Price Prediction of Motorcycles using an Ensembled Machine Learning Model

Tabea Hacheney

tabea.hacheney@tu-dortmund.de

My GitHub Repository

Due Date: 31st of July 2024

Contents

1. Motivation

When making buying decision for items such as clothing, a new CPU or even a motorcycle, it is essential to have a clear understanding of an appropriate price. This might be more or less easy for common items like food and clothing, but a quite challenging task for more complex and expensive things, like houses, rare **PoKéMoN** cards or motorcycles. A potential buyer is almost obligated to invest a significant amount of hours to comprehend an item's price and to check if it is appropriate relative to the market.

This is a very tedious task. Using Machine Learning algorithms, up-to-date market prices and appropriate attributes, which influence the price, one can get an appropriate price with minimal time and effort (*at least for the end user*). When the author of this report was trying to sell her last motorcycle on the internet, she was faced with the dilemma of determining the highest possible profit without seeming unreasonable. This dilemma serves as a basis for the following report, in which an ensembled model of multiple regressor models will be used to estimate an appropriate market price based on factors such as age, mileage, brand, model, and respective bike specifications.

The python libraries used for the programming part of this report consist of **sklearn**[1], **pandas**[2], **numpy**[3], **matplotlib**[4], **seaborn**[5], **catboost**[6] and **xgboost**[7].

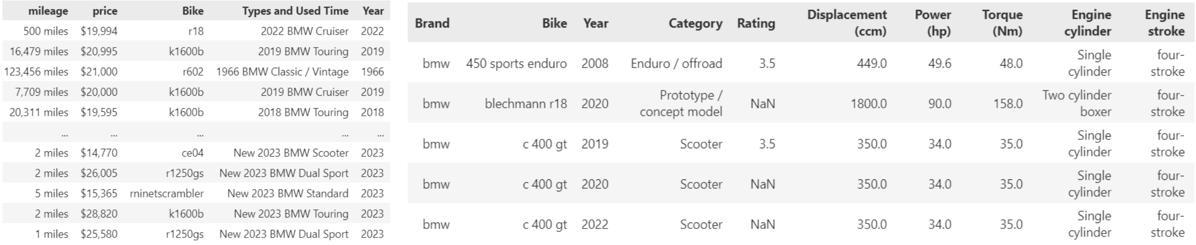
2. Dataset

To train a machine learning model effectively and achieve accurate market price predictions, it is crucial to have access to latest high quality data. For this report, the used data is obtained from the dataset *"USA Comprehensive Motorcycles Dataset 9k+"*[8] and *"Motorcycle Specifications Dataset"*[9], available on **kaggle.com**. Both are licensed under *CC0:Public Domain.*

The former dataset provides market prices, model, mileage and year of manufacture for the most popular brands *BMW*, *KTM*, *Royal Enfield*, *Suzuki*, *Yamaha* and *Ducati* up to 2023, covering a significant portion of the motorcycle market. The second dataset provides further details for most motorcycle models like displacement, power or number of cylinders.

2.1. Preprocessing

An important initial task for machine learning projects is the preprocessing of the dataset. The complexity of this task varies strongly depending on the underlying quality of the data. As for the dataset used in this report, this step took longer than anticipated. As the data for all brands were provided separately, one major task was to both merge all brand datasets into one and following that, merging the resulting dataset with the one containing further details about the model specifications. An excerpt of the used datasets is shown in Figure 12. An enlarged view of the datasets is shown in the Appendix A.



mileage	price	Bike	Types and Used Time	Year
500 miles	$19,994	r18	2022 BMW Cruiser	2022
16,479 miles	$20,995	k1600b	2019 BMW Touring	2019
123,456 miles	$21,000	r602	1966 BMW Classic / Vintage	1966
7,709 miles	$20,000	k1600b	2019 BMW Cruiser	2019
20,311 miles	$19,595	k1600b	2018 BMW Touring	2018
...	...	...	...	...
2 miles	$14,770	ce04	New 2023 BMW Scooter	2023
2 miles	$26,005	r1250gs	New 2023 BMW Dual Sport	2023
5 miles	$15,365	rninetscrambler	New 2023 BMW Standard	2023
2 miles	$28,820	k1600b	New 2023 BMW Touring	2023
1 miles	$25,580	r1250gs	New 2023 BMW Dual Sport	2023

Brand	Bike	Year	Category	Rating	Displacement (ccm)	Power (hp)	Torque (Nm)	Engine cylinder	Engine stroke
bmw	450 sports enduro	2008	Enduro / offroad	3.5	449.0	49.6	48.0	Single cylinder	four-stroke
bmw	blechmann r18	2020	Prototype / concept model	NaN	1800.0	90.0	158.0	Two cylinder boxer	four-stroke
bmw	c 400 gt	2019	Scooter	3.5	350.0	34.0	35.0	Single cylinder	four-stroke
bmw	c 400 gt	2020	Scooter	NaN	350.0	34.0	35.0	Single cylinder	four-stroke
bmw	c 400 gt	2022	Scooter	NaN	350.0	34.0	35.0	Single cylinder	four-stroke

Figure 1: Excerpt of one of the brand datasets (BMW) containing price and selling specifications (left) and the dataset containing further model specifications (right).

Before merging all datasets, it is necessary to clean the columns properly. For this purpose, a descriptive column of the brand datasets was discarded, due to its lack of consistently formatted data. Some entries of the removed column provided details about the bike's condition or the limitation of the model. Additionally, the year of manufacture was extracted from the *Types and Used Time* column and all entries with no information on the price and mileage were dropped. The *Bike* column of the price datasets and the *Bike* column of the specifications dataset were brought to the same formatting and style, such that a merge is possible.

The datasets are combined, based on the bike and the year of manufacture column. In cases with no matching bike model for the exact year of manufacture in the specifications dataset, the entries are merged with another year of manufacture of that model. Some bike models of the specifications dataset are missing some column entries like *Category* for specific years of manufacture. These entries were filled with data from matching bike models from different years of manufacture. At this stage, all entries for which the column *Category* is empty are dropped.

`NaN` entries are filled with the mean value of the corresponding bike category. Additionally, the *mileage* and *price* column data types are converted into integers the values and transformed into kilometres and euros instead of miles and dollars.

All brand subsets are concatenated to form one comprehensive dataset. A new condition column containing boolean values for Used (`True`) and New (`False`) is created based on the mileage. The *Year* column is exchanged with the relative *Age [a]* of the bike. The final columns selected for further analysis are chosen as *Mileage [km], Price [€], Bike, Brand, Category, Displacement [ccm], Power [hp], Torque [Nm], Condition* and *Age [a]*. Data entries with a price larger than 50 000 € are removed, since they are most likely limited editions.

2.2. Exploratory Data Analysis

Before applying machine learning algorithms to the dataset and hoping for good results, it is important to first analyze the data distributions, correlations between attributes and explore useful data scaling methods. As some of the columns contain categorical data (like *Bike*), it is essential to convert them into numerical data for plotting purposes. For the *Bike* and *Category* column, a frequency mapping approach is chosen, meaning that the categorical entries are replaced with their corresponding frequency in the dataset. This is a common approach for categorical columns with a wide variety of entries. For the *Brand* column, a dummy column is created. This means, that for each unique value of the respective column, a new column with boolean values is created.

To gain a general understanding of the data distributions, a scatter matrix containing all columns (except the dummy columns) is presented in Figure 2.

It is evident that there is a significant amount of newer motorcycles with low age and mileage. The distribution of displacement, power and torque is mostly normal, with significant peaks around the mean values, which is due to the filling of `NaN` values with the mean. Several scatter plots show strong correlations, such as *Power [hp]* and *Torque [Nm]* (which is very much expected due to their causal link). The Figure 3 shows clear dependencies between the price and attributes, indicating their usefulness for regression problems.

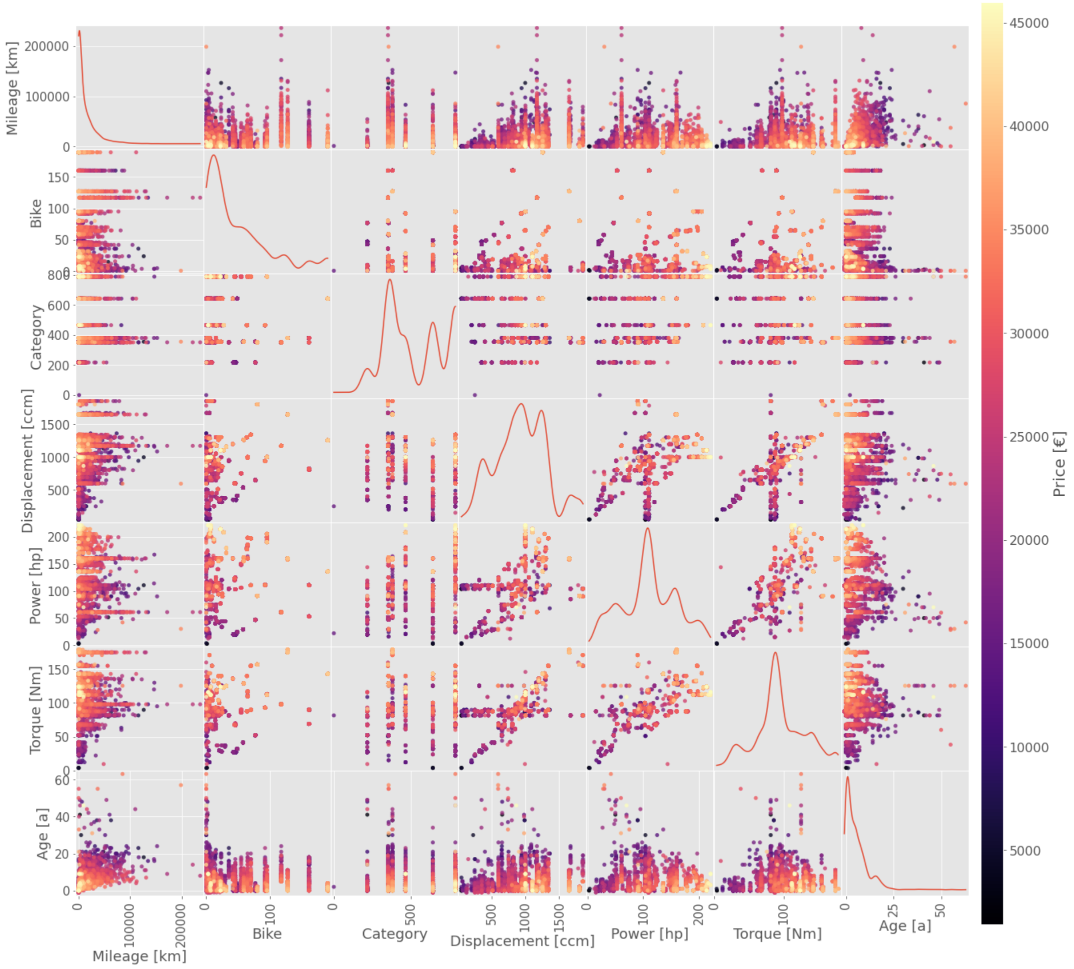


Figure 2: Scatter matrix of all dataset attributes (except dummy columns) with their histograms presented on the diagonal.

When handling large datasets with a high amount of attributes and a very significant amount of data entries, long training times, overtraining and long waiting times for hyperparameter optimization is inevitable. Hence, it is crucial to take the correlations among the used attributes into account. The correlation matrix for all attributes is shown in Figure 4.

The attributes such as torque, power and displacement show high correlation, which is expected given their mechanical nature. Similarly, mileage and age, as well as the brand *Royal Enfield* and the torque are highly correlated. However, at this stage, no variables are discarded, due to the already limited number of attributes and the later performed feature selection.

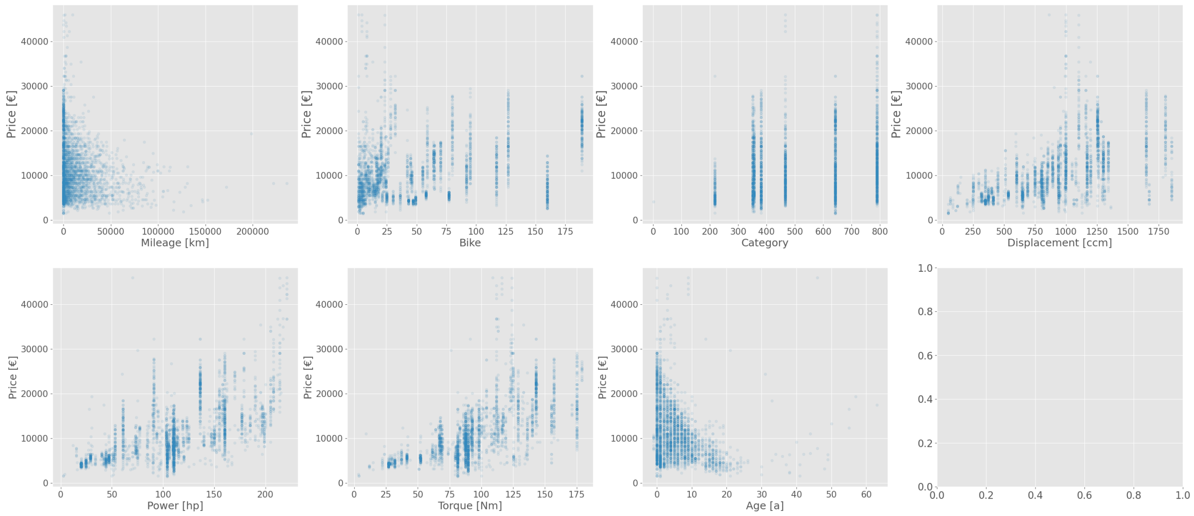
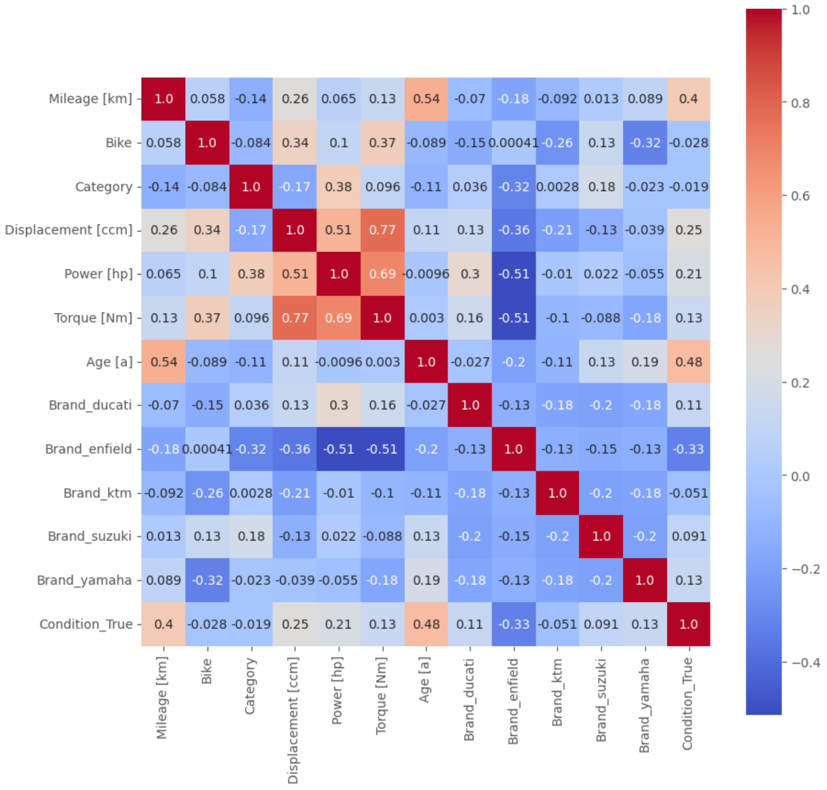


Figure 3: Scatter plots of the training attributes with the target value (*Price [€]*).



	Mileage [km]	Bike	Category	Displacement [ccm]	Power [hp]	Torque [Nm]	Age [a]	Brand_ducati	Brand_enfield	Brand_ktm	Brand_suzuki	Brand_yamaha	Condition_True
Mileage [km]	1.0	0.058	-0.14	0.26	0.065	0.13	0.54	-0.07	-0.18	-0.092	0.013	0.089	0.4
Bike	0.058	1.0	-0.084	0.34	0.1	0.37	-0.089	-0.15	0.00041	-0.26	0.13	-0.32	-0.028
Category	-0.14	-0.084	1.0	-0.17	0.38	0.096	-0.11	0.036	-0.32	0.0028	0.18	-0.023	-0.019
Displacement [ccm]	0.26	0.34	-0.17	1.0	0.51	0.77	0.11	0.13	-0.36	-0.21	-0.13	-0.039	0.25
Power [hp]	0.065	0.1	0.38	0.51	1.0	0.69	-0.0096	0.3	-0.51	-0.01	0.022	-0.055	0.21
Torque [Nm]	0.13	0.37	0.096	0.77	0.69	1.0	0.003	0.16	-0.51	-0.1	-0.088	-0.18	0.13
Age [a]	0.54	-0.089	-0.11	0.11	-0.0096	0.003	1.0	-0.027	-0.2	-0.11	0.13	0.19	0.48
Brand_ducati	-0.07	-0.15	0.036	0.13	0.3	0.16	-0.027	1.0	-0.13	-0.18	-0.2	-0.18	0.11
Brand_enfield	-0.18	0.00041	-0.32	-0.36	-0.51	-0.51	-0.2	-0.13	1.0	-0.13	-0.15	-0.13	-0.33
Brand_ktm	-0.092	-0.26	0.0028	-0.21	-0.01	-0.1	-0.11	-0.18	-0.13	1.0	-0.2	-0.18	-0.051
Brand_suzuki	0.013	0.13	0.18	-0.13	0.022	-0.088	0.13	-0.2	-0.15	-0.2	1.0	-0.2	0.091
Brand_yamaha	0.089	-0.32	-0.023	-0.039	-0.055	-0.18	0.19	-0.18	-0.13	-0.18	-0.2	1.0	0.13
Condition_True	0.4	-0.028	-0.019	0.25	0.21	0.13	0.48	0.11	-0.33	-0.051	0.091	0.13	1.0

Figure 4: Correlation matrix of the used dataset attributes with encoded categorical columns.

Another effective method is to explore the behaviour of the data after applying different scaling transformations. The different scaling methods are observed for the *Mileage* and *Age* attributes, as they display the most continuous distribution and correlation with the price. and nice correlations with the price. The scatter plots for these two attributes showcasing the *"Standard"*, *R"obust*, *"Gaussian"*, *"Min-Max"*, *"Max-Abs"* and *"Uniform"* scaling methods are shown in the Appendix A. The online documentation offers further insides on the transformations applied. In the training of the individual regressor models, the standard, robust and normalised scaling are compared with no scaling.

3. Machine Learning Models

As price prediction is a common regression task, multiple regressors are trained separately. The best-performing models are then ensembled using a **VotingRegressor** to achieve the best possible performance.

3.1. Regression Models

For the single regressor models, the models *XGBoost* (one tree-based and one based on linear models), *CatBoost*, *AdaBoost* and *ExtraTrees* are trained and compared using the three scaling methods discussed in subsection 2.2 and also compared to no scaling.
The target variable *Price* is transformed by a logarithmic function. Models based on a Min-Max scaling of the price yielded on worse results on average. To split the data into training and testing subsets, the train-test split function of the sklearn.modelselection library is used with a ration of 30 % for testing and 70 % for training.
For the training of the individual models, a Pipeline comprising a *preprocessor*, *feature selection* and the respective *regressor* is deployed. The preprocessor includes a numeric and categorical transformer. The numeric preprocessor uses a SimpleImputer to fill missing entries with the mean and scales entries according to the chosen scaling method. The categorical preprocessor uses a SimpleImputer to fill `NaNs` with the most frequent entry value and handles categorical values with a OneHotEncoder, creating dummy columns. For the CatBoost model, no categorical transformation is applied, as one of the strengths of this model is the handling of categorical data.

The feature selection is carried out using an Extra Trees Regressor, to evaluate the most important features, which are then used for the training. For the CatBoost model, no feature selection is performed, due to challenges with transformating the categorical data. Hyperparameters are optimised using a **RandomizedSearchCV** grid search with 10-fold cross validation. The best parameters for each model's scaling can be found in the Appendix A.

To evaluate the model performance a 10-fold Cross-Validation of the Root Mean Squared Error (RMSE) is performed and the Mean Squared Error (MSE), Mean Absolute Error (MAE), R^2 are extracted using the test subset and the Training Time (TT) is recorded. The results for the best-performing models are displayed in Table 4. The full table can be found in the Appendix A. The best models are chosen based on their R^2 score, MAE and the standard deviation of the RMSE. The R^2 score indicated how well the true prices can be explained by the regression, with values ranging from 0 to 1. The MSE on the other hand, measures the average squared difference between predicted and true values, penalizing large residuals. The MAE measures the average absolute error of the predicted data points. In this use case, the R^2 - score and MAE are chosen as the main metrics to compare the different data models. Large residuals usually arise from motorcycles that are limited in quantity or are of particularly bad quality and therefore sell way below or above the market value. Since this information is not included in the training of the model, these entries result in high MSE. Therefore, the MAE appears to be the more effective measure to quantify errors. The most important attributes chosen by feature selection with the ExtraTrees Regressor and the different scaling methods applied are visually presented in the Appendix A.

According to the training with optimised hyperparameters, the **tree-based XGBoost** model with **gaussian** scaling, the **CatBoost** model with **standard** scaling and the **Extra Trees** model with **robust** scaling are performing best and are chosen for the ensembled model.

3.2. Ensembled Model

To create an ensembled model of the best-performing models, a VotingRegressor is used. The weights of the individual models for the VotingRegressor are determined through

Evaluation Individual Regressors

Model	Scaler	RMSE / €	MSE / €	MAE / €	R^2	TT / s
XGBoost (Tree)	Standard	2400.5 ± 429.7	4557951.9	1350.6	0.894	2.37
	Gaussian	2379.3 ± 395.6	4514569.4	1352.1	0.897	2.42
	Robust	2405.0 ± 425.2	4500529.4	1356.1	0.895	2.71
	No Scaling	2389.3 ± 415.9	4479635.3	1342.4	0.897	2.35
CatBoost	Standard	2334.1 ± 386.9	4146873.1	1297.3	0.902	2.15
	Gaussian	3460.8 ± 500.0	4407888.2	1320.6	0.899	1.73
	Robust	2343.9 ± 395.1	4174227.2	1305.9	0.899	2.51
	No Scaling	2310.0 ± 417.6	4148775.5	1300.4	0.901	1.81
Extra Trees	Standard	2494.6 ± 407.3	4481129.5	1349.5	0.889	3.47
	Gaussian	2507.0 ± 391.9	4426545.2	1357.3	0.886	3.47
	Robust	2470.0 ± 408.5	4394657.3	1339.0	0.890	3.76
	No Scaling	2517.8 ± 435.9	4590153.2	1369.7	0.891	3.94

Table 1: RMSE, MSE, MAE, R^2 score and training time for the best performing individual models and applied scaling methods. The best-performing values are highlighted.

hyperparameter optimisation using a RandomizedSearchCV grid search. The optimal weights are 2 : 1 : 1 corresponding to CatBoost:ExtraTrees:XGBoost, respectively. A schematic visual representation of the ensembled model pipeline is shown in Figure 5. The extracted evaluation metrics of the ensembled model are shown in Table 2.

Evaluation Ensembled Model

RMSE / €	MSE / €	MAE / €	R^2 Test	R^2 Train	TT / s
2341.56 ± 392.62	4116364.15	1291.63	0.905	11.21	0.951

Table 2: RMSE, MSE, MAE, R^2 score for training and testing and training time for the ensembled model.

It seems that the model is prone to overfitting, based on the high discrepancy between the R^2 of training and testing. This could be due to too many attributes (especially

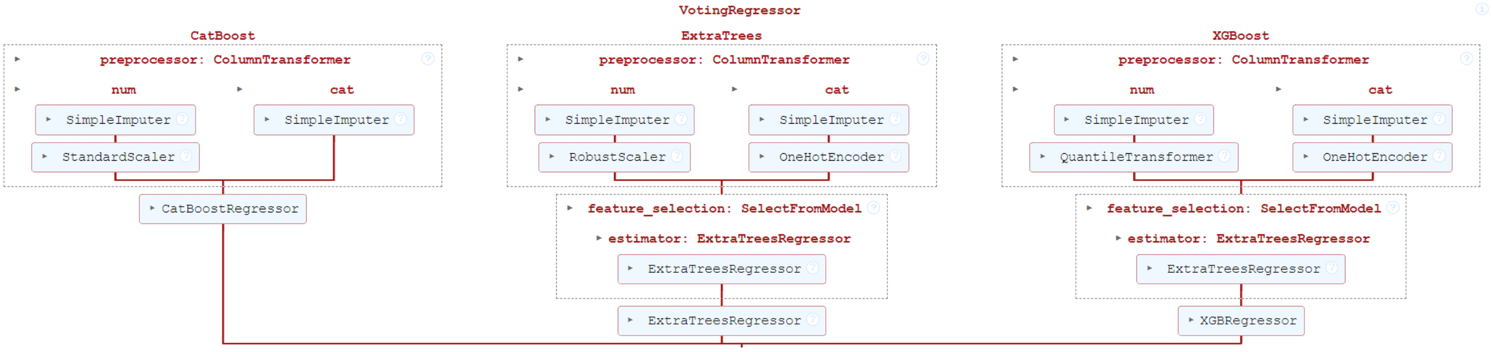


Figure 5: Schematic view of the ensembled model showing the single pipeline steps (preprocessing and feature selection).

after encoding categorical attributes) in comparison to the provided data entries. There is no clear improvement after ensembling multiple different models, as the CatBoost regressor is already performing exceptionally well for this dataset. The learning curve of the training and evaluation dataset shown in Figure 6, illustrates the scaling of the training and validation error, depending on the size of the data sample used for training. The validation error appears to decrease with more data samples, suggesting that a larger dataset might improve the performance of the ensembled model.

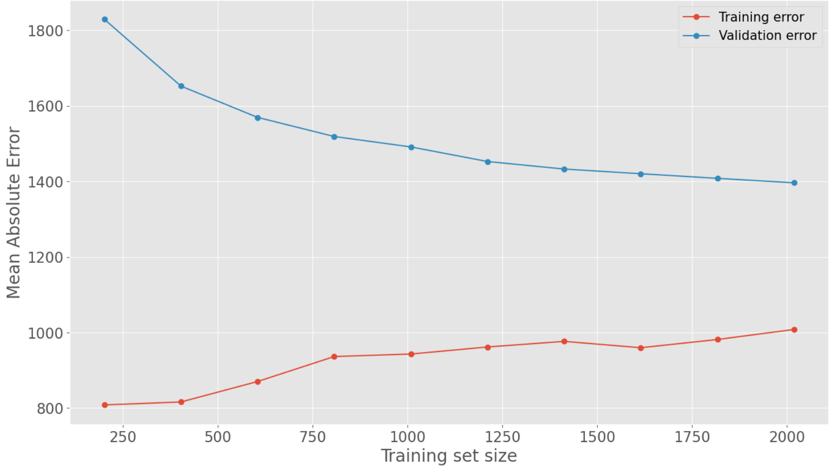


Figure 6: Learning curve of the ensembled model for ten different sizes of the training subset.

The residuals (true minus predicted price) of the predicted and true prices for the ensembled model for the predicted prices are visualised in the scatter plot in Figure 7.

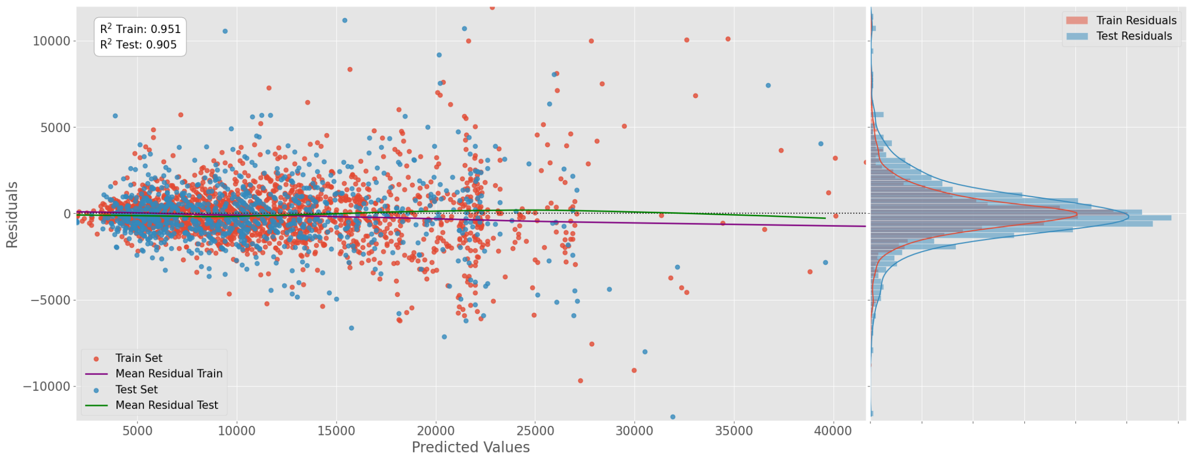
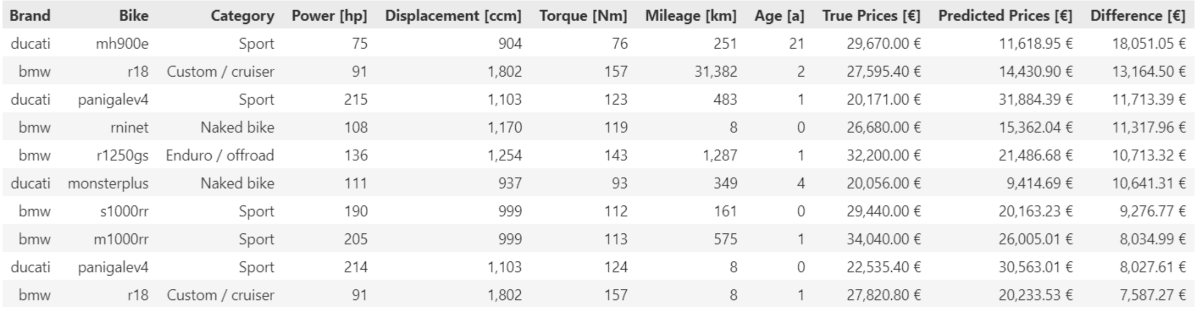


Figure 7: Residuals for different predicted prices for the ensembled model for training and testing subsets. The model seems to overestimate the prices.

Figure 8 shows the data entries with the highest absolute residuals and therefore worst predictions. As suggested, these bikes are limited editions, which sell for very high prices. An example of this is the first entry.



Brand	Bike	Category	Power [hp]	Displacement [ccm]	Torque [Nm]	Mileage [km]	Age [a]	True Prices [€]	Predicted Prices [€]	Difference [€]
ducati	mh900e	Sport	75	904	76	251	21	29,670.00 €	11,618.95 €	18,051.05 €
bmw	r18	Custom / cruiser	91	1,802	157	31,382	2	27,595.40 €	14,430.90 €	13,164.50 €
ducati	panigalev4	Sport	215	1,103	123	483	1	20,171.00 €	31,884.39 €	11,713.39 €
bmw	rninet	Naked bike	108	1,170	119	8	0	26,680.00 €	15,362.04 €	11,317.96 €
bmw	r1250gs	Enduro / offroad	136	1,254	143	1,287	1	32,200.00 €	21,486.68 €	10,713.32 €
ducati	monsterplus	Naked bike	111	937	93	349	4	20,056.00 €	9,414.69 €	10,641.31 €
bmw	s1000rr	Sport	190	999	112	161	0	29,440.00 €	20,163.23 €	9,276.77 €
bmw	m1000rr	Sport	205	999	113	575	1	34,040.00 €	26,005.01 €	8,034.99 €
ducati	panigalev4	Sport	214	1,103	124	8	0	22,535.40 €	30,563.01 €	8,027.61 €
bmw	r18	Custom / cruiser	91	1,802	157	8	1	27,820.80 €	20,233.53 €	7,587.27 €

Figure 8: Data entries with the highest residuals.

Further plots for evaluation of the model are not included in this chapter to due a lack of space, but can be found in the Appendix A.

4. Alternative Method - kNN Regressor

As an alternative approach, a simple kNN regressor of sklearn is utilized, as the true pricing is expected to be highly influenced by similar motorcycles, such as motorcycles within the same brand or similar power. For the kNN classifier, the same Pipeline as for the evaluation of the other models is employed. The regressor is trained for the different scaling methods introduced in subsection 3.1, optimised with a RandomizedSearchCV grid search and evaluated with the same metrics as the other models.

The metrics of the best-performing kNN regressor are listed in Table 3. The kNN regressor appears to perform slightly worse in every computed metric, except the training time. Considering the remarkably low training time, an extensive hyperparameter optimisation or an application on a bigger dataset might be applicable.

Evaluation kNN Regressor					
RMSE / €	MSE / €	MAE / €	R^2 Test	R^2 Train	TT / s
2515.53 ± 415.84	5167510.12	1418.12	0.877	0.995	0.02

Table 3: RMSE, MSE, MAE, R^2 score for training and testing and training time for the kNN regressor.

4.1. Comparison to Ensembled Model

The comparison of the computed quality measures for the ensembled model and the kNN regressor indicates a better overall performance of the ensembled model. However, the achieved performance of the kNN regressor is still better than for some of the other individually trained models, which makes the kNN model a potential candidate to achieve a higher overall performance in the ensembled regressor.

Taking the difference of R^2 score of the training and testing dataset, there are significant indications of potential overtraining. This issue can be addressed by the implementation of regularisation techniques and a more aggressive feature selection process involving the removal of many attributes with low feature importance.

As demonstrated by the residuals in the scatter plot in Figure 9, both regressors seem to overestimate the price in lower ranges and underestimate it for higher values. This observation aligns with the already mentioned concern of many outliers, as special editions or very bad conditions cannot be predicted by the regressor model. The projected histogram of the residuals on the right shows, that the kNN regressor seems to have a higher spread in general and a higher tendency towards underestimating the price.

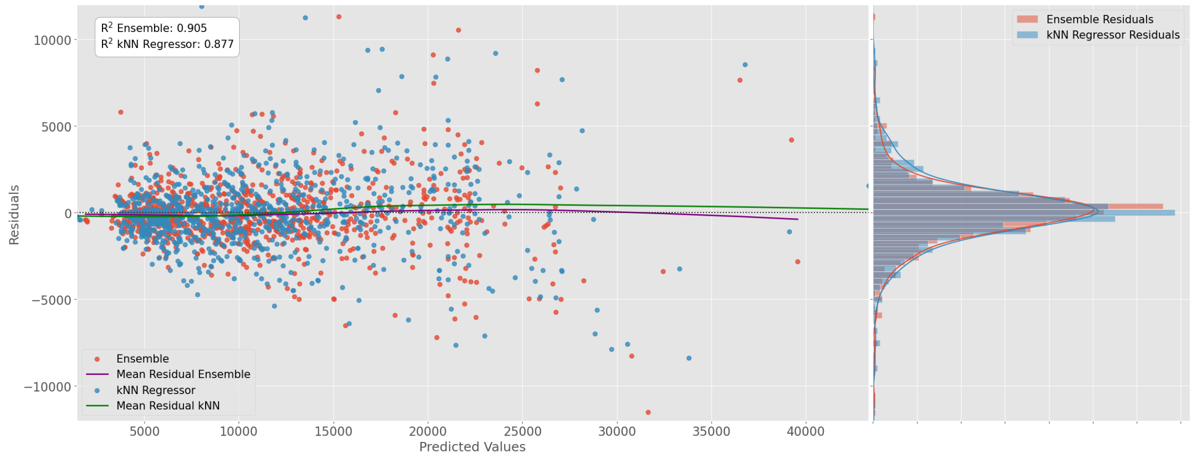


Figure 9: Residuals for different predicted prices for the ensembled model for training and testing subsets.

Although the ensembled model performs better than the alternative approach of a kNN regressor, the CatBoost regressor performs almost equally well. Hence, there is no justification to use an ensembled model over the individual CatBoost regressor. The CatBoost regressor might perform best in this dataset, as many attributes selected as important by the feature selection, seem to have a categorical origin.

Further plots comparing the performance of the ensembled model with the kNN regressor are not included in this chapter due to a lack of space and are included in the Appendix A.

References

[1] F. Pedregosa et al. 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[2] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[3] Charles R. Harris et al. 'Array programming with NumPy'. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[4] John D. Hunter. 'Matplotlib: A 2D Graphics Environment'. Version 1.4.3. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55. URL: http://matplotlib.org/. Current version 3.4.3, DOI: 10.5281/zenodo.5194481.

[5] Michael L. Waskom. 'seaborn: statistical data visualization'. In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.

[6] 'CatBoost'. Version 1.2.5. In: (2024). URL: https://catboost.ai/.

[7] T. Chen and C. Guestrin. 'XGBoost: A Scalable Tree Boosting System'. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785.

[8] *USA Comprehensive Motorcycles Dataset 9k+*. kaggle. URL: https://www.kaggle.com/datasets/joyshil0599/comprehensive-motorcycles-dataset (visited on 30/07/2024).

[9] *Motorcycle Specifications Dataset*. kaggle. URL: https://www.kaggle.com/datasets/emmanuelfwerr/motorcycle-technical-specifications-19702022 (visited on 30/07/2024).

A. Appendix

Evaluation Individual Regressors						
Model	Scaler	RMSE / €	MSE / €	MAE / €	R^2	TT / s
XGBoost (Tree)	Standard	2400.5 ± 429.7	4557951.9	1350.6	0.894	2.37
	Gaussian	2379.3 ± 395.6	4514569.4	1352.1	0.897	2.42
	Robust	2405.0 ± 425.2	4500529.4	1356.1	0.895	2.71
	No Scaling	2389.3 ± 415.9	4479635.3	1342.4	0.897	2.35
XGBoost (Linear)	Standard	3438.8 ± 488.6	8740382.4	1920.3	0.783	2.37
	Gaussian	3349.4 ± 348.1	9572071.3	2082.5	0.765	2.45
	Robust	3460.8 ± 500.0	8402075.3	1915.2	0.787	3.43
	No Scaling	3444.1 ± 498.6	9371909.1	2027.9	0.764	2.54
CatBoost	Standard	2334.1 ± 386.9	4146873.1	1297.3	0.902	2.15
	Gaussian	3460.8 ± 500.0	4407888.2	1320.6	0.899	1.73
	Robust	2343.9 ± 395.1	4174227.2	1305.9	0.899	2.51
	No Scaling	2310.0 ± 417.6	4148775.5	1300.4	0.901	1.81
AdaBoost	Standard	3583.1 ± 440.2	10952648.3	2337.3	0.768	2.62
	Gaussian	3540.2 ± 427.0	11411895.8	2325.4	0.767	2.54
	Robust	3650.5 ± 421.2	11127472.7	2370.4	0.762	3.22
	No Scaling	3626.5 ± 463.8	11975364.1	2421.71	0.751	2.86
Extra Trees	Standard	2494.6 ± 407.3	4481129.5	1349.5	0.889	3.47
	Gaussian	2507.0 ± 391.9	4426545.2	1357.3	0.886	3.47
	Robust	2470.0 ± 408.5	4394657.3	1339.0	0.890	3.76
	No Scaling	2517.8 ± 435.9	4590153.2	1369.7	0.891	3.94

Table 4: RMSE, MSE, MAE, R^2 score and training time for all individual models and applied scaling methods. The best-performing values are highlighted.

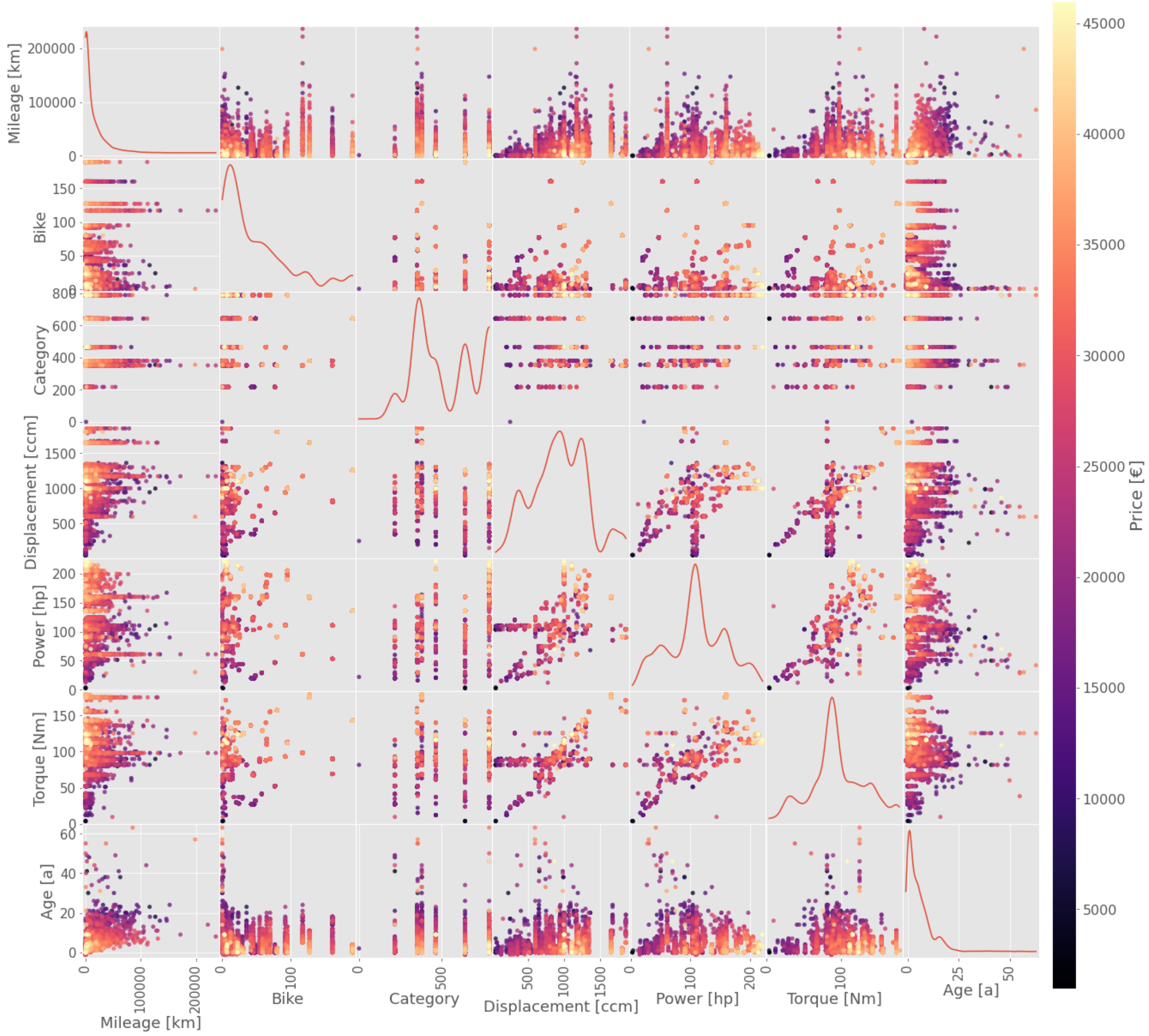
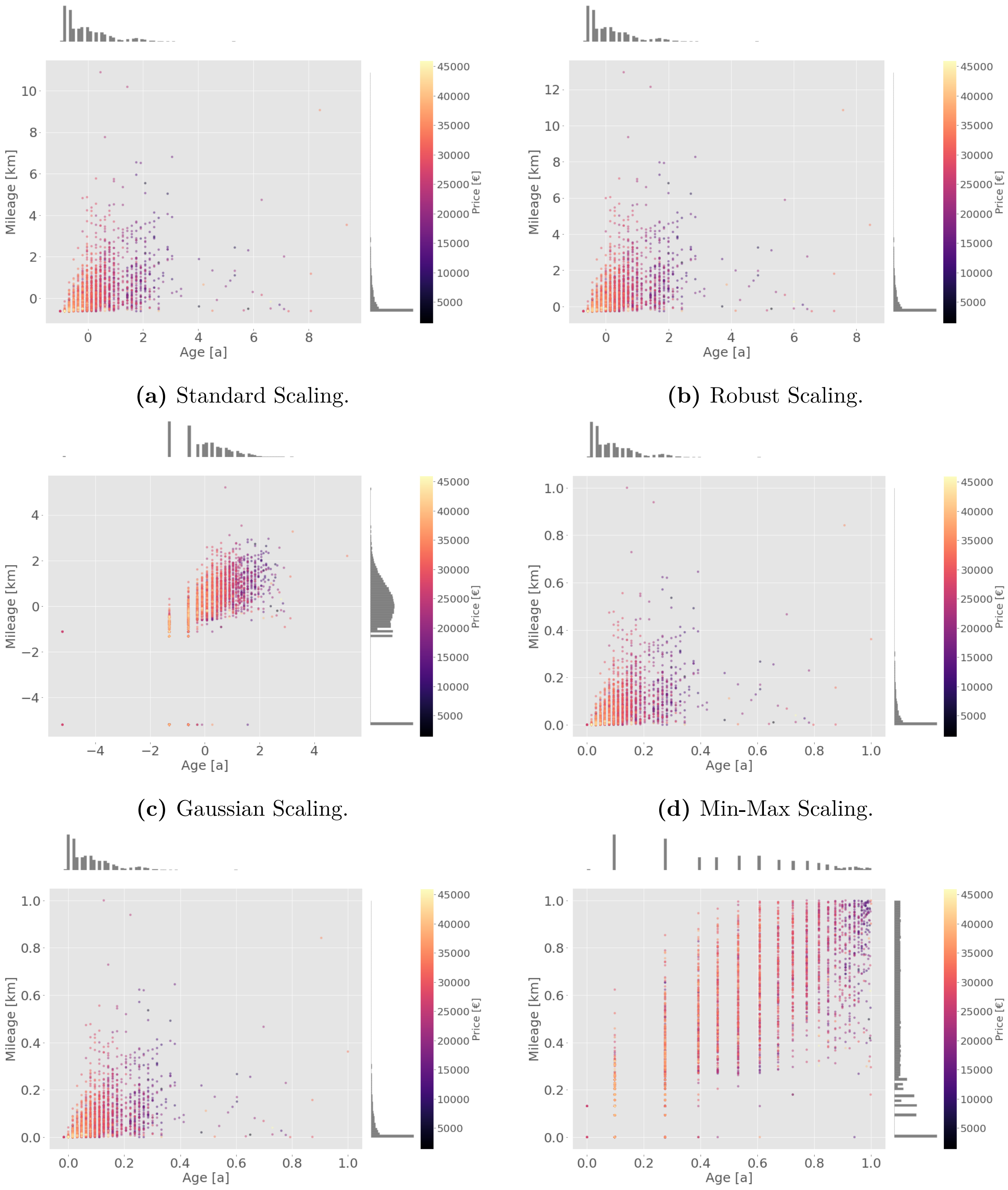


Figure 10: Scatter Matrix of the input variables.



(a) Standard Scaling.

(b) Robust Scaling.

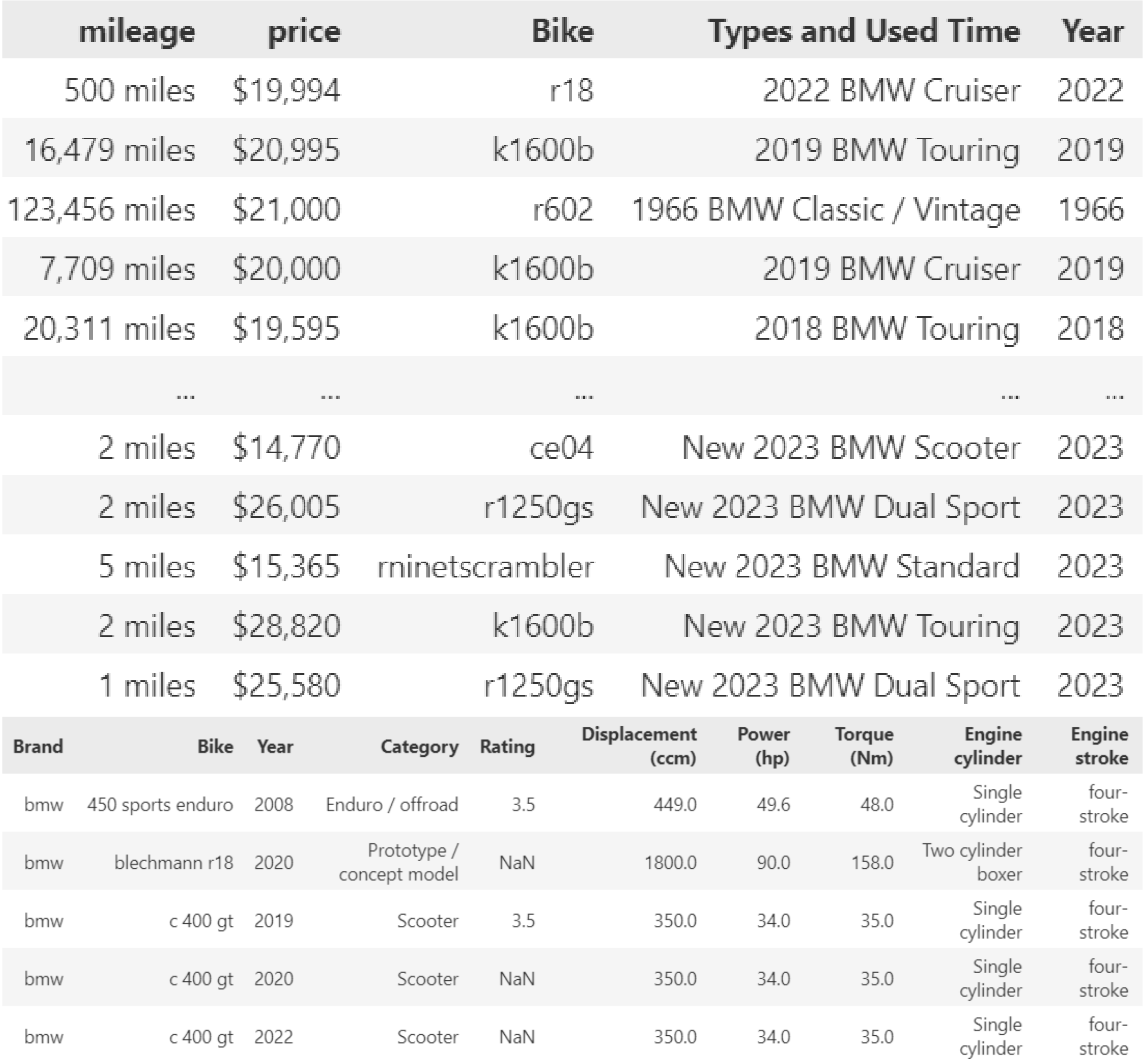
(c) Gaussian Scaling.

(d) Min-Max Scaling.

(e) Max-Abs Scaling.

(f) Uniform Scaling.

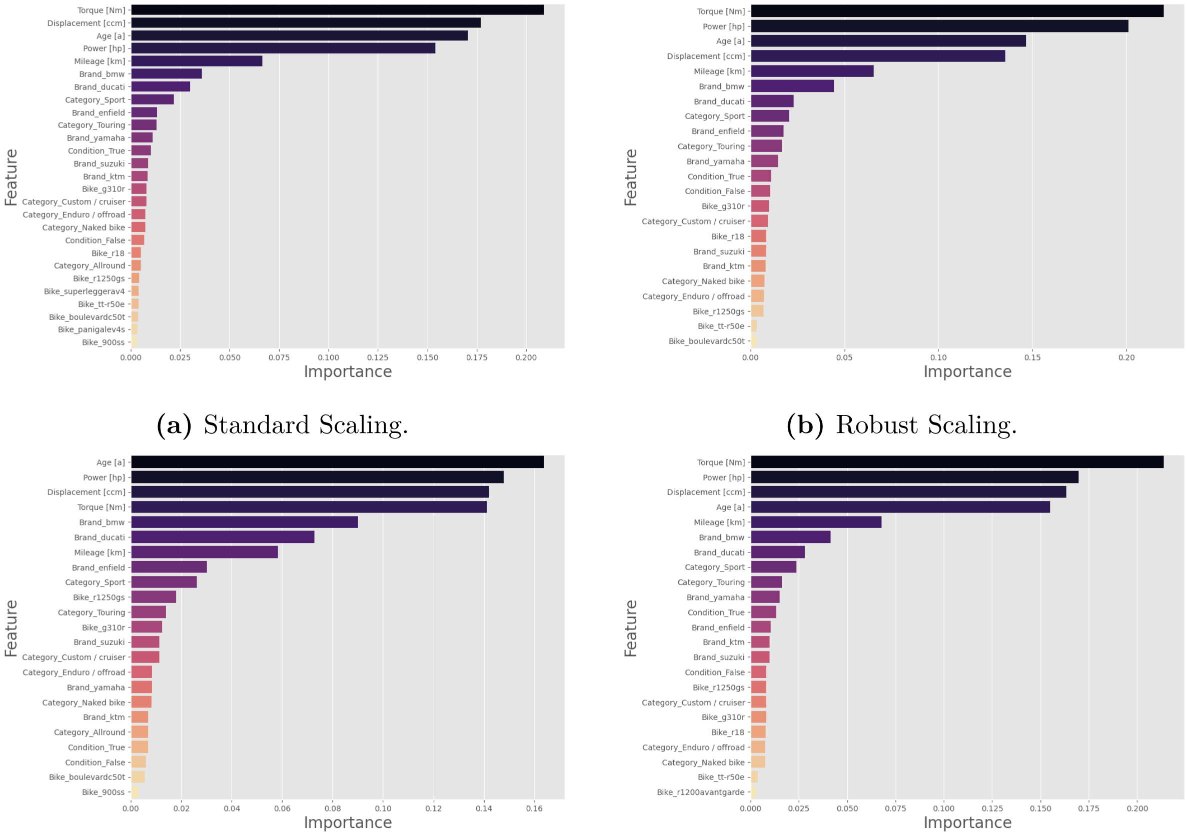
Figure 11: Scatter plot of *Mileage* and *Age* with different scaling methods.



mileage	price	Bike	Types and Used Time	Year
500 miles	$19,994	r18	2022 BMW Cruiser	2022
16,479 miles	$20,995	k1600b	2019 BMW Touring	2019
123,456 miles	$21,000	r602	1966 BMW Classic / Vintage	1966
7,709 miles	$20,000	k1600b	2019 BMW Cruiser	2019
20,311 miles	$19,595	k1600b	2018 BMW Touring	2018
...	...	...	...	...
2 miles	$14,770	ce04	New 2023 BMW Scooter	2023
2 miles	$26,005	r1250gs	New 2023 BMW Dual Sport	2023
5 miles	$15,365	rninetscrambler	New 2023 BMW Standard	2023
2 miles	$28,820	k1600b	New 2023 BMW Touring	2023
1 miles	$25,580	r1250gs	New 2023 BMW Dual Sport	2023

Brand	Bike	Year	Category	Rating	Displacement (ccm)	Power (hp)	Torque (Nm)	Engine cylinder	Engine stroke
bmw	450 sports enduro	2008	Enduro / offroad	3.5	449.0	49.6	48.0	Single cylinder	four-stroke
bmw	blechmann r18	2020	Prototype / concept model	NaN	1800.0	90.0	158.0	Two cylinder boxer	four-stroke
bmw	c 400 gt	2019	Scooter	3.5	350.0	34.0	35.0	Single cylinder	four-stroke
bmw	c 400 gt	2020	Scooter	NaN	350.0	34.0	35.0	Single cylinder	four-stroke
bmw	c 400 gt	2022	Scooter	NaN	350.0	34.0	35.0	Single cylinder	four-stroke

Figure 12: Excerpt of one of the datasets containing the price and selling specifications (upper) and the dataset containing the motorcycle specifications (lower).



(a) Standard Scaling.

(b) Robust Scaling.

(c) Gaussian Scaling.

(d) No Scaling.

Figure 13: .

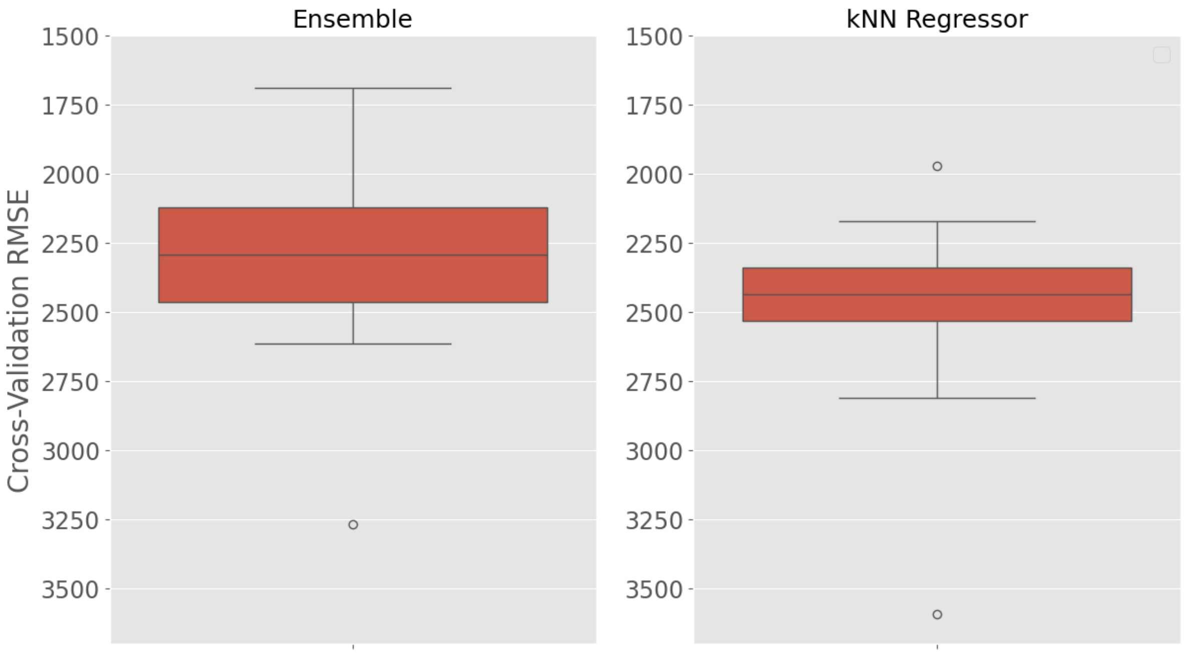


Figure 14: Box plots of the cross-validation results of the RMSE for the kNN regressor and the ensembled model.

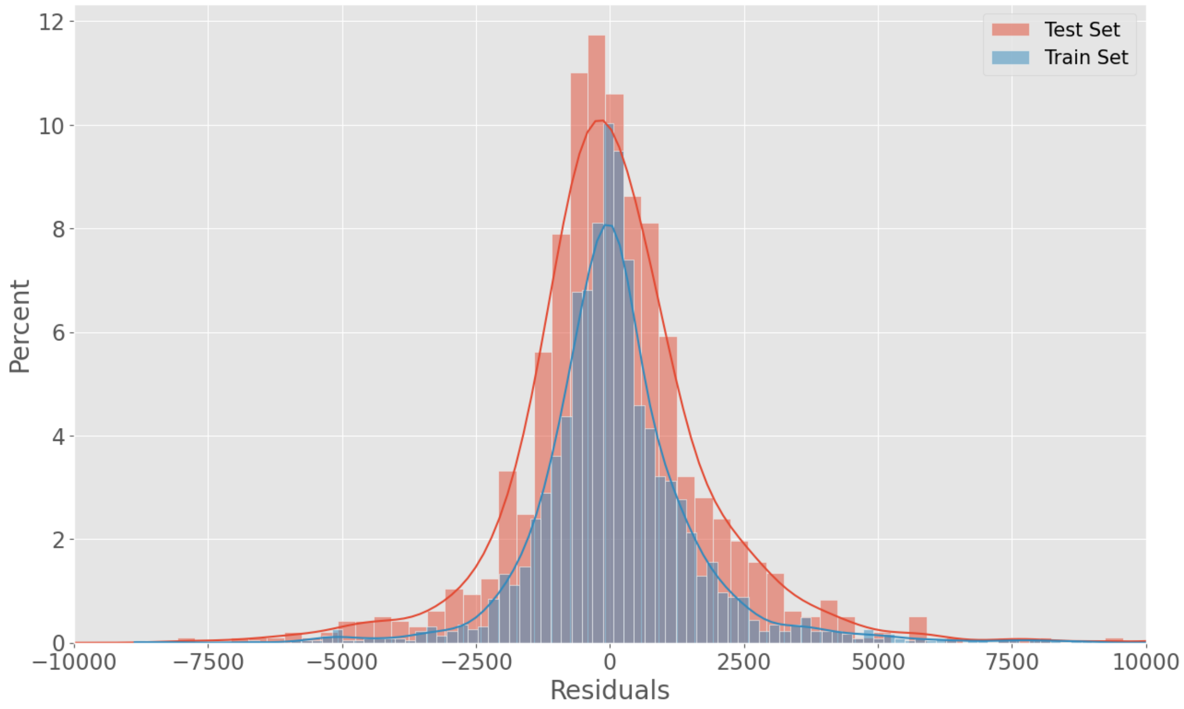


Figure 15: Distribution of the residuals for the ensembled model for the training and testing subset.

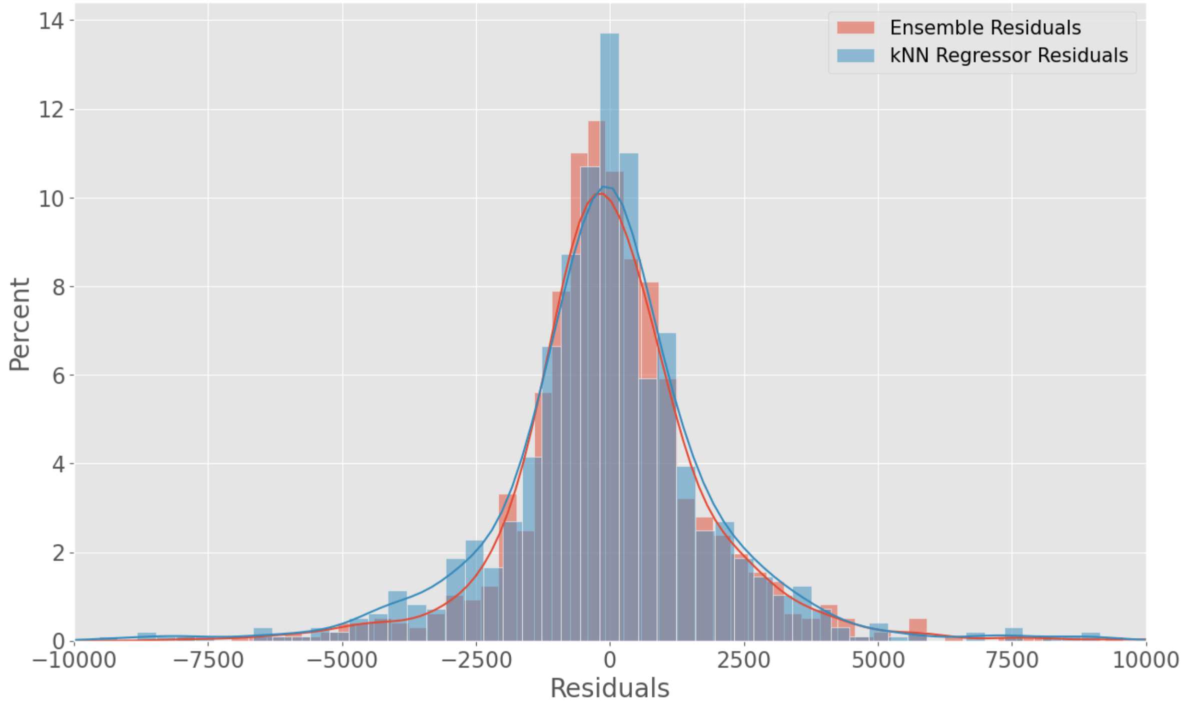
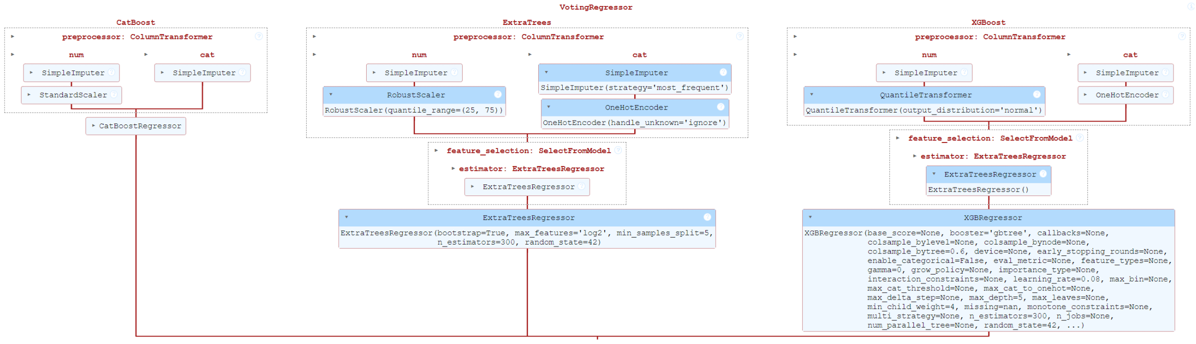


Figure 16: Distribution of the residuals for the testing subset for the ensembled model and kNN regressor.



```
SimpleImputer(strategy='most_frequent')
RobustScaler(quantile_range=(25, 75))
OneHotEncoder(handle_unknown='ignore')
ExtraTreesRegressor(bootstrap=True, max_features='log2', min_samples_split=5,
                    n_estimators=300, random_state=42)
QuantileTransformer(output_distribution='normal')
ExtraTreesRegressor()
XGBRegressor(base_score=None, booster='gbtree', callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.6, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=0, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.08, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=5, max_leaves=None,
             min_child_weight=4, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=300, n_jobs=None,
             num_parallel_tree=None, random_state=42, ...)
```

Figure 17: More detailed view of the model schematic of the ensembled model.

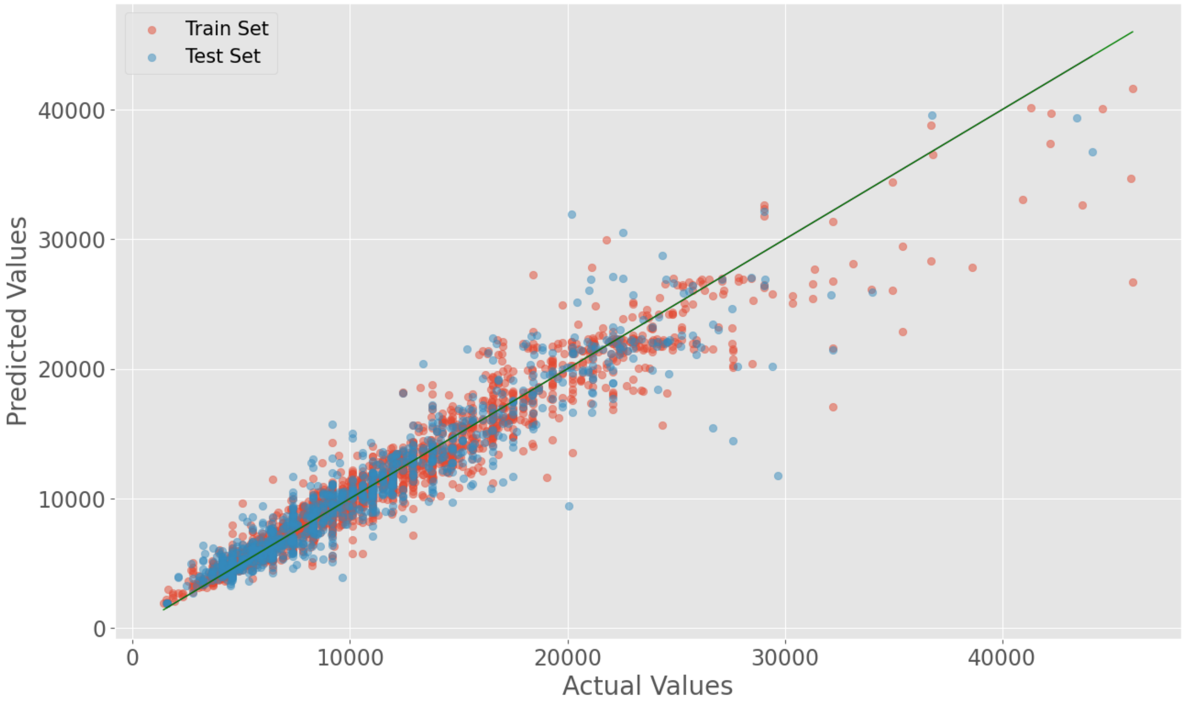


Figure 18: Scatterplot of the true and predicted values for the training and testing subset for the ensembled model.

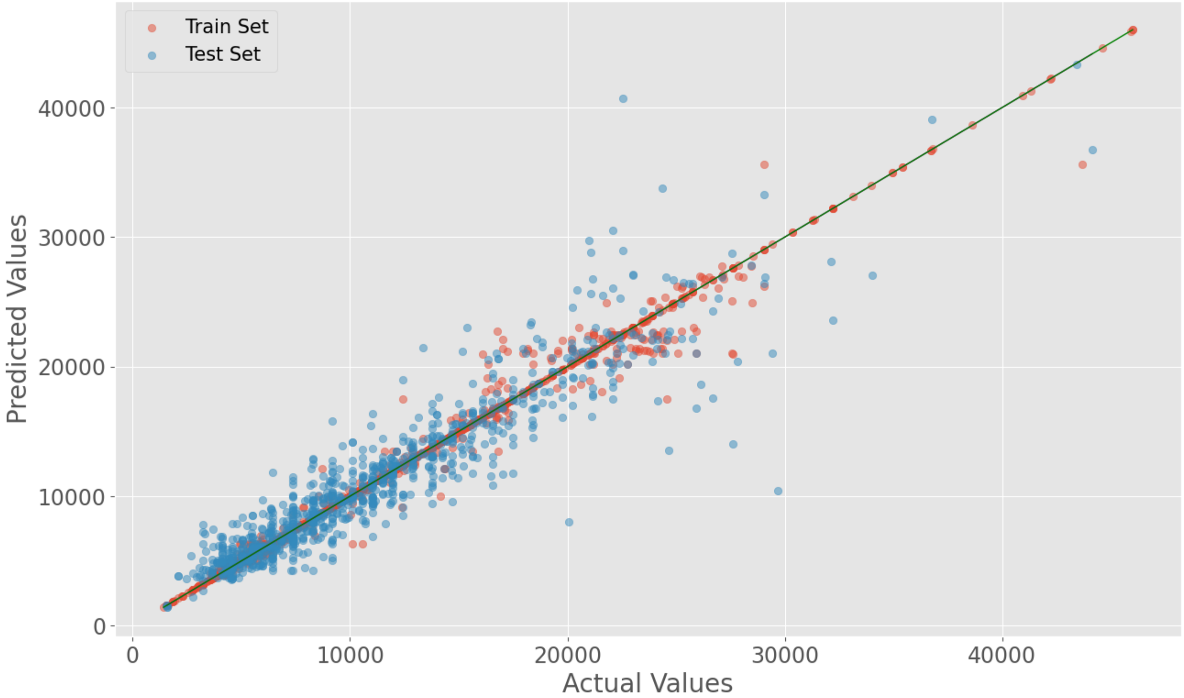


Figure 19: Scatterplot of the true and predicted values for the training and testing subset for the kNN regressor.

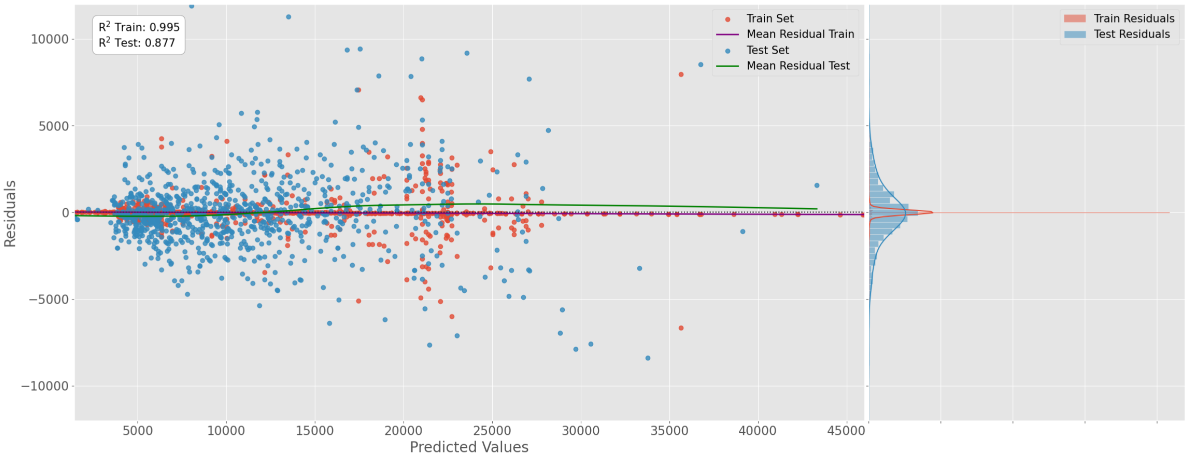


Figure 20: Scatter plot of the residuals for the training and testing subset of the kNN regressor.

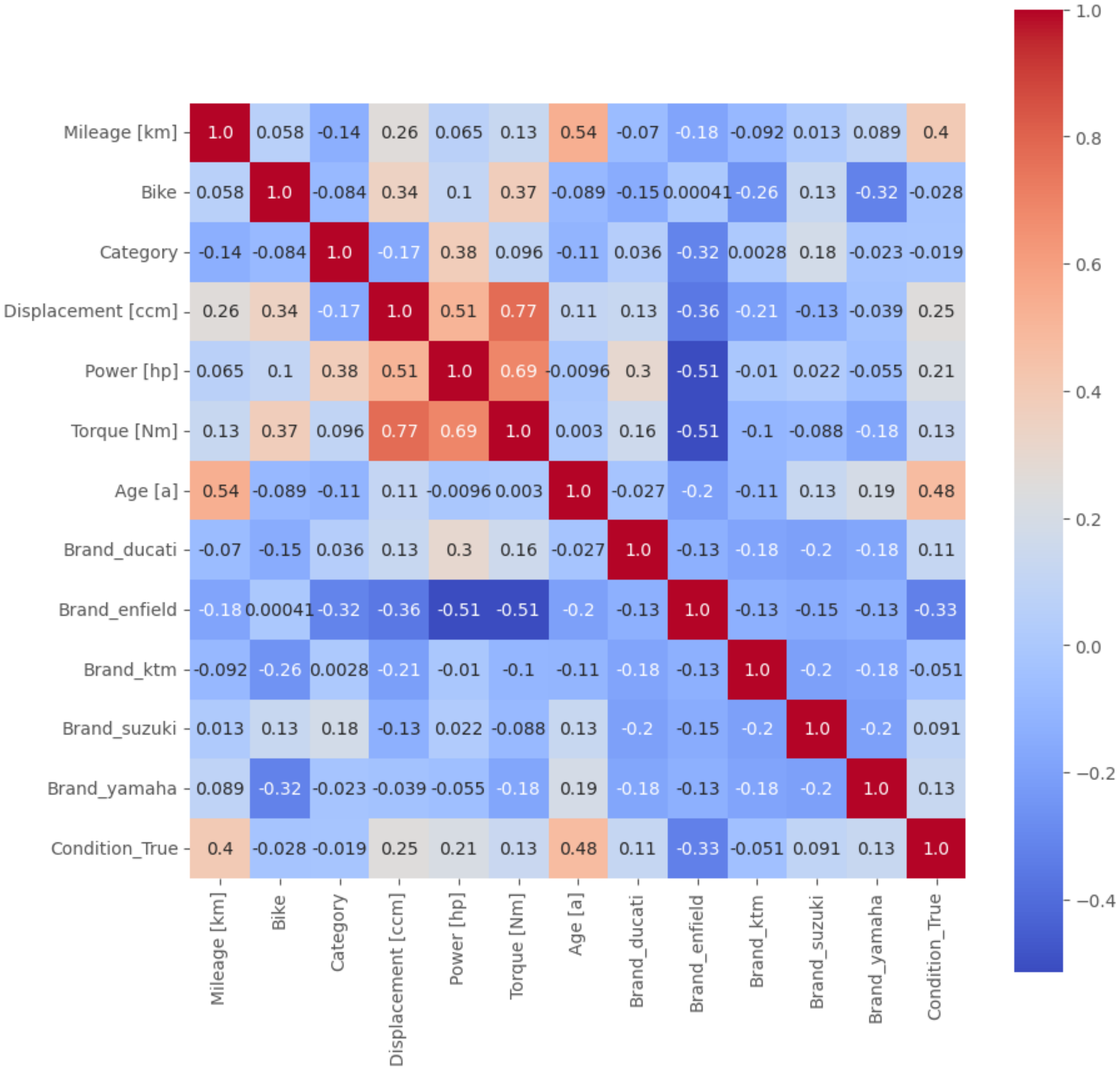


Figure 21: Correlation matrix of the used dataset attributes with encoded categorical columns.