

Università di Pisa

Artificial Intelligence and Data Engineering

Course: Data Mining and Machine Learning

**UsedVehiclesPricePredictor**

Authors:

Giulio Bello

Federico Frati

Index

[1. Introduction 3](#_Toc127175775)

[2. Dataset 4](#_Toc127175776)

[3. Preprocessing 6](#_Toc127175777)

[3.1 Missing values 6](#_Toc127175778)

[3.2 Outlier 6](#_Toc127175779)

[3.3 Price distribution issue 7](#_Toc127175780)

[3.4 Correlation analysis 8](#_Toc127175781)

[4. Regressor comparison 10](#_Toc127175782)

[4.1 K-Fold Cross-Validation 10](#_Toc127175783)

[4.2 Results 11](#_Toc127175784)

# Introduction

UsedVehiclesPricePredictor is a simple application that allows the user to ask for a price evaluation of its used vehicle.

This application can be used both by owners and sellers, the first to estimate the price and the seconds to verify if the deal is fair or not.

The key idea is to use different versions of a dataset to train some models and evaluate their performances. To do this, we select the most relevant features of the dataset using a pre-processing algorithm based on common sense, correlation between features, quantiles and z-score. This process is also useful to delete outliers.

The target of the regression is the attribute *price.* To have better performances is important to have a quite uniform distribution of prices in the range of values considered.

# Dataset

The data used is taken from the website 'craigslist.org' where various items, including used vehicles, are put up for sale. We used datasets relating to three different years: 2020, 2021, 2022. To be able to use all of them we had to standardize the formats, as there were small differences between the columns present in the three years.

In the dataset there are 1.326.489 rows.

After this process, the resulting dataset contained the following attributes:

* *Id*
* *url*
* *region*, the region in which the vehicle is put up for sale
* *region\_url*
* *year*, the production year
* *manufacturer*
* *model*
* *condition,* the condition of the vehicle [*savage, fair, good, like new, new, excellent*]
* *cylinders*
* *fuel*
* *odometer, miles travelled by the vehicle*
* *title\_status*
* *transmission,* the type of transmission [*manual, automatic*]
* *VIN,* vehicleidentification number
* *drive,* type of drive [*4wd, rwd, fwd*]
* *size*
* *type,* generic type of the vehicle
* *paint\_color*
* *image\_url*
* *description*
* *state,* the state in which the vehicle is put up for sale
* *lat*
* *long*
* *posting\_date*
* *price*

To standardize the data, it was decided to collapse the attributes 'year' and 'posting\_date' into a single attribute 'age', which is calculated as the difference between the year of 'posting\_date' and the value of 'year'.

By merging the three versions of the dataset, we obtain this distribution for the target attribute 'price':

Chart, histogram

Description automatically generated

Figure 1: initial price distribution

# Preprocessing

## Missing values

Exploring the dataset, the problem of missing values was subsequently highlighted. As is known, this problem must be handled during preprocessing.

We first eliminated the attributes that certainly did not influence the price or that had a considerable percentage of rows without value, namely: *'id', 'url', 'region\_url', 'VIN', 'size', 'image\_url', 'description'.*

To handle the absence of values in the “*condition*” attribute, the value "*good*" was inserted, which represented an intermediate evaluation.

Then we eliminated the occurrences that had even just one missing value in any of the remaining attributes.

The features listed before can be divided into two categories: categorical and numerical one. Categorical features are characterized by a finite set of possible value that can be assumed while numerical ones are characterized by a decimal value.

We decided to enumerate categorical features.

## Outlier

To identify which attributes were the most determining in predicting the price, the correlation between the individual features and price was calculated in three different ways: Pearson correlation, Spearman correlation, and Kendall correlation. For our decisions, we considered the Spearman correlation values because the latter is able to highlight even non-linear relationships. It is assumed that there are also relationships of this type in our dataset. It was seen that the features with the highest correlation were 'odometer' and 'age.'

In order to standardize the data, we decided to collapse the 'year' and 'posting\_date' attributes into a single attribute 'age', which is calculated as the difference between the year of 'posting\_date' and the value of 'year'. Both these columns and 'price' showed the presence of outliers within them. We then proceeded to remove these outliers to avoid damaging the performance. To do this, we removed the upper and lower deciles of price and odometer, while for age we only kept vehicles with a production date after 1990. On the remaining values, another test was performed, the z-score. This value is influenced by the mean of the values and the standard deviations of the same, and it is for this reason that it was done after a first more simple processing, in fact if we had calculated it earlier, the mean and standard deviations (and therefore the z-scores) would have been affected by the distribution of the outliers leading to results that are not completely reliable.

After these steps there are 349 705 rows remained into the dataset.

## Price distribution issue

Plotting the histogram of the attribute *price* we can note that the target column is unbalanced. So, we decided not to merge all the rows in the dataset, but only those belonging to a given range. In particular, from the 2022 dataset, we took all the rows, from the 2021 dataset, only the rows with a price less than 5000 or greater than 13000, and from the 2020 dataset only the rows with a price greater than 20000. This operation led to a significant balance, as we can see in the figure.

After these operation there are 221812 rows remained into the dataset.

Chart, histogram

Description automatically generated

Figure 2: price distribution after preprocessing

## 3.4 Correlation analysis

Now, in the dataset, we have all the initial features except for those that were manually eliminated because they clearly did not influence the price. In order to perform a dimensional reduction, it was decided to use the correlation coefficient.

The ideas behind this approach are two. The first is that two columns with a correlation tending towards 1 carry almost duplicated information. It was therefore decided to calculate the Spearman correlation coefficient between the various feature pairs and, if it was found to be greater than 0.9, to eliminate one of the two. To choose which one to eliminate, the correlation between both and the target feature 'price' was calculated: the one less correlated (considering the absolute values) was eliminated.

The second key idea is that it makes sense to keep only the features with a non-negligible correlation with 'price'. We therefore calculated the correlation of each feature with 'price' itself and decided to eliminate the column each time the value obtained was in absolute value less than 0.15.

# Regressor comparison

As we have already said, one of our goals was to find the best regressor to predict the price.

We then tested various models by evaluating their performance through various metrics: R-squared, Mean Squared Error, Mean Absolute Error.

The models that we analyzed are: Adaboost Regressor, M5Rules, K-Nearest Neighbors, Random Forest.

|  |  |  |
| --- | --- | --- |
| Adaboost | n\_estimators=100 | learning\_rate=0.3 |
| M5Rules | criterion="friedman\_mse" |  |
| K-NN | K = 5 |  |
| Random Forest | n\_estimators = 500 | max\_features=4 |

Table 1: model's parameters

## 4.1 K-Fold Cross-Validation

To have results that are not dependent on how the data was divided into test and train, we decided to use a K-Fold Cross-Validation (with K=10). For each possible division, we calculate all the metrics to verify the robustness and model generalization. After that, we trained the model using all the data.

For each model, the version used is available in the folder *Models*.

Chart, diagram

Description automatically generated

Figure 3: 10-Fold Cross Validation

## 4.2 Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Adaboost | M5R | 5-NN | Random Forest |
| R-squared | 0.592 | 0.861 | 0.673 | 0.912 |
| RMSE | 6558.21 | 3822.29 | 5870.39 | 3045.68 |
| MAE | 5219.52 | 1827.14 | 3737.27 | 1657.15 |

Table 2: metrics' results

As we can see in the table, Random Forest regressor is the best one considering the three metrics evaluated. Having a value of R-squared higher than 0.9 means that our regressor is able to explain the variance of the price in a quite precise way.

A mean absolute error of around 1600 dollars is an appreciable result, considering that we are predicting prices of vehicles. After preprocessing, the mean of the prices of the vehicles in the dataset is 19 049 $.

Considering Random Forest regressor, it has a mean absolute error of 1657.15 $ that is only the 8.7% of the average price.

Considering Adaboost Regressor, it has a mean absolute error of 5219.52 $ that is the 27.4% of the average price.

Considering 5-NN Regressor, it has a mean absolute error of 3737.27 $ that is the 19.6% of the average price.

Considering M5R Regressor, it has a mean absolute error of 1827.14 $ that is the 9.6% of the average price.

For the continuation of the discussion, we will consider only Random Forest and M5R, as they have shown to be significantly better in terms of score. The comparison we will make is related to the execution time of the two models and the memory space occupied by them. The results are available in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Load time | Execution time | Memory occupied |
| Random Forest | 30.50 s | 0.08s | 5 515 115 kB |
| M5R | 1.02 s | 0.002s | 15 531 kB |

Table 3: statistics on M5R and Random Forest

What can we conclude observing these data?

Random Forest has a higher accuracy score compared to M5R, but the latter has the great advantage of being much faster both in the model loading phase and in the prediction phase.

In the loading phase, M5R is about 30 times faster, while in the execution phase it is even 40 times faster. Which of the two models do we consider the best?

Given the type of service we need to provide, for the user it is more important to receive an accurate result rather than an instantaneous one. Sacrificing a portion of precision for a gain that is proportionally high, but practically quantifiable in a few seconds for the load and a few hundredths of a second for the execution, is not a choice consistent with the goal of the application itself. It should be considered that the appreciable and visible gain in terms of time, visible to the naked eye, is that related to the loading of the model, but this latter operation is only performed once. For these reasons, despite M5R's appreciable aspects, we feel that Random Forest is the better choice.