Comparative Analysis of Machine Learning Models for Secondary-Market Car Price Estimation

Paweł Wolny

Abstract

This paper evaluates the possibility of utilizing machine learning in secondary-market car price prediction. A dataset of more than 250 thousand samples, has been analyzed and preprocessed to serve as a base to train and fine-tune two models utilizing k-nearest neighbors and random forest regression algorithms. Afterward, both models were compared regarding performance using several different metrics on a previously unseen comparison test set. It was determined that for this particular dataset, the random forest regressor performed better than the k-nearest neighbors regressor for all the metrics.

1. Introduction

The majority of people at some point in their lives will purchase a car. Since it can be quite expensive and risky to get a brand-new model as the first vehicle, most drivers look for more affordable options on the secondary market. However, finding a good deal is not always easy if one does not have the technical expertise. A car is a very complex machine with many variables that define its actual value. Mileage, model, engine type, and year of production are just a few of the general factors that we have to consider, not even mention the more model-specific details that we have to be aware of. For a newcomer who just wants to get a reliable car at a decent price, it can be quite overwhelming.

In this project, an attempt was made to utilize machine learning methods to solve the problem introduced above. The initial idea was to train two different regression models to estimate the price in pounds given the car's specification using a car sales dataset that links the purchase price with various parameters for each car sample. The performance of the two models will then be compared and based on that appropriate conclusions regarding the more favorable approach to this problem shall be drawn. For this project, the latest Anaconda distribution of Python along with Jupyter Notebook has been used. The link to the repository containing the source code along with the dataset can be found in the references.

2. Exploratory analysis and preprocessing of the data

As the first step, an exploratory analysis of the dataset was performed to better understand the structure of the data, discover interesting correlations, and fix any potential imperfections. The dataset used in this project created by Huang et al. (2022) is a large-scale automotive dataset that among other data contains 268255 car advertisements for 899 car models that have been sold in the UK market over 20 years.

Importing the data

The original dataset consists of sixteen attributes, out of which five were automatically interpreted as numerical during import, and the rest were assumed to be of type object. In Table 1. the first five samples of the dataset are shown. Our target variable in this regression problem is the *price* attribute. Upon closer inspection, it was determined that some of the attribute types seem to differ from the types that were automatically inferred during import. Approach towards fixing this as well as other issues shall be addressed in the next subsection.

Initial preprocessing of the data

	maker	genmodel	genmodel_id	adv_id	adv_year	adv_month	color	reg_year	bodytype	runned_miles	engin_size	gearbox	fuel_type	price	seat_num	door_num
0	Bentley	Arnage	10_1	10_1$$1	2018	4	Silver	2000	Saloon	60000	6.8L	Automatic	Petrol	21500	5	4
1	Bentley	Arnage	10_1	10_1$$2	2018	6	Grey	2002	Saloon	44000	6.8L	Automatic	Petrol	28750	5	4
2	Bentley	Arnage	10_1	10_1$$3	2017	11	Blue	2002	Saloon	55000	6.8L	Automatic	Petrol	29999	5	4
3	Bentley	Arnage	10_1	10_1$$4	2018	4	Green	2003	Saloon	14000	6.8L	Automatic	Petrol	34948	5	4
4	Bentley	Arnage	10_1	10_1$$5	2017	11	Grey	2003	Saloon	61652	6.8L	Automatic	Petrol	26555	5	4

Table 1. Head of the dataset

To start with, the two unnecessary columns *genmodel_id* and *adv_id* were removed, since they do not contain any real information. Afterward, the attributes with the type assigned incorrectly, namely *engin_size*, *price*, and *runned_mile*s, were converted to the float type. In the first case, the issue was related to the trailing "L" for each of the column values, which had to be stripped before converting. In the remaining two cases, the problem was that some of the entries in these columns had a value of "Unknown", which was replaced by NaN values before the conversion.

Once all the attributes had the appropriate type, a boxplot was plotted for each numerical variable to visualize the distribution of the data and detect any potential anomalies. In Figure 2. some of the boxplots for which anomalies were observed are shown. These anomalies were deemed erroneous since they contradicted either basic logic or laws of physics. All of these incorrect values had to be replaced. The general approach was to first examine the anomalies and try to determine what was the likely cause of the errors. In the case of *engin_size* for example, it was simply due to the values being expressed in incorrect units. In other cases, the outliers seemed to be random human errors. In these cases, to replace the incorrect value either the median value for similar instances was used, or if there were too few similar samples in the dataset, an appropriate value was determined by looking up similar car models online. After all the evident anomalies were replaced, it was time to address the missing values across the attributes. The majority of the attributes had missing values that needed to be handled appropriately before proceeding any further. Starting with the price value, it was decided to remove all the instances where it is missing for simplicity. Such an approach has been chosen since the samples with missing price values

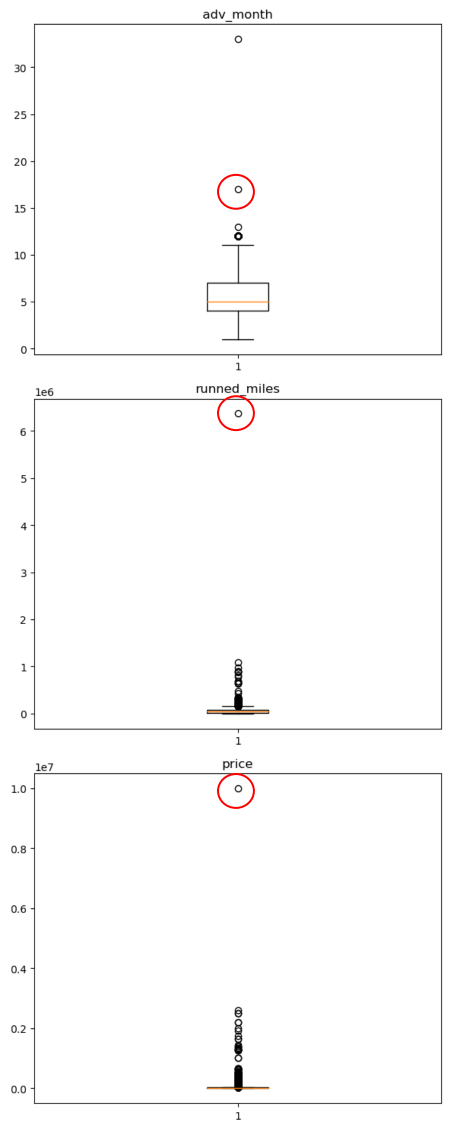

Figure 1. Boxplots for attributes with visible anomalies (in red circles)

account for only 0.4% of the dataset, hence replacing the values of the target variable would not yield that big of a benefit however, it could potentially introduce further complications. For the remaining columns with missing values, the following approach was taken. If we have an entry with a missing value, first we are going to look at all the matching car models with a similar price. If such models exist, their mode or median is imputed depending on whether the column is of categorical or numerical type. If there are no matching models with a similar price, then the mode or median of all the same models is imputed. However, if there are no matching models at all, the entry is removed from the dataset. In this way, all the remaining missing values in the dataset were replaced.

Visualizing the data

After the initial preprocessing of the dataset, several visualizations were performed to get a better understanding of the data's structure and potentially spot some useful patterns.

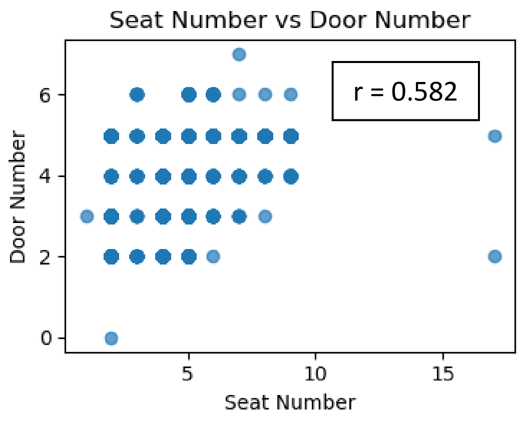

Figure 2. One of the scatterplots from the scatter matrix showing the correlation between seat number and door number along with their Pearson's coefficient.

One of the visualization methods that has proven to be especially informative was the scatter matrix for all the numerical attributes. In Figure 3. the scatterplot shows the correlation between the seat number and the door number. The Pearson's coefficient between these two attributes is quite high, which means that cars with more seats tend to have more doors and vice versa. Additionally, both seat number and door number seem to be negatively correlated with the price with the corresponding Pearson's coefficients of -0.139 and -0.141. Because of the facts above, an idea arose to combine both features into one by summing them together. This could potentially reduce redundancy in the data and the resulting feature could be more informative and have a stronger correlation with the price. After merging the attributes, the resulting attribute named *sum_seats_and_doors* has shown a correlation coefficient of -0.156 which is a slight improvement when compared to the original attributes.

3. Model selection and the training process

Model selection

After exploratory analysis and preprocessing of the data, the next step was to select the two machine-learning models that were going to be used. The first one was decided to be the k-nearest neighbors regressor since its working principle – averaging similar instances - is quite similar to what one usually would do to estimate the price of cars, houses, or other more complex assets. It would be interesting to see if an algorithm based on this intuitive principle would do well in this setting. The second approach would be to utilize a random forest regressor as it is robust, accurate, and easily interpretable method. It is also known to perform better with outliers compared to the aforementioned method. This might be crucial in the dataset as there are quite a lot of relatively rare and expensive samples that can be seen as the long tail on the right side of the histogram of the price attribute in Figure 3.

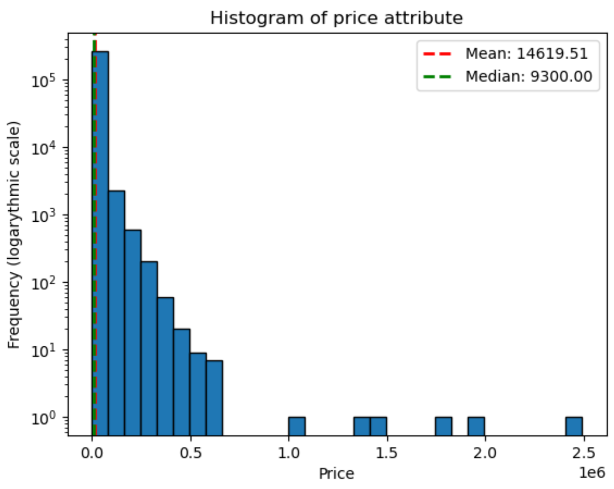

Figure 3. Histogram of price attribute with the mean and median marked. A long tail is visible on the right.

Additional preparations

For both of the chosen regression models, all the input data had to be of numeric type. For this reason, a method to convert all the remaining categorical attributes in the least invasive way possible had to be established.

```
maker            67
genmodel        773
color            22
bodytype         18
gearbox           3
fuel_type        13
```

Figure 4. Number of unique categories for each categorical attribute.

For the gearbox attribute, because of only three existing categories, the best choice was to use one-hot encoding, since it is simple and does not introduce any false assumptions or biases. However, for the other attributes, the same approach is not reasonable because adding so many columns would drastically worsen the computational efficiency and raise the risk of the curse of dimensionality. Another option would be to use ordinal encoding, however, this method would create an artificial ordering which could negatively affect the performance of the models. Finally, a method called target encoding was selected, in which the particular category is replaced by the mean of the target variable for all the instances sharing this category in the training set. In this way, the relationship between the categories and the target attribute is captured. Naturally, this method also has its drawbacks that were taken into consideration. A major flaw of this approach is the risk of poor generalization, especially in the case of categories with a small number of samples. Also when using this method one needs to be certain that the encoder is fitted solely on the training set to prevent information leakage. However, despite the drawbacks this still seemed to be the most reasonable choice to encode the categorical attributes.

In Figure 5. the encoding of the maker attribute on the training set can be seen and it seems to behave as expected. The more expensive sports brands such as Lamborghini, Ferrari, or McLaren visibly stand out from the others. Also, the premium brands do have larger encodings compared to the regular ones. However, this plot draws attention to an important concern which is the case when the samples of a particular category are so rare that they are not included in the training set during the splitting and thus they are not encoded.

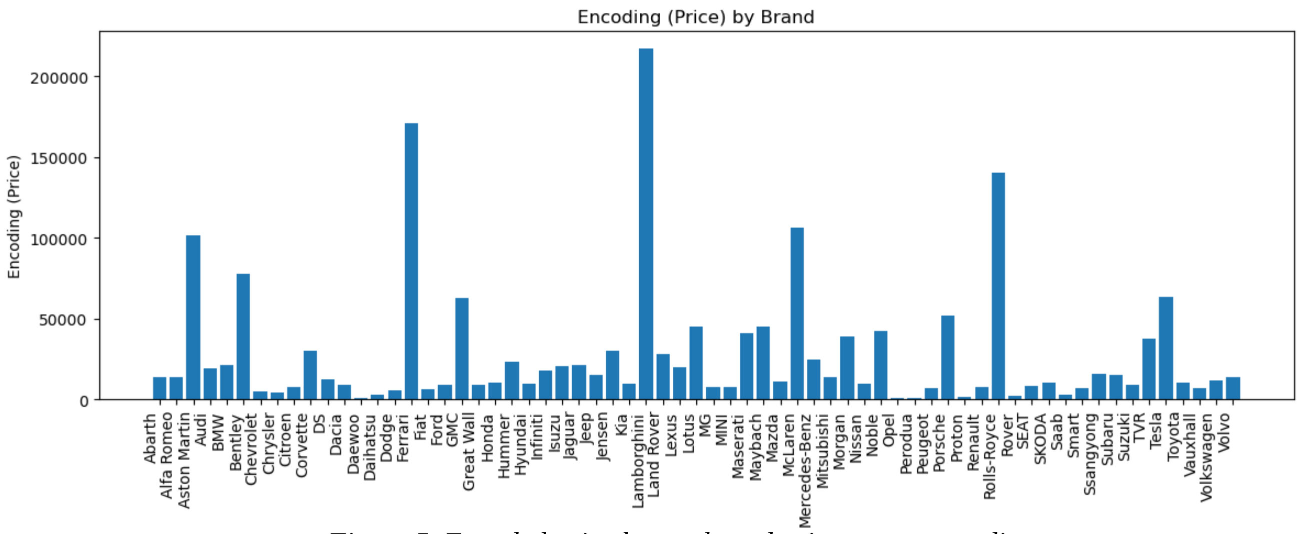

Figure 5. Encoded price by car brand using target encoding.

Tests have been performed to examine this and the scikit-learn target encoder implementation encodes unknown categories with the mean of the target variable. This will significantly decrease the accuracy when it comes to rare and expensive car models.

After encoding the categorical variables, the remaining step was to standardize all the attributes to diminish the impact of different scales of the attributes. This step is mandatory for the k-nearest neighbors regressor to ensure proper performance. When it comes to random forest regressor, this step is not necessary and apparently can even slightly worsen the accuracy (Filho, 2023).

The train, validation, test split

After all the necessary preparations had been made, it was time to split the dataset into three parts, namely the train, validation, and test sets in 0.6 to 0.2 to 0.2 proportions. The test was supposed to be used for training, the validation set for hyperparameter tuning, and the test set for evaluating and comparing the models. Since the dataset is quite large, the default random strategy was used.

Hyperparameter tuning

To ensure decent performance on the test set, both models were optimized using cross-validation on the validation set. The number of folds was set to three which is smaller than it should usually be, however, due to the size of the dataset, the available computing power, and limited time, such a compromise had to be made. For the error metric, the default mean squared error has been selected.

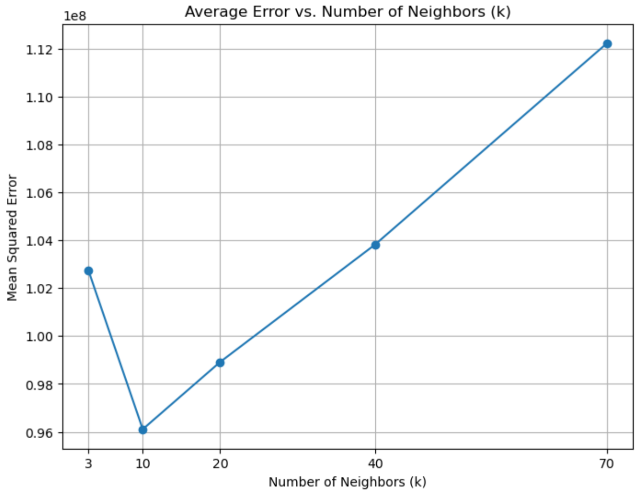

Figure 6. A graph relating the MSE with the number of neighbors

For the k-nearest neighbors regressor, the main goal was to tune the number of neighbors labeled as k. Evaluating the performance on the validation set for a wide range of k values, resulted in the plot visible in Figure 6.

After analyzing the graph, the test was run once again for a smaller range of k, and through that, the number of neighbors was set to a near-optimal value of 6.

For the random forest regressor, both the number of estimators and max depth were tuned. To find the best combination of these two parameters, a test evaluating a wide range of combinations has been performed. The results shown in Figure 7. The lowest error is observed for a combination of max depth and number of estimators being 50 and 50 correspondingly. However, with such a big

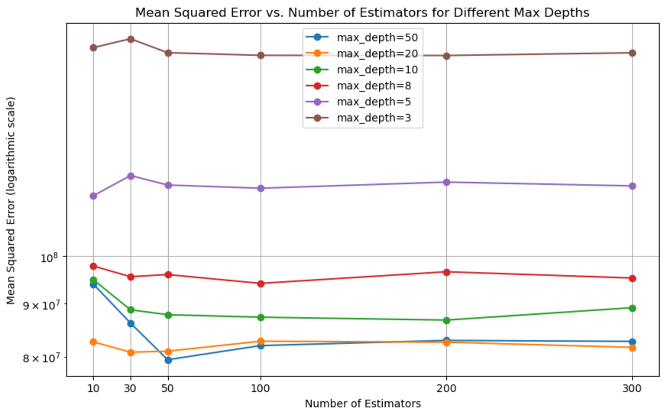

Figure 7. A graph with several curves for different values of max depth showing the relationship between the number of estimators and MSE

depth, there is quite a big risk of overfitting, and hence by Ockham's razor principle, it was decided to decrease the depth to 10. The error with the decreased depth was slightly higher but the risk of overfitting was greatly reduced.

Evaluation and comparison

After tuning the hyperparameters, the two models have been trained on the training set and evaluated on the test set. In Table. 2 both RMSE and MAE error metrics are presented for both models.

	RMSE	MAE	MAE (relative)
KNN	10725	2805	0.295
Random Forest	6990	2534	0.284
Baseline Model (Mean)	21998	10840	2.119

Table 2. Error metrics evaluated for the predictions of both models compared with the baseline

It can be seen that in the first two metrics, the random forest outperforms the k-nearest neighbors. However, the difference in the mean relative absolute error, where the absolute error is divided by the actual price, the difference is not as significant. Also, it can be seen that both models perform way better than the baseline model.

After comparing the basic error metrics, a significance test has been performed to determine if the obtained results are statistically meaningful. The method of choice was the one-sided t-test, with the p-value threshold of 0.01 and the null hypothesis being that the mean absolute error of the k-nearest neighbors regressor is smaller or equal to the mean absolute error of the random forest regressor. This method assumes that the error data is normally distributed. After plotting histograms of the absolute errors for both models, it turned out that both had highly right-skewed distributions. To fix this a logarithmic transformation was applied to all the error records of both models, which made the distributions more normal.

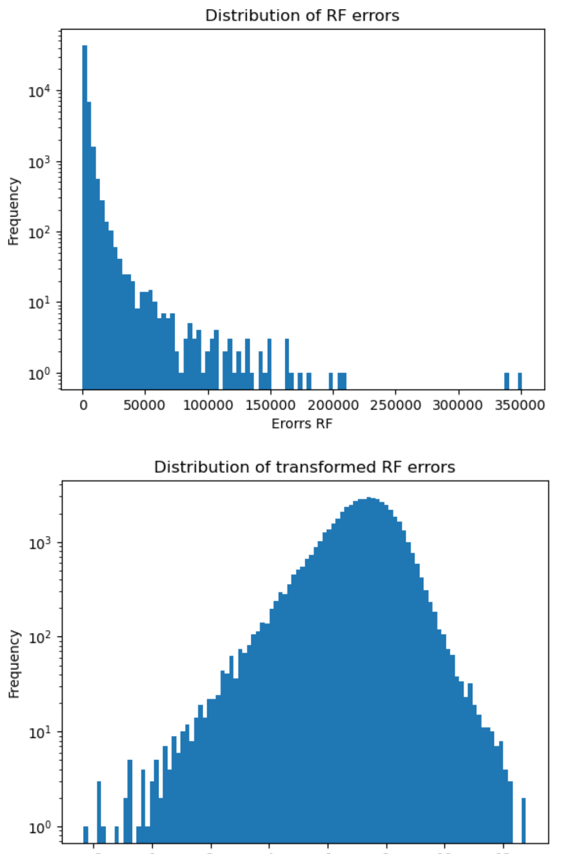

Figure 8. Two histograms representing the absolute error distribution for random forest before (top) and after (bottom) the transformation

After the transformation, the t-test has been performed and it resulted in $p < 0.01$, which means that the null hypothesis is rejected, and the data indeed shows that the random forest is

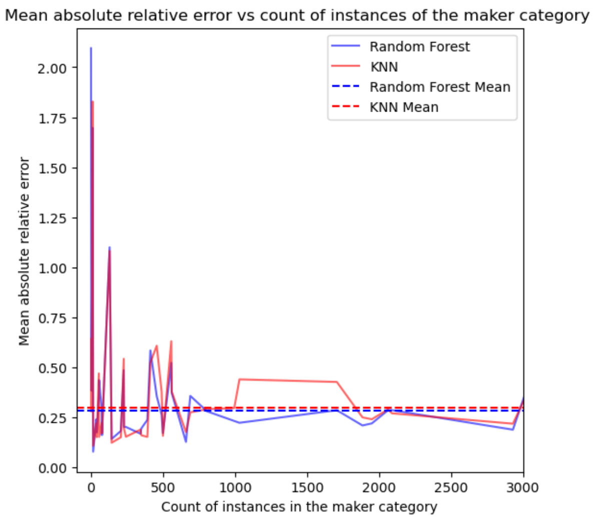

Figure 9. A plot showing the correlation between the number of samples of a given brand in the training set and the mean absolute

significantly better than the k-nearest neighbors regressor. A similar significance test has been performed for the relative error which also resulted in $p < 0.01$.

Last but not least, a visualization has been made to see how both of the models perform depending on the number of samples of a particular car brand in the training set. As can be seen in Figure 9, for both models the relative error is significantly greater for brands with few samples in the dataset. Interestingly, the random forest regressor seems to perform similarly to the k-nearest neighbors regressor in this case.

4. Conclusions

Results of the comparison show that it is feasible to utilize machine learning for car price prediction. In this task random forest regression seems to perform slightly better than the k-nearest neighbors regression for this particular dataset. What could be improved in the future is backing up important design choices with appropriate experimentation and significance checks. Also, a more extensive fine-tuning of the hyperparameters could be performed to get even more optimal and representative results.

5. References

Repository with the source code: https://github.com/pawel-t-wolny/car-price-predictor

Bibliography

Jingming Huang, Bowei Chen, Lan Luo, Shigang Yue, and Iadh Ounis. (2022). "DVM-CAR: A large-scale automotive dataset for visual marketing research and applications". In Proceedings of IEEE International Conference on Big Data, pp.4130–4137

Munoz, J. J. (2024, February 5). Encoding Categorical Variables: A Deep Dive into Target Encoding. Medium. https://towardsdatascience.com/encoding-categorical-variables-a-deep-dive-into-target-encoding-2862217c2753

Filho, M. (2023, June 29). Does Random Forest Need Feature Scaling or Normalization? Forecastegy.com. https://forecastegy.com/posts/does-random-forest-need-feature-scaling-or-normalization/