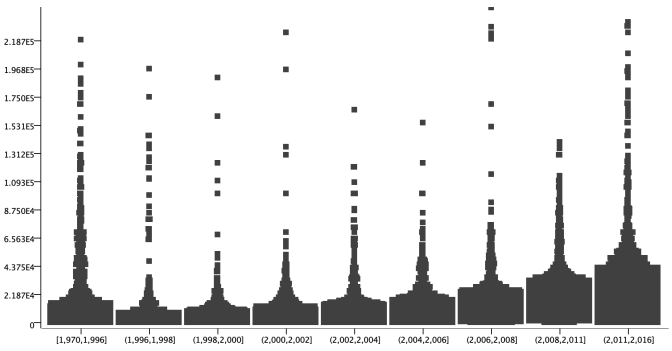

Figure 6: Year Range (X) vs Price (Y)

The next two figures are useful to understand if there is some kind of connection between the vehicle type and its price and between the brand and its price.

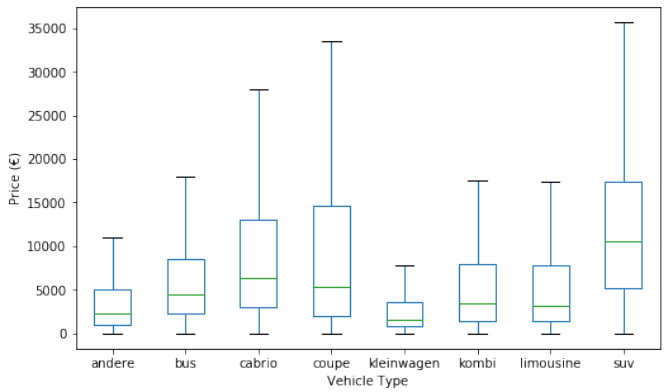

Figure 7: Vehicle Type (X) vs Price (Y)

In Figure 7, it is possible to see how the *type of a vehicle* influences the price in different ways. For example, *"suvs"* have an higher median price than *coupe cars*, while *utility cars ("kleinwagen")* usually have lower prices in general.

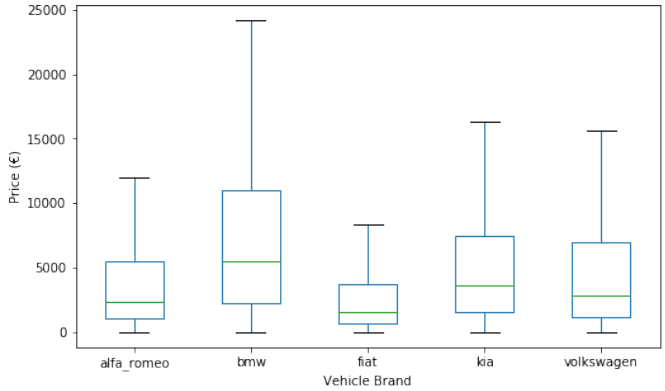

Figure 8: Vehicle Brand (X) vs Price (Y)

Taking a look at some selected brands in Figure 8 it is possible to see how the different brands introduce variations in the distribution of the price. In particular, as expected, *"BMW's"* have higher listed prices, while *"Fiat"*, have lower ones; this means that having the brand is definitely an useful information.

Overall it is shown how this dataset **contains very dirty and full-of-outliers data**, but also **useful information** to predict the price of a car. During the preprocessing phase information gathered in this part will have to be kept in consideration.

V. Pre-Processing & Data Cleaning

Before using the dataset for the training and evaluation of a regression model, the data is **filtered and later preprocessed** in a data cleaning pipeline.

V.i. Feature Filtering

From the initial dataset the following attributes are removed:

- **seller**: for about 99% of the listings it is set as *"private"*;
- **offerType**: for about 99% of the listings it is set as *"sale"*;
- **nrOfPictures**: always set to 0;
- **postalCode**: not considered useful for our analysis;
- **dateCreated, dateCrawled, lastSeen**: not needed for our analysis;
- **abTest**: not needed or our analysis.

Before the preprocessing pipeline, the dataset consists of **12 attributes**.

V.ii. Preprocessing

The first step in the pipeline tries to **infer missing car models** in the dataset: as seen before, the database has 20,500 missing values for the *model* attribute.
To attempt to retrieve some of the missing values for the *model* column, the *title* of the listing can be used as an hint. The retrieval is done by comparing each word of the title with a list of the unique models of cars, if a match is found, it is, with very high probability, the model of the car. For example, for a listing with title *"Fiat Punto 1200 Red New"* with a null *model* value (which in this case would correspond to *"Punto"*) and *"Fiat"* as *brand*, the model can be retrieved if other *"Fiat"* cars in the dataset have *"Punto"* as model attribute. Comparing all the *Fiat* models with each word of this particular listing's title, a match is found for the world *"Punto"*, which is then used as the *model* for the listing. The method is also applied for cars that had the model specified as *"andere"*, which, as already said, means "other" in German.
With this method about **10,000 models are recovered** out of 20,500 missing/unspecified, records with unrecovered *model* are discarded. After this step the listing title is removed from the available data.
Then next step consist in **removing the listings that have a brand specified** as *"sonstige_autos"* or a model as *"andere"*: these terms, in German, translates to "other". It is not possible to make any further assumptions if both the brand and the model are not available.
Another filter that is applied on the input data is **for listings of cars older than 1970 or newer than 2016**. In fact, for cars registered after 2016, it is sure that these are error because the dataset is last updated in 2016. Cars older than 1970 are instead filtered out because they mostly are typos and/or very specialized vehicles that are out of scope of this proposed general model.

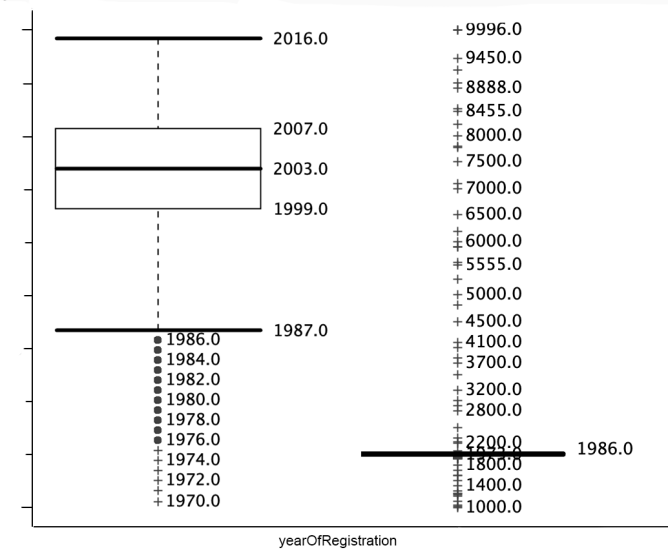

Figure 9: Registration Years after (l) and before (r) prepossessing - normalized

Another filter that is applied consists in **removing listings with a listed price lower than €100 or higher than €300000**. This filter allows to exclude from the dataset a lot of dirty data: it is found that out of this range a lot of records are clearly wrong user-submitted data. Either people inserting special values like 0 or 123456789 for the price. The upper bound of the filter also makes sense since that when trading higher valued cars, one should do some manual research and use some expert in the field.
Overall, the previously mentioned filters **removed about 60000** listings out of the 370000, around 15%.
However, although some filtering is applied for the boundary values of the price columns as a whole, there are still problems in the limit values of a single car model.
For example, for almost all the models there are listings with prices that are very unrealistic, both very low and very high. It is seen, for example, that some *"Porche 911"* are posted for sale for €1,000 and less, and some *"Fiat Punto"* for €90,000 and more. This are either typos or errors committed on purpose by the authors of the listings with the idea to draw more attention on the listing. Therefore to filter these problematic listings, some filtering is needed: all the cars of the same model (and brand) are grouped together and, after that, these groups are again subdivided in others **based on kilometers of the cars** (10 ranges of kilometers are defined for this grouping). This means that each group contains listings that have the same model, brand and range of kilometers. Then, calculating the median for the price of each group (and not the mean since the mean is not a robust statistic and therefore not resistant to outlier prices), it is possible to **calculate which cars are different by which relative amount to the median price of the group**.
In particular, it is decided to remove all the cars with a relative price difference respect to the median of the group higher than 2, because, based on our intuition, they are incorrect data. The value of 2 is chosen as a good trade-off since lower values would filter many valid listings, while higher values would still keep many listings with very anomalous data for the price. This step **removed about 18,000 listings**, leaving 311,166 listings to be available for later preprocessing steps.
In the next step the ***powerPS* attribute**, indicating the horsepower of the car being sold, **is set as missing** for listings that had it set to 0 and to over 500 (mostly fake and/or unreliable listings). The missing values of *"PowerPS"* are then later calculated as the mean of the *"PowerPS"* values of the listings having the same brand and model as the one with the missing value.
For example, the missing *powerPS* of a listing of a *"Fiat Punto"* is calculated as the mean *"powerPS"* of all the listings of *"Fiat Punto"*.
After this step, the listings that have *"Vehicle Type"* set to *"andere"* ("other" in German) have been set as missing values, in order to be recovered later.
The attributes *"RegistrationMonth" and "RegistrationYear"* are also **merged in a new attribute called *"ageMonths"*** which represents the months elapsed since the car's registration, calculated as $ageMonths = ((2016 \cdot 12) + 4) - ((RegYear \cdot 12) + RegMonth)$, 2016 and 4 since the dataset is from April of 2016.
The attribute *"notRepairedDamage"*, when missing, has been **set to the most frequent value**, *"no damage"*, by the assumption that damaged cars are usually noted as such, in the best interest of the buyer and the seller.
To replace the missing values for the attributes *gearbox*, *fueltype* and *veichleType*, **the most frequent value is inserted**, based on listings of the same brand and model.
Finally, **the *price* attribute is converted in logarithmic scale**, this has to be done since the price range is very large and its distribution is non suited for some machine learning models.
To conclude, the dataset that is used for further experiments **consists of 292,167 records**, all without any missing value, since all data are either present from the start or imputated through the preprocessing pipeline.

V.iii. Correlation Matrix

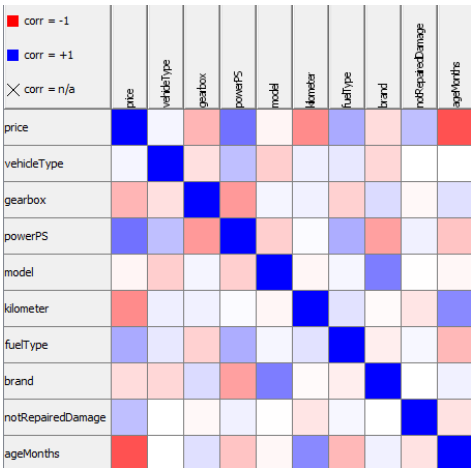

Figure 10: Correlation Matrix after Prepossessing

In Figure 10 the **correlation matrix** obtained after the preprocessing is reported: the target variable, *price*, is highly inversely correlated with the *age of the car*, as expected, since newer cars tends to cost more. The *power of the car*, instead is directly correlated with the *price*: more powerful cars cost more.

Finally, there is a small but important correlation between the *kilometers (inverse) and price*, as well as the *gearbox (inverse) and price*.
It is interesting to note that *vehicleType*, according to the table, does not correlate with the *price* at all, probably because it is paired with other variables to influence the price.

VI. Objective and Evaluation

The aim of this work, as specified earlier, is to **predict the price a used car is selling for**, starting from the various available attributes of each listing.
The task is modeled as a regression problem, thus it is possible to use the **standard metrics for evaluating these problems**.

After fitting a regression model, is has to determine how well the model fits the data **using goodness-of-fit statistics**. In general, a model fits the data well if the differences between the observed values and the model's predicted values are small and unbiased. The first goodness-of-fit measure, used to evaluate the performance of the proposed models is the R^2 **coefficient**, the coefficient of determination or the coefficient of goodness of fit. The R^2 coefficient measures the **closeness of the data to be predicted** in respect to the fitted regression line. More formally, it is the percentage of the response variable variation that is explained by a linear model. The R^2 coefficient varies between 0 (0%) and 1 (100%): a value of 0 indicates that the model explains none of the variability of the response data around its mean, while vice-versa a value of 1 indicates that the model explains all the variability of the response data around its mean. In general, higher values of R^2 indicates a **better fit on the data**.
However, R^2 has also some key-limitation: it cannot determine whether the coefficient estimates and predictions are biased, and it does not indicate whether a regression model is adequate; reason why, for a more detailed analysis, R^2 coefficient **has to be compared to other metrics**.
In a more general way, R^2 does not give any direct insight on the meaning of the predictions and their values.

To further evaluate the performance on the task **Mean Absolute Error, or** MAE, is also used.

$$MAE = \frac{\sum_{i=1}^{n} |p_i - e_i|}{n}$$

where n is the number of records to be evaluated, p_i the predicted value for the target variable and e_i the expected variable for the target variable.
MAE is the simplest regression metric to understand and it **calculates the residual for every data point**, taking only the absolute value of each so that negative and positive residuals ($p_i - e_i|$) do not cancel out. It then takes the average of all these residuals. Effectively, MAE **describes the typical magnitude of the residuals**.
A small MAE suggests that the model is **great at prediction**, while a large MAE suggests that the **model may have trouble in certain areas**. Therefore, small MAE is better.
MAE has also the benefit to **downpay the outlier's significance** and this is of great importance since in our case of interest the presence of outliers is possible given that the dataset has been compiled by users.
In the end, this measure is useful to get a global understating on the **average error the model commits on a listing**.

Furthermore, is used the **Mean Absolute Percentage Error (MAPE)** which is the percentage equivalent of MAE. Like before but with adjustments to convert everything into percentages:

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} |\frac{p_i - e_i}{p_i}|$$

MAPE is, in percentage, **how far the model's predictions are off from their corresponding outputs on average**. This is a very interesting metrics since it gives a way to **estimate quantitatively how well the model is doing**.
One of the main issues with MAPE is that it **biased towards predictions that are less than the actual values**. That is to say, MAPE will be lower when the prediction is lower than the actual compared to a prediction that is higher by the same amount.

Other existing measures to evaluate the model's performance for regression tasks, are not strictly suitable for this kind of task. In particular, it is not possible to trust the Mean Square Error and Root Mean Square Error measures, since, by squaring the error, it would **penalize more high-value errors** which are usually correlated with high-value cars, for which an exact price (to the hundreds of € for example) might be difficult to define and most of the times the bargain window is bigger, so the price on the listings could be inflated.
It could be possible to use **Root Mean Square Logarithmic error measure**, which is suitable when **not wanting to penalize bigger errors in respect to smaller ones**. It is decided to not use it since it's value is not really directly understandable by a human and also difficult to compare and comment.

In the end, R^2, MAE and $MAPE$ are used to evaluate the performance of the predictive models. The various metrics are calculated on the linear scale price, and not on the $\ln(price)$, thus the predictions are first transformed from $\ln(price)$ to a linear price and then evaluated.

VII. Proposed Models

In this section the **machine learning models** that will be compared and used in our analysis are presented.

VII.i. Standard Machine Learning

First **5 machine learning models** are compared on the regression task, listed in increasing complexity:

- **Simple Linear Regression (WEKA)**: which uses only a single numerical feature (the one yielding the highest R^2) from the dataset to obtain the prediction for the target variable;
- **Linear Regression**: which uses all (but the model) features to predict the price using linear regression;

- **Simple Regression Tree**: which uses all the feature to build a regression tree;
- **Random Forest Regressor**: which builds an ensemble of 20 trees to predict the price value from all the attributes of the dataset;
- **Gradient Boosted Regression Trees**: which uses 100 very shallow regression trees and a special form of boosting to build an ensemble of regression trees from all the attributes.

The following considerations are necessary: for the Linear Regression model it is not possible to use the *"model"* attribute due to a limitation of *Knime*, since the *"model"* attribute has **too many different discrete values**. Binarization and PCA on the attribute was attempted but it only **made performance worse**. For the Simple Linear Regression model only numerical features were used.

VII.ii. Ensemble-like Machine Learning

At the base of this section there is the intuition that **building multiple specialized models**, one for each subset of the listings, can yield better performance than a single model for all the listings. In particular, the idea is to first categorize the cars into a "Cheap", "Medium" and "Expensive" category and then to use a specialized model for the selected category for the regression task.

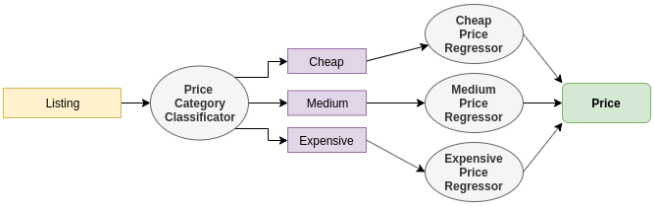

Figure 11: Overview of the model

To build this ensemble model the first step is to build a classifier that predict if a car is "Cheap", "Medium" or "Expensive". To do so, a new attribute is added to the dataset, called *"Price [Binned]"*, with the following binning range:

- **Cheap**: for prices in the range $]€0 - €10000[$
- **Medium**: for prices in the range $]€10000 - €50000[$
- **Expensive**: for prices in the range $]€50000 - €\infty[$

To predict the price range of a listing, a *Random Forest predictor* is used, with 100 internal trees specified as hyper-parameter.
For each price range a different *Gradient Boosted Tree Regressor* is used, with 100 trees specified as hyper-parameter. The aim of this regressor is to predict the price of a particular listing, the same as the machine learning presented before, but this time each model is "specialized" in a certain price range.
Thus, the training process for this model first trains the *"Price Range Predictor"* and then, after splitting the training data according to the price range, **trains each model with the corresponding training data**.
For predicting the price of a listing first the listing is classified by the price range predictor, and then, according to the classification, the listing is **passed to the appropriate regressor**, where the price is predicted.

VIII. Performance Evaluation

To evaluate each model two standard procedures are used:

- **Holdout**: in which the dataset is split in 70% training and 30% testing data, to simulate a real-world usage in which a single model is built and used. Performance are evaluated on the test set.
- **10-fold cross validation**: where a standard *10-fold cross validation* procedure is carried out. The overall performance are the average of the performance obtained from the test set of each fold.

In both cases the dataset is split by stratifying according to the *"model"* variable, with the intuition that the model distribution would look similar on training (experimental) and test (real-world) usage.

VIII.i. Holdout

For the **holdout evaluation** 70% of the dataset is used for training while the remaining 30% for testing the models. The **same partitioning** of the data is used for all models, ensuring fair performance comparison.

Standard ML Models.

Model	R^2	MAE (€)	MAPE
Simple Linear Regressor	0.152	3312	1.308
Linear Regressor	0.504	2046	0.716
Regression Tree	0.823	1252	0.481
Random Forest Regressor	0.827	1347	0.491
GB Regressor	0.845	1170	0.445

Table 4: Holdout Performance Metrics

As it can seen from table, the **worst model** in terms of the various metrics is the *Simple Linear Regressor*. This is expected since it using only a **single numeric variable** to build the regression model instead of all the attributes. Namely the regression line that is found by running the model is:

$$\ln(price) = -0.01 * ageMonths + 9.41$$

This means that the *ageMonths* attribute yields the best R^2 score.
The formula is presented as $\ln(price)$ since in the preprocessing phase the price is transformed in logarithm form, as said before.

The Linear Regressor model, yields a R^2 score of 0.5 and a $MAPE$ of 0.72. This means that on average the model estimates about 70% more or less than the real value of a listing. Again, from the Linear Regressor **great performances** can't be expected, it doesn't use the *"model"* attribute and a *simpleregressionline* probably can't fit data as complex as the one in the dataset.

The Tree-based Regressors instead achieve **much better performance** than the Linear Regression ones. They are able to make choices that are **"smarter"** than a simple regression line. Overall all 3 tree-based regressors **achieve the same performance**, with small differences in term of

R^2 and more substantial in term of MAE and $MAPE$, but which are not statistically interesting.
Finally, by raw numbers, the best model, which is also the most expensive in term of computational time is the one based on **Gradient Boosted Regression Trees**. The Gradient Boosted Regressor obtains a $MAPE$ of 44%, meaning that on average the model commits an error of 44% on predicting the price (price is 44% higher or lower than the real one). It can be observed that increasing the complexity of the model, at least for this particular situation, usually yields to better performance.
In conclusion, it can be said that there is still space for improvement, especially for a real-world usage: when predicting a price the error should be as small as possible, or at least not in the range of 40% of the price.

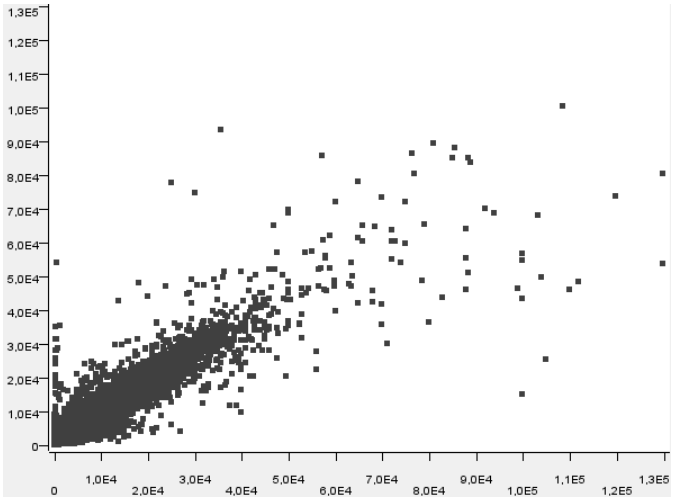

Figure 12: Actual Price (x) vs Predicted Price (y)

In Figure 12 is presented a **scatter plot** in which the actual price (on x-axis) and the predicted price (on y-axis, by the Gradient Boosted Regressor) are compared. Most of the points are distributed near the $x = y$ line, but a few more substantial errors are observable, especially as the price for a car increases.

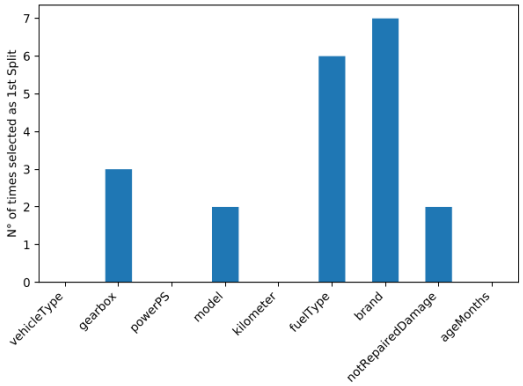

Figure 13: Number of times an attribute is selected as first split in a decision tree of the Random Forest of 20 trees

In Figure 13, it is possible to observe the number of times a certain attribute is selected as the first split for each tree in the Random Forest Regression method, this plot is interesting since it will allow to see which attribute is the one providing the **highest information gain** in respect to the others. Of course, it is important to remember that in random forest not all the attributes are available for each tree in the forest, but this will still give a general idea on the importance of the attributes.
The attribute that is used most of the times as first split is the *brand* one, followed closely by the $fuelType$ attribute. *Brand* is definitely an expected choice, some brands are more expensive then others and using it as first split is a good idea to introduce a first partition in the data.
More interesting to note is the splitting on $fuelType$, because it isn't expected to be such an important factor in reducing the entropy of the data as much as being selected so many times as a a first split.

Ensemble-based Predictor Evaluation.
For the ensemble-based predictor, both steps of the prediction pipeline have to be evaluated: the first classifier that predicts if a listing is either "Cheap", "Medium" or "Expensive" and the sequential price predictors built for each price range.
For the first classifier, the **confusion matrix** will be evaluated as well as the standard metrics for a classification problem (accuracy, precision, recall and F-measure).

Real \Predicted	Cheap	Medium	Expensive
Cheap	73020	1034	3
Medium	1709	11627	26
Expensive	4	71	159

Table 5: Confusion Matrix of Price Range Classifier

Analysing the confusion matrix, most mistakes (in absolute measure) are for the "Cheap" and "Medium" class, many "Cheap" listings gets classified as "Medium" or vice-versa. This is expected since the line between the two categories is very thin, practically speaking. It is also interesting to note that a lot of expensive cars get predicted as medium. This is reflected in the numeric measures table, which can be observed in Table 6.

Category	Precision	Recall	F-Measure
Cheap	0.977	0.986	0.982
Medium	0.913	0.870	0.891
Expensive	0.846	0.679	0.753
Overall Accuracy		0.968	

Table 6: Performance of the Price Range Classifier

The classification model obtains the **worst performance** in term of *F-measure* for the *Expensive class*. This is probably due to the fact that there are very few examples of "Expensive" listings when compared to the other classes (tens of thousands vs a few hundreds). Overall, the performance of the model are satisfying, achieving an **average accuracy of 0.96**, but some problems can be expected especially with the expensive class.

Then, the performance for the three exact price regressors are evaluated, first one by one and also by merging the predictions in a single result and re-evaluating the performance, just like the simple ML models.

Price Regressor	R^2	MAE (€)	MAPE
Cheap	0.732	830	0.419
Medium	0.731	2554	0.424
Expensive	0.48	13653	2.386
Overall	0.89	1108	0.424

Table 7: Performance of the Price Regressors

Overall, by combining the results and re-computing the metrics, better results are obtained in respect with the simple ML models. In particular, an higher R^2 score and a lower MAE and $MAPE$ scores are achieved. Quantitatively the MAE is just a few tens of euros lower than the simple Gradient Boosted model, thus the **difference is not particularly significative**. The same goes for the two other metrics.
By looking at the single regressors, the expensive cars regressor obtains the **worst performance** overall, probably due to the least amount of listings to be able to train from and due to the difficulty of the task.
The Cheap and Medium regressor obtains **similar results** in term of $MAPE$ and R^2 (but of course the MAE is lower for the Cheap regressor, since it is dealing with lowered-value listings). Again, even if using a much complex model, there is still room for improvement.

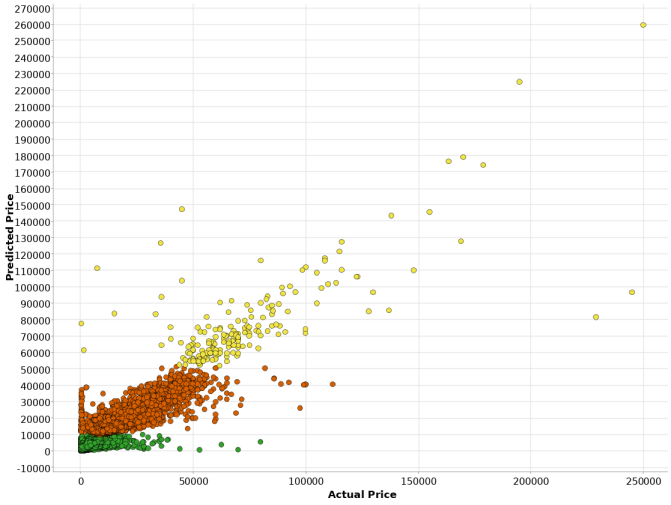

Figure 14: Predicted Price vs Actual Price - Color is Price Category

Taking a look at the above scatter plot which compares the predicted price and the actual price (Figure 14), it can be seen a **drawback of the proposed model**, made especially clear by the color-coding. Since the regressors are based on Regression Trees, the predicted price is not based on a regression line equation, thus the regressors are only able to output values in the range of prices seen in the training data. This is made clear since there is no overlap between the various color-coded areas.
A possible solution to this issue would be to **use real regressors**, based on building a real line equation, but this is not investigated since it proved bad performance in the simple section for the ML models.

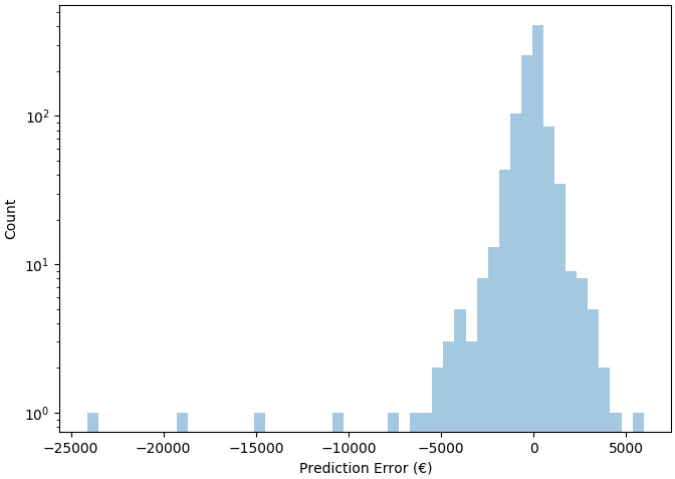

Figure 15: Prediction Error Distribution

Taking a look at the error distribution for the ensemble-based model it can be seen that most errors are concentrated in the range of € $\pm$ 2500. This plot is useful to **evaluate the qualitative performance** of the price predictor.

VIII.ii. Cross-Validation

To further evaluate the models, **a standard 10-fold cross validation procedure** is carried out. The same training and testing procedure is executed, but each time a different fold of the dataset is used for testing while the other 9 are used for training. This procedure is useful to evaluate the models in all the conditions presented on the dataset and to ensure that when performing the holdout-based evaluation, the results were not obtained by pure luck.

Standard ML Predictors.

Model	R^2	MAE (€)	MAPE
Simple Linear Regressor	0.136	3339	1.333
Linear Regressor	0.497	2072	0.737
Regression Tree	0.845	1165	0.476
Random Forest Regressor	0.82	1382	0.49
GB Regressor	0.85	1171	0.45

Table 8: 10-Fold-CV Performance Metrics (Average)

The results in Table 7 are **similar** to the ones obtained for the holdout method, being almost the same for some of the models. This confirms that the results obtained by the holdout method were not gained by a lucky partitioning in train and test sets. As before, the worst performing model is the *SimpleLinearRegressor* due to its configuration, while **the best one is the Gradient Boosted Regressor**, this time together with the Regression Tree approach, which tied in terms of R^2.
Figure 16 gives a graphical representation of the performance of the various models during cross validation.

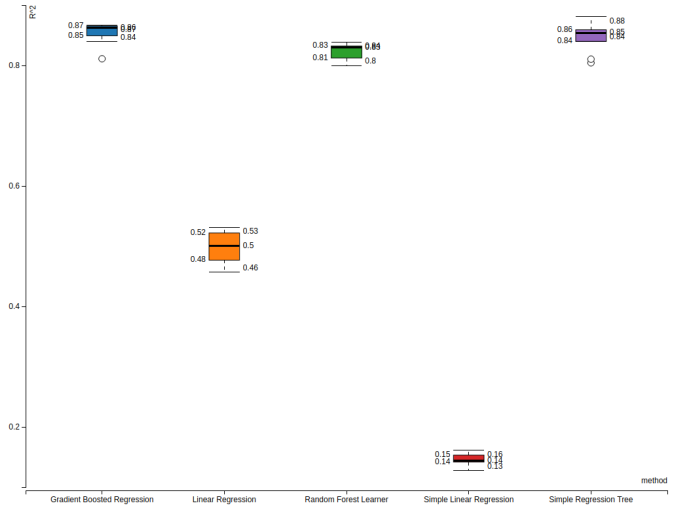

Figure 16: 10-Fold-CV: R^2 score distribution on testing folds

Ensemble-based Predictor.
Looking at Table 8, which shows the performance of the classifier which is the first step of the complex regression pipeline, it is possible to see that **the results are similar to the one of the holdout method**. Still, the most difficult class to correctly predict is the one of expensive cars, again, probably due to the smaller size of the partition of the data. Thanks to cross validation there is now confirmation that it is just not an issue with an unlucky holdout partition.

Category	Precision	Recall	F-Measure
Cheap	0.977	0.987	0.982
Medium	0.917	0.872	0.894
Expensive	0.837	0.690	0.753
Overall Accuracy		0.968	

Table 9: 10-Fold-CV Average Performance of Price Range Classifier

R^2	MAE (€)	MAPE
0.891	1104	0.435

Table 10: 10-Fold-CV Compound Model (Average)

Taking a look at the performance measures for the 10-fold Cross Validation of the ensemble-based model (Table 9), it can be seen that **the results are similar to the one obtained with the holdout method**. Overall, this method is still the best in terms of all the different metrics. The mean absolute percentage error for this model is about 43%, the lowest seen so far for the cross validation procedure by a few percentage points.
The complexity added by the combined ML model presented in this paper only results in **less than an hundred euros average improvement** on the prediction, which is still an improvement from the holdout method, but not a big breakthrough.

IX. Conclusions

Overall it is possible to can say that for predicting used car prices, for this dataset, the **composed ML model** is the one performing the best in terms of R^2, MAE and $MAPE$ measures. If the complexity of the model is not sustainable for the execution environment, Gradient Boosted Regression trees can be used, by taking a small hit on the performance. Comparing to the work of [Pudaruth 2014], which a**chieved an R^2 score of** 0.86 on the same dataset but in more ideal conditions, without attempting to recover missing values and repairing malformed values, **our best non-complex regressor model achieves an R^2 of 0.85**, in more general conditions than the one by [Pudaruth 2014], while our **complex ML model achieves an R^2 of 0.89**. This means that, overall, the preprocessing work that is applied, add some improvements, as well as using the composed ML models improved the performance.
By itself the R^2 score is just a pure number but, to get a better understanding on the performances of the model, MAE and, in particular, $MAPE$. Unfortunately, this score is not available in similar papers that deals with this dataset, so comparison is not direct. With the achieved solution a best **MAPE of 0.435**, meaning that on average the error committed in the prediction task is of 43.5% of the original price. This result is not good enough for a real world usage of the model, but at the same time it gives an insight on the complexity of the problem that has been tackled. Even with the amount of data available, it is not possible to obtain impressive results.
Actually, for the case being, the various models could perform better with less data available but with higher quality in terms of outliers and errors. In particular it's important to remember that part of the complexity of the problem comes from the state of the starting dataset, an user-submitted dataset without additional cleaning and filtering at the source is all but ideal for this kind of problem. At the same time it can't be expected for such a model to achieve a $MAPE$ of $\sim 0\%$. At the same time, it's important to remember that the problem it is dealing iwth, is the used market, prices are not final and most of the time it is expected to bargain on the price. Two listing with the same model of the car, kilometers and conditions could have a different selling price depending on the personality of the seller, one could prefer to overestimate the price and lower it later, while another could just put his final fixed price.
In the end, it is possible to consider our research successful, since we have proposed a way to preprocess very dirty and outlier-rich data and we have showed how different models would perform on the problem.

IX.i. Future Developments

In future there are several improvements and experiments that could be applied to the proposed project, dataset and methodologies, especially since the results still have some rooms for improvement.
In particular:

- **Improve outlier detection**. The work deals with some of the dirty and outlier data in the dataset but still plenty are left in, causing worse results than the one achievable with an ideal distribution of data. Improving the outlier filtering and detection would result in better performance.

- **Improve validation of the combined ML model**. It could prove beneficial to the performance to experiment more with the regressor models for the combined, ensemble-like ML model, in particular trying to use other regression models instead of the GB Trees one.
- **Enrich the dataset**: having other and more detailed attributes on each listing (color of the car, general condition, accessories) would improve the accuracy of the model even more, as shown in the cited literature. This is not possible in this dataset since it access to the scraped source code of the web pages was not possible.
- **Directly comparing against other Kagglers**. *Kagglers* use different parts of the dataset, especially filtering the most difficult listings to classify and/or non-ideal conditions. Comparing directly against them was not the objective of this paper, since it would have limited the scope of the proposed solution.
- **Make it a classification problem.** As shown by some of the cited literature it is possible to see the regression problem in the case as a classification problem, since exact price to the last € is not important, but a range of price still is.

References

[Geg+19] Enis Gegic et al. "Car price prediction using machine learning techniques". In: *TEM Journal* 8 (Feb. 2019), pp. 113–118. DOI: 10.18421/TEM81-16.

[Kag19] Kaggle. *Used cars database*. 2019. URL: https://kaggle.com/orgesleka/used-cars-database (visited on 12/12/2019).

[NJ17] Kanwal Noor and Sadaqat Jan. "Vehicle price prediction system using machine learning techniques". In: *International Journal of Computer Applications* 167.9 (2017), pp. 27–31.

[Pal+17] Nabarun Pal et al. "How much is my car worth? A methodology for predicting used cars prices using Random Forest". In: *arXiv:1711.06970 [cs]* (Nov. 19, 2017). arXiv: 1711.06970. URL: http://arxiv.org/abs/1711.06970 (visited on 12/12/2019).

[Pud14] Sameerchand Pudaruth. "Predicting the Price of Used Cars using Machine Learning Techniques". In: 2014.

[Tec17] Technavio. *Global Used Car Market - Size, Projections, Drivers, Trends, Vendors, and Analysis Through 2021 by Technavio | Business Wire*. 2017. URL: https://www.businesswire.com/news/home/20170517006119/en/Global-Car-Market---Size-Projections-Drivers (visited on 12/12/2019).