Vehicle price estimator based on the Romanian market

Anghelina Ion-Marian (407)
ion.anghelina@s.unibuc.ro

Buță Gabriel-Sebastian (407)
gabriel.buta@s.unibuc.ro

Dănilă Mihaela-Alexandra (406)
mihaela.danila1@s.unibuc.ro

Enache Alexandru (407)
alexandru.enache1@s.unibuc.ro

Popescu Ioana-Livia (407)
ioana.popescu1@s.unibuc.ro

January 2024

University of Bucharest
Faculty of Mathematics and Informatics

Abstract

The purpose of our project is to create a system for predicting the correct price of a vehicle in the Romanian market. To achieve this, we trained various artificial intelligence models using an original dataset, generated by crawling data from the Autovit platform. The models' architecture aligns with the inherent characteristics of the task, specifically designed for regression.

1 Dataset

The data was obtained from the Autovit platform — the largest online platform in the Romanian space for purchasing cars, where they can be sold by both individuals and legal entities.

At the time of extraction, approximately 36,900 articles were listed as active on the site, but only the "complete" articles, which had all the necessary data for price prediction, were retained.

Therefore, the dataset contains just over 9,000 examples that can be used for training. The vehicles in the dataset are not selected based on a specific criterion; they are all those that have all the necessary data and come from all possible classes, from vehicles sold for dismantling with prices in the hundreds of euros to luxury cars with prices in the hundreds of thousands of euros.

In addition, the dataset has been made available to the community on the Kaggle platform: EDA Project Dataset - Vehicle Price Estimator.

2 Features

For each vehicle in the dataset, the following features have been created:

- Price (Pret): a numerical value representing the amount of money (in euros) requested by the seller for the purchase of the vehicle. This is the value we will predict using the other information.

- Brand (Marca): the brand of the car for sale. This is a crucial indicator because there are brands that manufacture only luxury vehicles, suggesting a higher price, or there are brands that produce budget cars, indicating a lower predicted price.

- Model (Model): the model of the car for sale. Similarly important, as the prices of a specific model compared to the prices of other models are easily comparable.

- Version (Versiune): the engine version of the car for sale. While the version of the car marginally affects the price, it is an indicator that assists in prediction.

- Age (Vechime): the age of the vehicle for sale. An extremely important indicator, as the age is inversely proportional to the value of the vehicle.

- Power (Putere): the power of the vehicle for sale, represented in horsepower. Also an important indicator, as vehicles with very high power indicate luxury cars whose prices are much higher than those of other cars.

- Cubic capacity (Capacitate cilindrica): the engine's cubic capacity. An important indicator that affects the price in various ways (e.g., cars with very low cylinder capacity are cheaper as they are less powerful, while cars with excessively high cylinder capacity also have a lower price due to higher taxes, except for the luxury car segment where buyers are not affected by tax differences).

- Transmission (Transmisie): the type of transmission the car has (manual, automatic). An important factor for the car's price, with manual transmission cars typically priced lower than the same car configurations with automatic transmission.

- Gearbox (Cutie de viteze): the name of the gearbox used by the car. Different from the transmission indicator, as this specifies the type of gearbox, while the transmission indicator indicates the transmission type.

- Fuel Type (Combustibil): the type of fuel used by the car offered for sale. A fairly important indicator, with diesel cars generally having a higher price than equivalent gasoline-powered models.

- Emission Standard (Norma de poluare): the vehicle's emission standard. This indicator is closely related to the vehicle's age (e.g., cars newer than 2016 must have Euro 6 emission standards), providing additional information beyond age (e.g., a car manufactured in 2009 may have Euro 4 or Euro 5 emission standards, not directly determined by age).

- Body type (Tip caroserie): the type of body of the vehicle offered for sale (SUV, Sedan, Hatchback, etc.). An important indicator, as vehicles from certain categories can have significantly different prices compared to vehicles from other categories.

- Condition (Stare): the condition of the vehicle offered for sale (second-hand or new). An indicator with a significant impact on the price, with new cars being significantly more expensive than used ones.

- Country of origin (Tara de origine): the country from which the car was imported. If this indicator's value is Romania, then the car is not imported.

- Color Options (Optiune de culoare): an indicator representing the type of paint the car has. Most cars have metallic colors, but there are also cars with other types of colors, significantly impacting the car's price.

3 Data distribution

The subsets of entries for training and validation were randomly selected in proportions of 80% and 20%, respectively, from the entire dataset.

Subset	Count
Train	7229
Validation	1809

Table 1: The size of the train and validation sets.

4 Exploratory data analysis

4.1 Numerical features

4.1.1 Summary of statistics

Feature name	**Min**	**Max**	**Mean**	**Std**	**Median**
Price	800.00	299999.00	26793.45	27041.73	17700.00
Age	1.00	24.00	6.88	3.92	6.00
Km	1.00	510000.00	133111.75	80668.16	136500.00
Power	54.00	650.00	181.78	88.67	156.00
Cubic Capacity	799.00	6417.00	1989.70	640.96	1968.00

Table 2: Summary of statistics for each numerical column

4.1.2 Correlation analysis

After studying the correlation between *Pret (Price)* and other attributes with numerical values observed in our dataset, the following conclusions were drawn regarding the correlation.

As expected, there is a strong negative correlation (and approximately proportional) between the price of a car and its age, as well as its mileage.

Additionally, a car with higher power or featuring an engine with a higher cylinder capacity will also have a higher price.

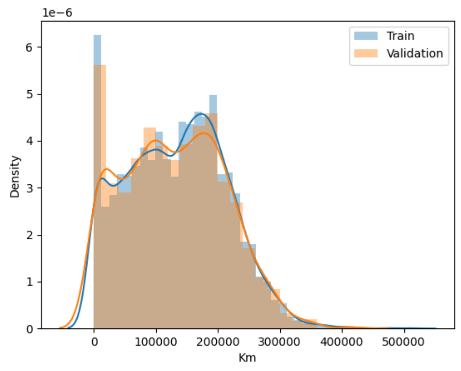
Feature name	**Age**	**Km**	**Power**	**Cubic Capacity**
Price	-0.523257	-0.532601	0.778838	0.633708

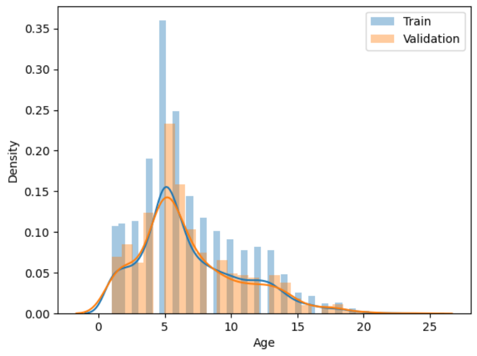
Table 3: Correlation analysis based on Price

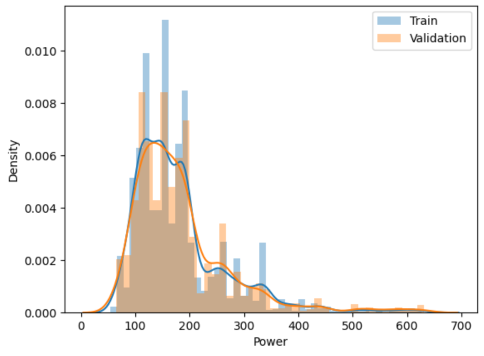
	Km	Age	Cubic Capacity	Power
Km	1.000000	0.724723	-0.060221	0.633708
Age	0.724723	1.000000	-0.064826	-0.236068
Cubic Capacity	-0.060221	-0.064826	1.000000	0.857884
Power	0.633708	-0.236068	0.857884	1.000000

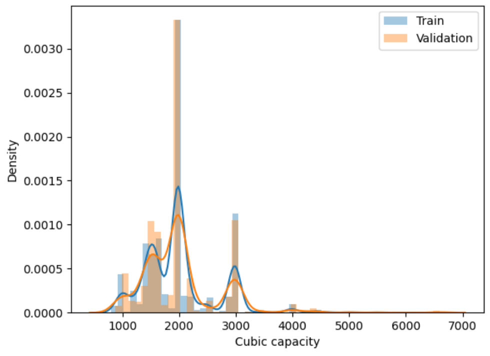
Table 4: The correlation matrix over the numerical attributes

4.1.3 Distributions









4.2 Categorical features

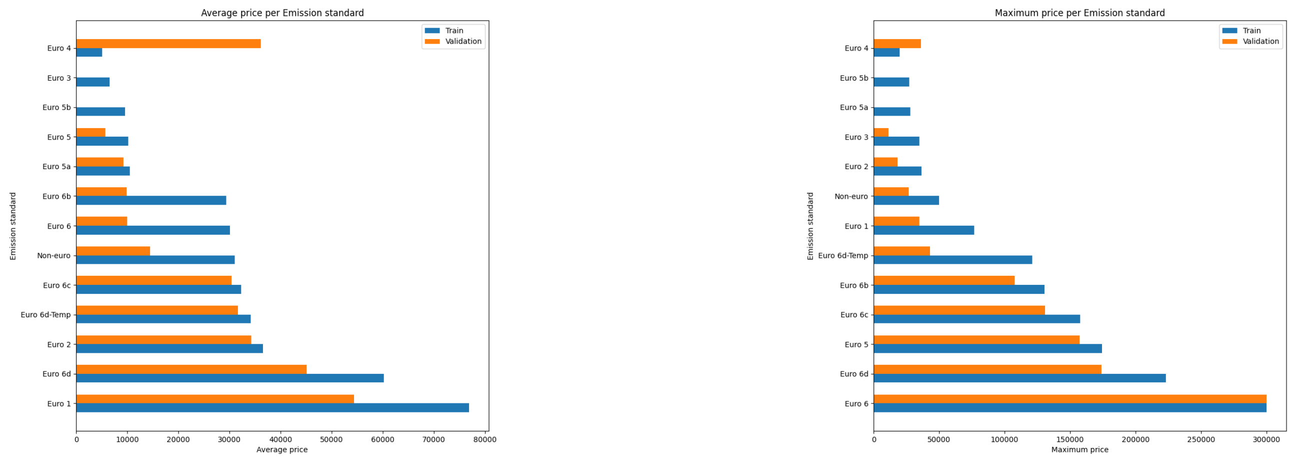


Figure 1: Average price and maximum price categorized by emission standard

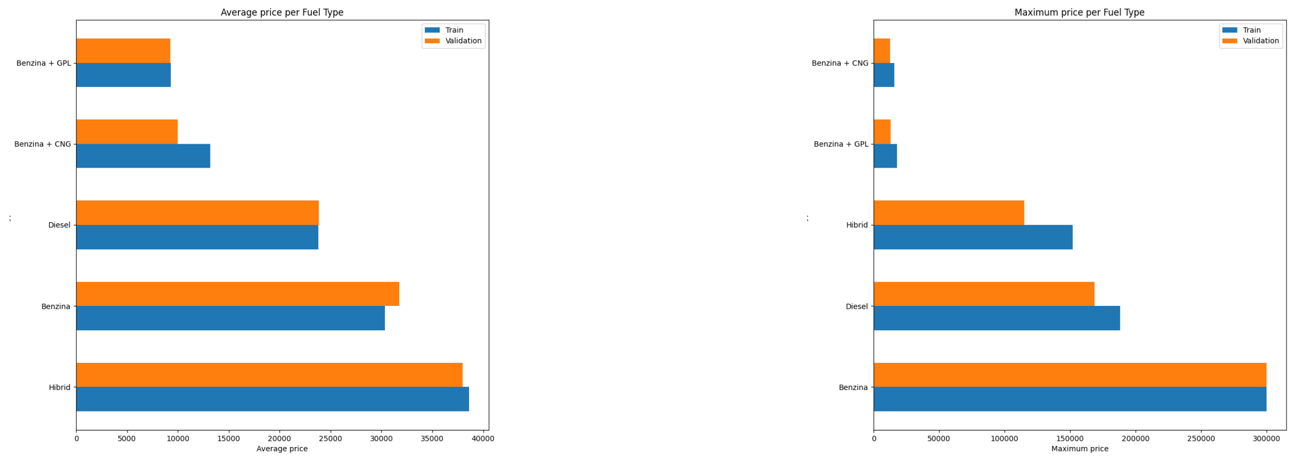


Figure 2: Average price and maximum price categorized by fuel type

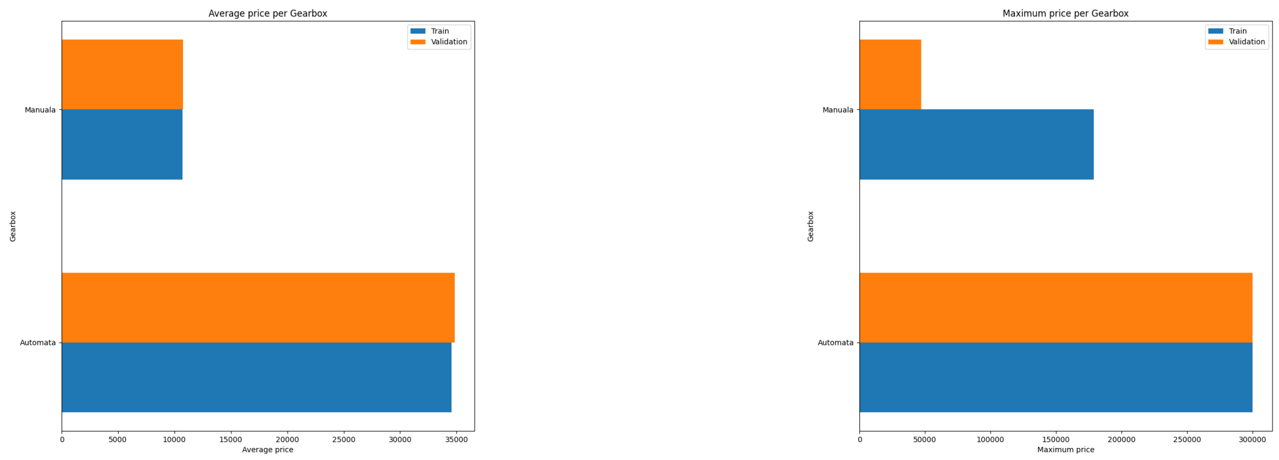


Figure 3: Average price and maximum price categorized by gearbox

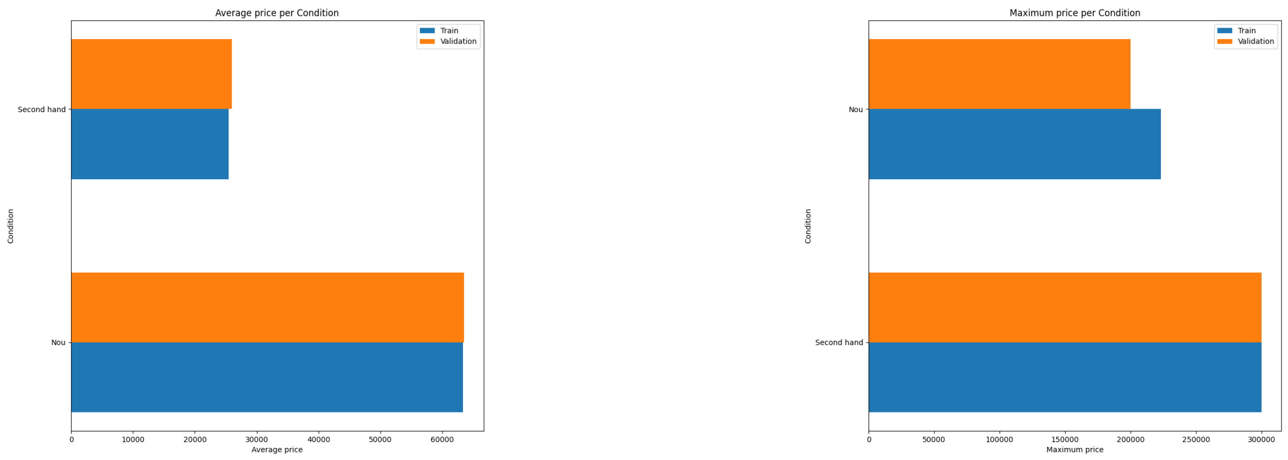


Figure 4: Average price and maximum price categorized by vehicle condition

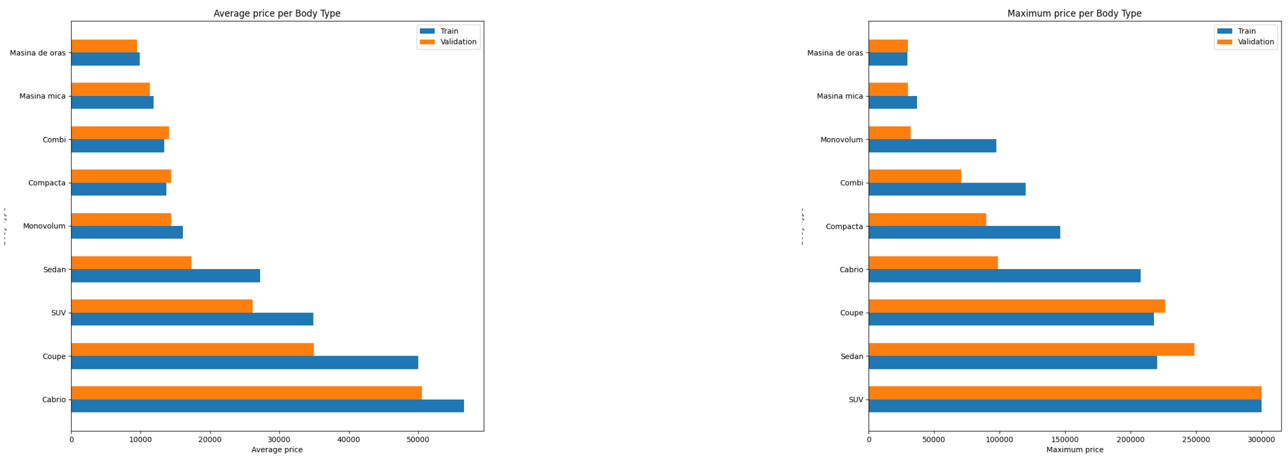


Figure 5: Average price and maximum price categorized by body type

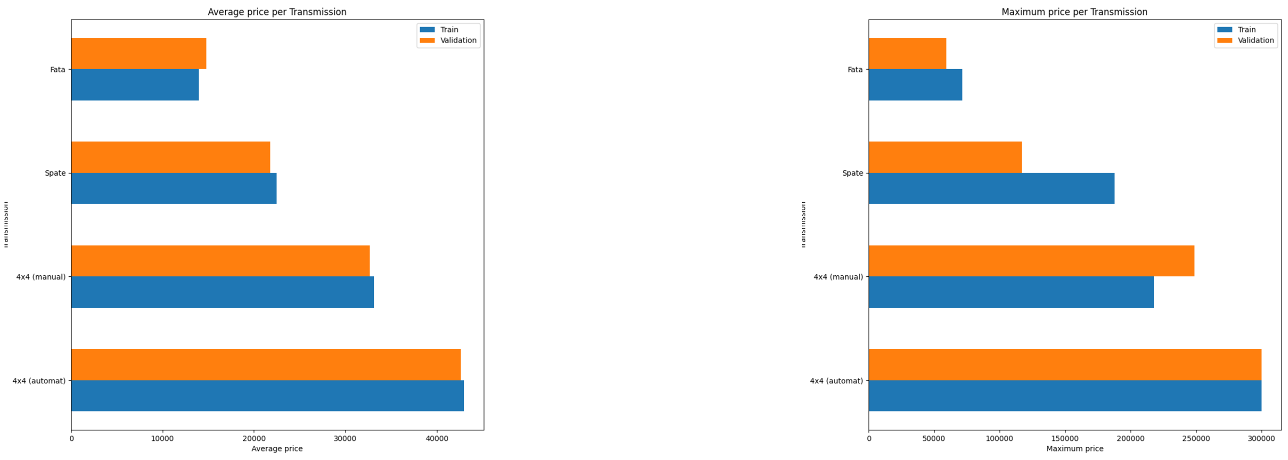


Figure 6: Average price and maximum price categorized by transmission type

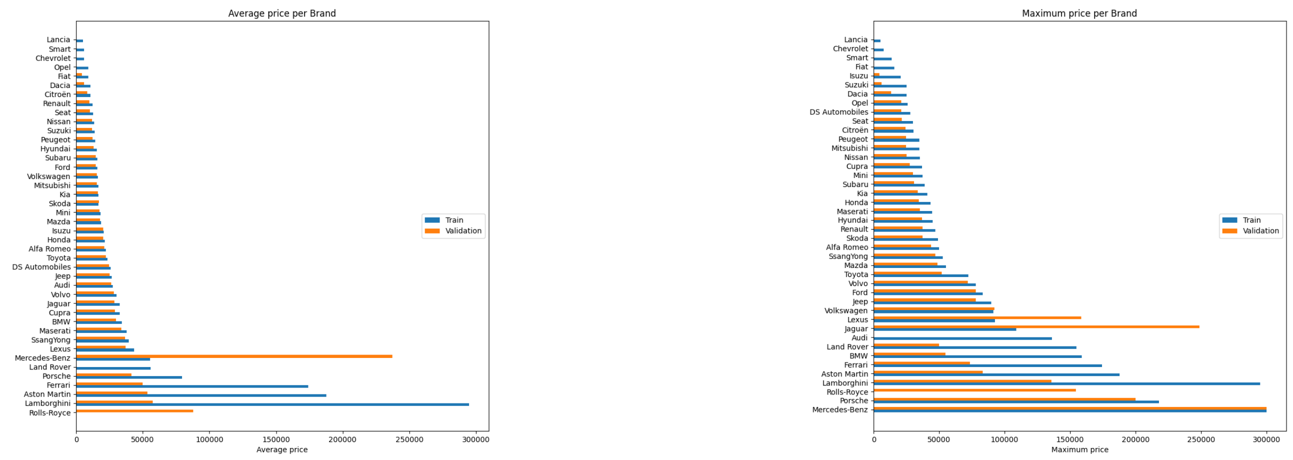


Figure 7: Average price and maximum price categorized by brand

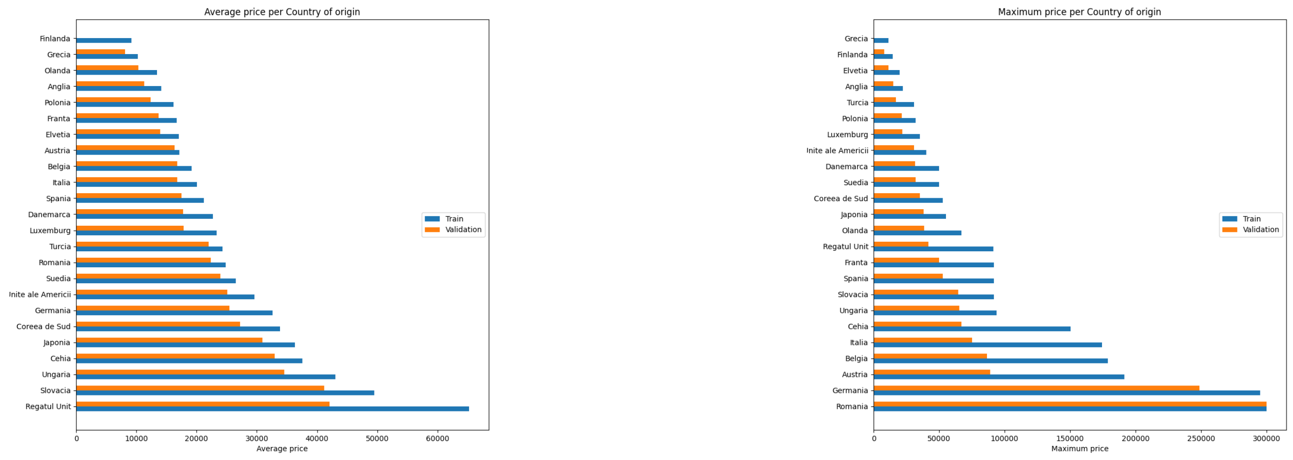


Figure 8: Average price and maximum price categorized by country of origin

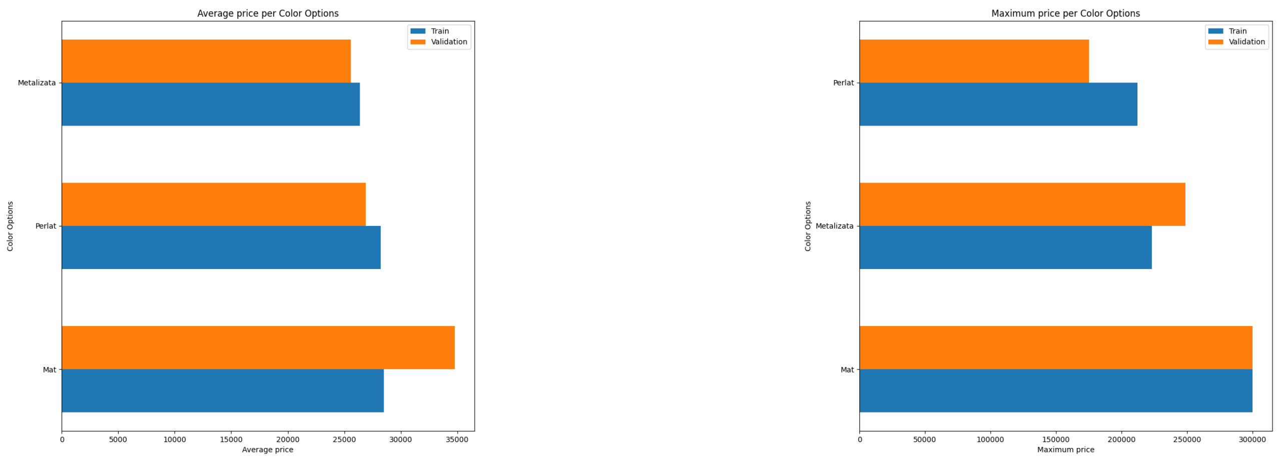


Figure 9: Average price and maximum price categorized by color options

5 Data preprocessing

Categorical features needed to be transformed into a numeric form to be used in the training process. For this purpose, we replaced each categorical feature with its One Hot Encoded representation. Consequently, we obtained a set of features with binary values.

On the other hand, for *numerical features*, we trained a StandardScaler. This scaler transforms the data in each numerical column, so that their values follow a standard distribution with a mean of 0 and a standard deviation of 1.

Through these transformations, we gain various advantages in the training process:

- Better convergence within Gradient Descent-based models..
- Prevention of major changes in the cost function caused by attributes with larger value ranges.
- Improved observation of the correlation between each attribute and the outcome, based on its magnitude order.

6 Proposed approaches

6.1 Metrics

We have chosen Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) as the primary evaluation metrics. Our project prioritizes achieving a lower average absolute error, considering the specific characteristics of our use case which focuses on enhancing the overall transparency in pricing in the Romanian market. Additionally, given the nature of the automotive market, where outliers such as luxury or rare vehicles can exist, MAE's robustness to outliers aligns well with our objective - this ensures that the model's performance is not overly influenced by extreme values.

6.2 Experiments

6.2.1 SVR / LinearSVR

The models in the Support Vector Machines (SVM) class are models based on finding the optimal hyperplane that best represents the training data. Prior to this, the data is projected into a higher-dimensional space using a kernel function. While SVR allows different values for the kernel function, LinearSVR only accepts a linear kernel function.

Model	Hyperparams	Feature Extraction	MAE	RMSE
SVR	$C = 300000, kernel =' rbf'$	All features	1579.24	9336.84
SVR	$C = 50000000, kernel =' rbf'$	All features	**1504.84**	9681.97
SVR	$C = 300000, kernel =' poly'$	All features	1590.83	**8434.16**
LinearSVR	$C = 500000, tol = 0.001$	All features	3068.70	10063.96

Table 5: The results obtained by the SVR / LinearSVR model on our dataset.

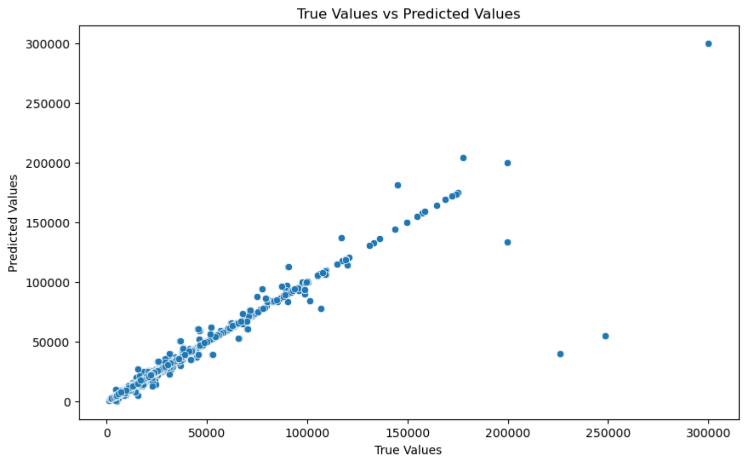


Figure 10: The true vs predicted values obtained by the best SVR / LinearSVR model.

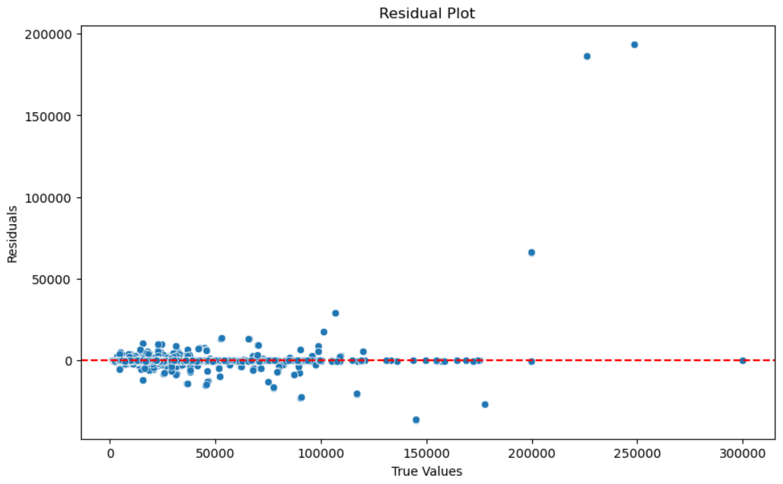


Figure 11: The residual values obtained by the best SVR / LinearSVR model.

6.2.2 KNN

KNN (K-Nearest Neighbors) models calculate predictions by finding the closest examples from the training dataset. Once a distance function is defined between two examples, the nearest k examples are identified, and in the case of regression, the result is the average of their prices.

Model	Hyperparams	Feature Extraction	MAE	RMSE
KNN	$n\_neighbors = 1$	All features	1771.64	9826.84
KNN	$n\_neighbors = 2$	All features	3463.55	10599.42
KNN	$n\_neighbors = 1$	Km, Age, Power, Brand, Model, Version, Fuel	**1492.11**	**8467.35**

Table 6: The results obtained by the KNN model on our dataset.

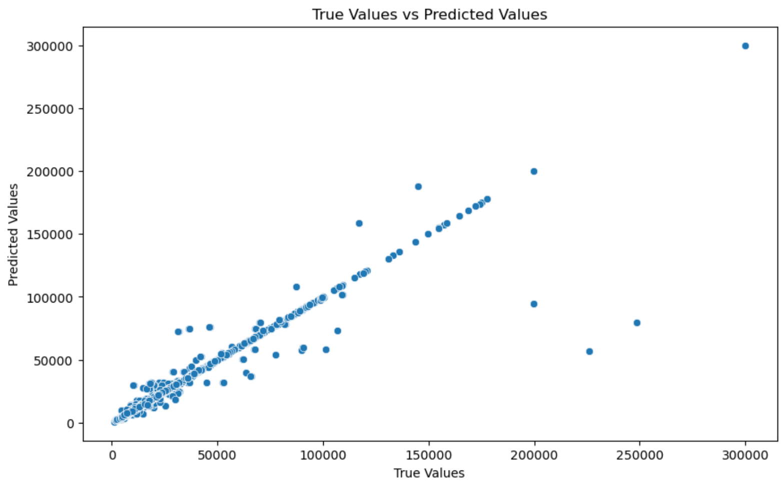


Figure 12: The true vs predicted values obtained by the best KNN model.

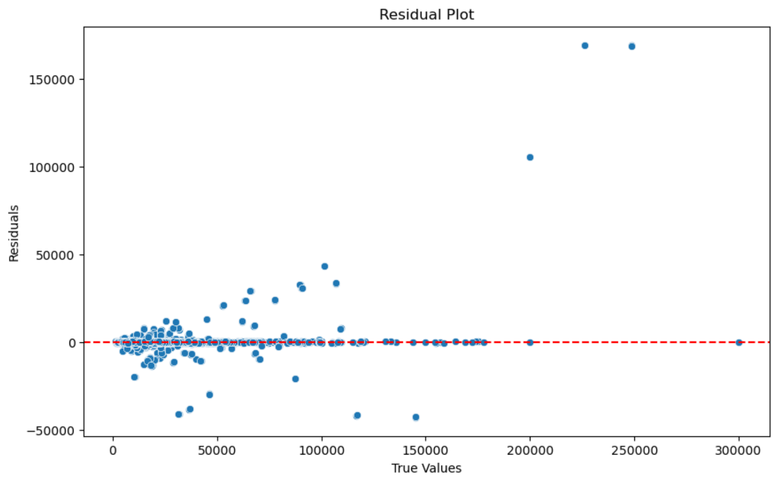


Figure 13: The residual values obtained by the best KNN model.

6.2.3 NN reg (MLPRegressor)

Neural network-based models predict results using layers of neurons connected by trainable edges. They can learn complex (non-linear) functions through various training functions (e.g., ReLU, Tanh, Sigmoid, etc.). For learning, they employ a Gradient Descent strategy to minimize the distance function between the model's prediction and the actual result.

Model	Hyperparams	Feature Extraction	MAE	RMSE
MLPRegressor	Hidden_layers = (32, 8), LR = 0.1	All features	2274.51	7035.46
MLPRegressor	Hidden_layers = (32, 8), LR = 0.01	All features	1936.56	6729.12
MLPRegressor	Hidden_layers = (32, 8), LR = 0.001	All features	1994.62	7425.41
MLPRegressor	Hidden_layers = (32, 8), LR = 0.01	Only Numeric	5900.96	11271.81
MLPRegressor	Hidden_layers = (32, 16, 8), LR = 0.1	All features	2281.00	7097.58
MLPRegressor	Hidden_layers = (100), LR = 0.1	All features	2267.77	7016.69
MLPRegressor	Hidden_layers = (100), LR = 0.01	All features	**1801.52**	6925.55
MLPRegressor	Hidden_layers = (50), LR = 0.01	All features	1878.79	7421.80
MLPRegressor	Hidden_layers = (20), LR = 0.01	All features	2015.67	**6684.32**
MLPRegressor	Hidden_layers = (10), LR = 0.01	All features	2057.93	7145.72

Table 7: The results obtained by the MLPRegressor model on our dataset.

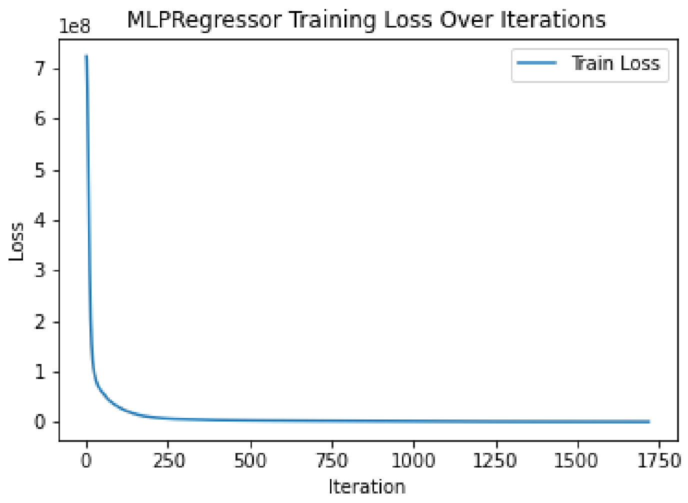


Figure 14: Best MLPRegressor Training Loss over iterations.

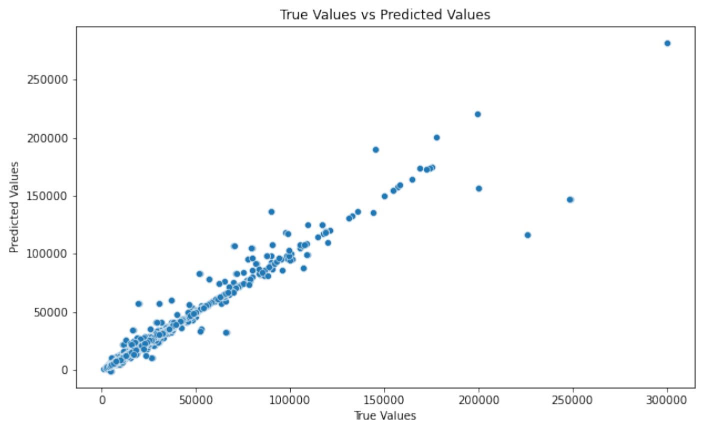


Figure 15: The true vs predicted values obtained by the best MLPRegressor model.

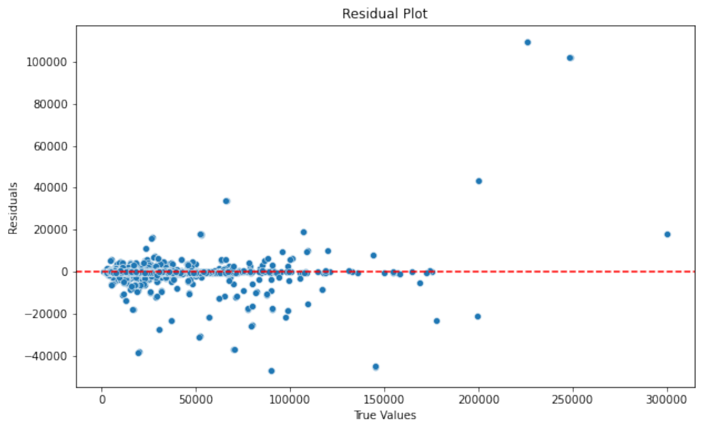


Figure 16: The residual values obtained by the best MLPRegressor model.

6.2.4 Linear Regression

Linear Regression models attempt to model regression through a linear function. The estimation of the function's coefficients is achieved by minimizing the sum of squares between the predicted values and the actual values for the regression objective (equivalent to minimizing the MSE value). The loss function to be minimized has the following formula:

$$\sum_i (y_i - p_i)^2$$

Model	Hyperparams	Feature Extraction	MAE	RMSE
LinearRegression	Defaults	All features	**3446.10**	**8549.60**

Table 8: The results obtained by the LinearRegression model on our dataset.

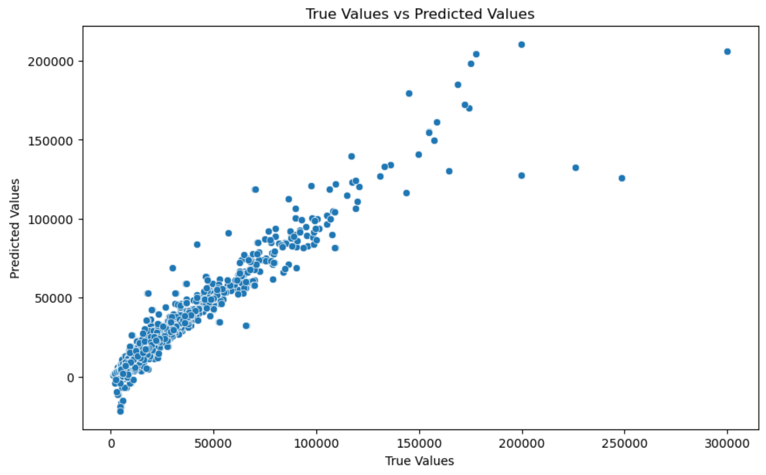


Figure 17: The true vs predicted values obtained by the best LinearRegression model.

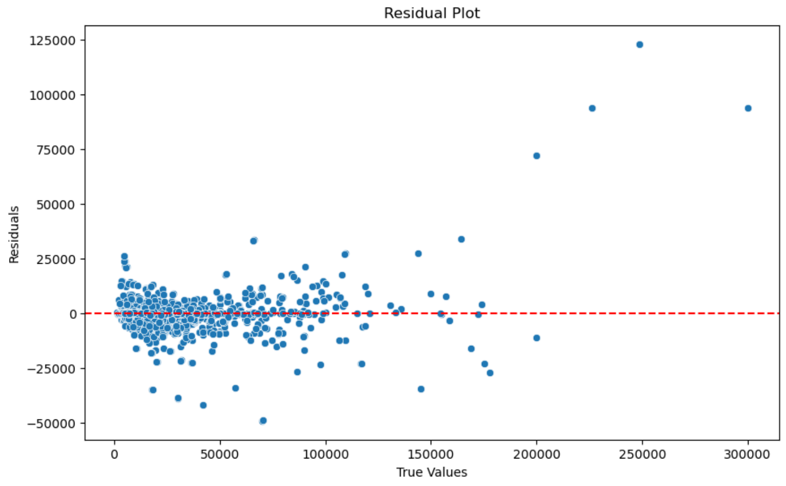


Figure 18: The residual values obtained by the best LinearRegression model.

6.2.5 Ridge Regression

In addition to linear regression, ridge regression incorporates a regularization component into the loss function calculation. Therefore, to the result of the loss function defined for linear regression, the squared value of the L2 norm (Euclidean norm) of the coefficient vector is added, multiplied by a regularization parameter:

$$\sum_i (y_i - p_i)^2 + \lambda ||\beta||^2$$

Model	Hyperparams	Feature Extraction	MAE	RMSE
RidgeRegression	$\lambda = 1.0$	All features	4197.08	9417.91
RidgeRegression	$\lambda = 0.0000001$	All features	**3502.92**	**9260.29**

Table 9: The results obtained by the RidgeRegression model on our dataset.

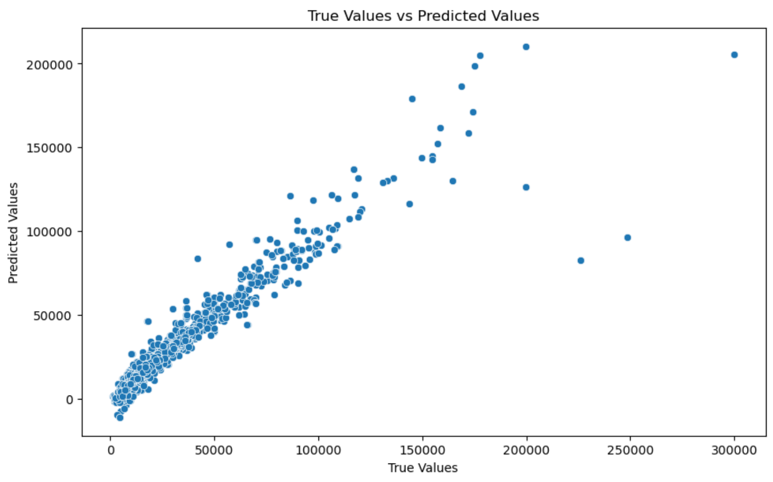


Figure 19: The true vs predicted values obtained by the best RidgeRegression model.

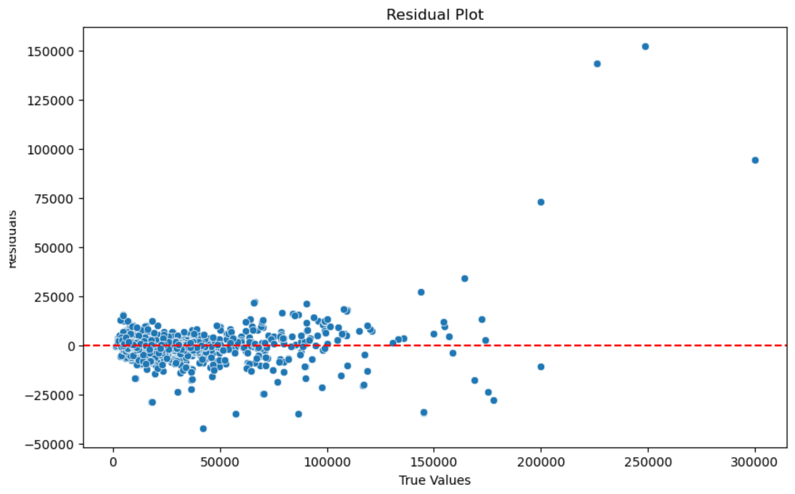


Figure 20: The residual values obtained by the best RidgeRegression model.

6.2.6 Lasso Regression

Similar to ridge regression, but for the regularization component, the L1 norm is used. Therefore, the loss function will be:

$$\sum_i (y_i - p_i)^2 + \lambda||\beta||$$

The hyperparameter tuning process consisted of using LassoCV which performs cross-validation in order to find and select the value of alpha which yields the best results. The initial run focused on values in the range of 0.1 to 1.1 with a step of 0.1. Once the range has been extended to 5, the same best estimator is found and kept as the final best result.

Model	Hyperparams	Feature Extraction	MAE	RMSE
Lasso	$\lambda = 0.98$	All features	4175.46	10135.57
Lasso	$\lambda = 0.30$	All features	**3577.92**	**9879.44**

Table 10: The results obtained by the Lasso model on our dataset.

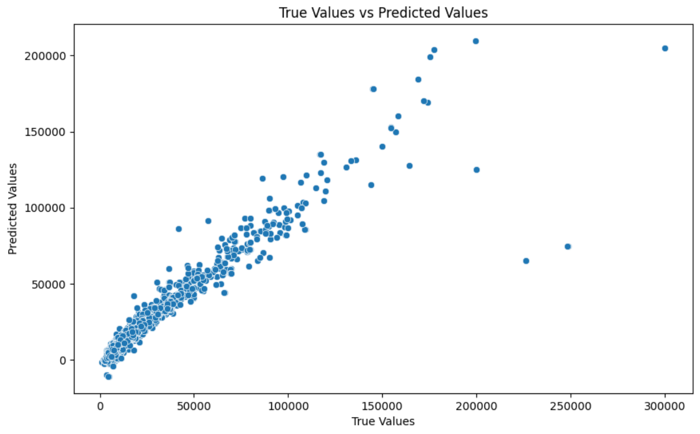


Figure 21: The true vs predicted values obtained by the best Lasso model.

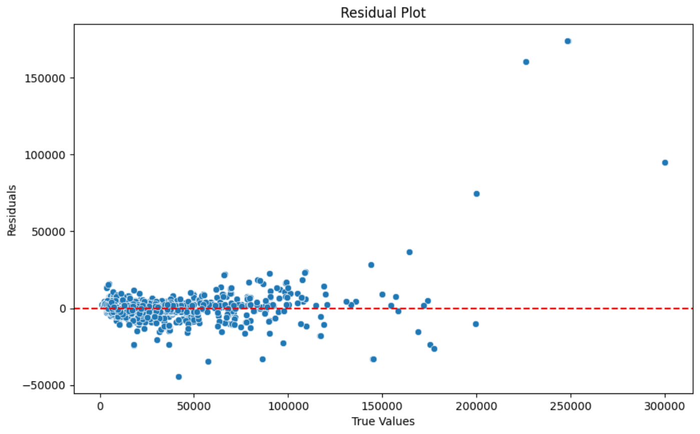


Figure 22: The residual values obtained by the best Lasso model.

6.2.7 RandomForest

Models based on Decision Trees structures. These are tree-like structures where, at each node except the leaves, the data is split based on the most important features. Each node of the tree represents a subspace of the value space, and each "decision" to choose a child at a step represents a narrowing of the search space. The decision-making process is carried out recursively until the algorithm reaches a leaf. Random Forest models create multiple decision trees trained on random subsets of the dataset. To obtain the final prediction, an average of the predictions from all decision trees in the structure is calculated

Model	Hyperparams	Feature Extraction	MAE	RMSE
RandomForest	Defaults	All features	2022.71	6682.99
RandomForest	N_estimators = 10	All features	2172.99	**6483.13**
RandomForest	N_estimators = 25	All features	2106.89	7326.10
RandomForest	N_estimators = 100	All features	2040.44	6954.67
RandomForest	N_estimators = 500	All features	2020.45	6956.03
RandomForest	N_estimators = 1000	All features	**2007.28**	6885.23
RandomForest	N_estimators = 1500	All features	2024.68	6962.41

Table 11: The results obtained by the RandomForest model on our dataset.

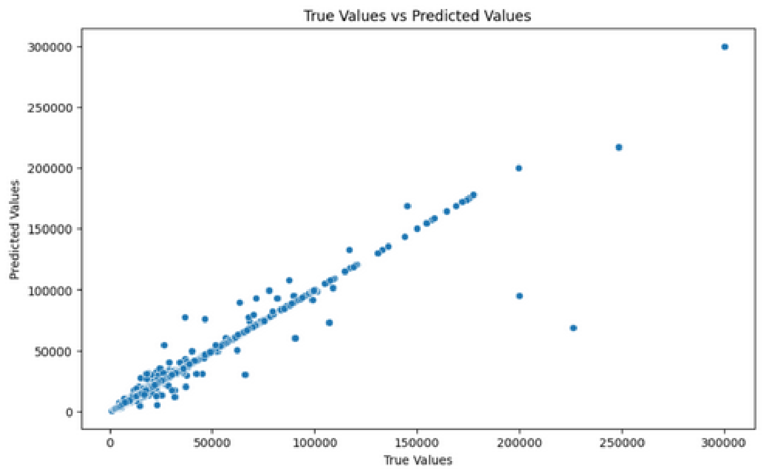


Figure 23: The true vs predicted values obtained by the best RandomForest model.

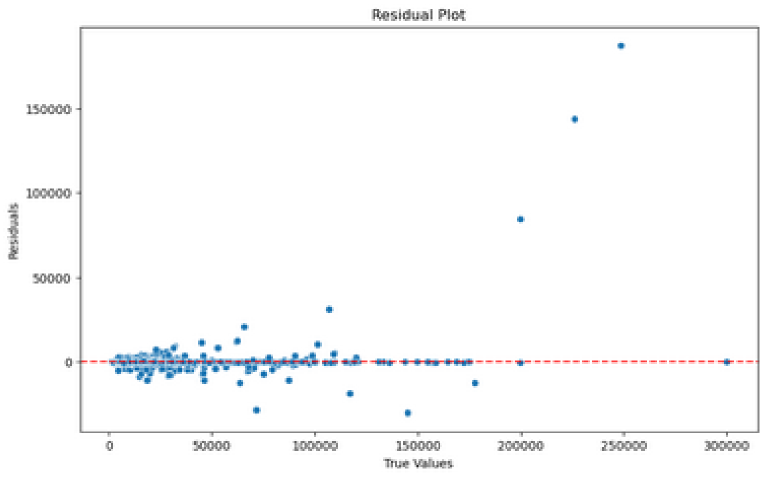


Figure 24: The residual values obtained by the best RandomForest model.

6.2.8 XGBRegressor

This model is also based on the construction of Decision Trees in a sequential manner. Each new tree corrects the errors of previously trained trees.

The hyperparameter tuning process consisted of using GridSearch which performs cross-validation in order to find the set of parameters that yields the best results. GridSearch has been performed on values in the range of [0.5, 0.8] for *colsample_bytree*, [3, 15] for *max_depth*, [0.001, 0.1] for *learning_rate*, [1, 5] for *min_child_weight*, [100, 500] for *n_estimators* and [0.5, 0.8] for *subsample*. The configuration that obtained the best results has been included in the table presented below.

Model	Hyperparams	Feature Extraction	MAE	RMSE
XGBRegressor	enable_categorical=True	All features	3136.66	8146.38
XGBRegressor	enable_categorical=True, colsample_bytree= 0.5, max_depth = 10, learning_rate = 0.1, min_child_weight = 1, n_estimators = 500, subsample = 0.7	All features	2500.27	8624.38
XGBRegressor	enable_categorical=True, colsample_bytree= 0.8, max_depth = 15, learning_rate = 0.1, min_child_weight = 1, n_estimators = 500, subsample = 0.8	All features	**1691.21**	**7967.24**

Table 12: The results obtained by the XGBRegressor model on our dataset.

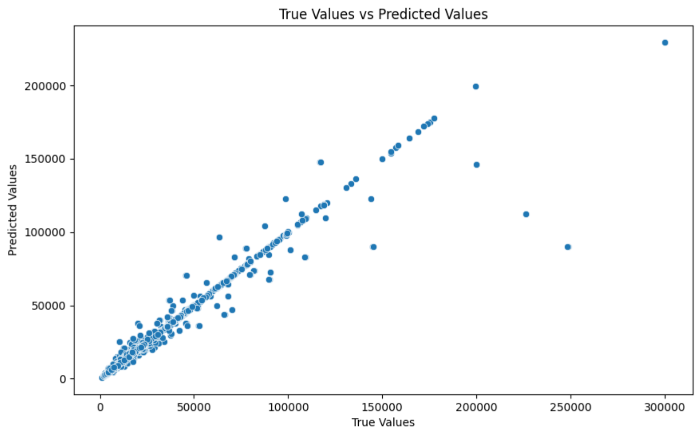


Figure 25: The true vs predicted values obtained by the best XGBRegressor model.

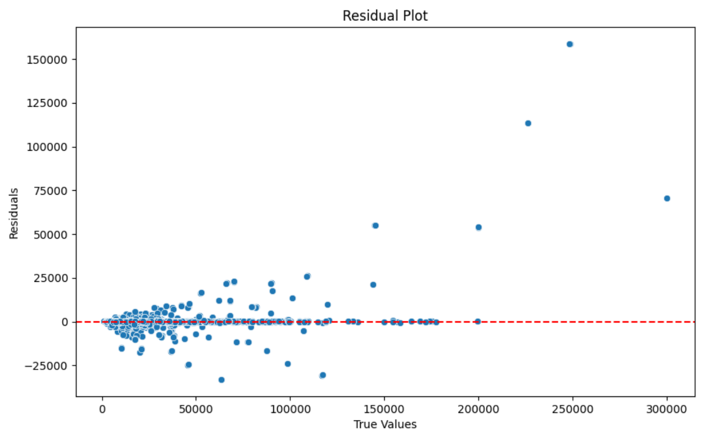


Figure 26: The residual values obtained by the best XGBRegressor model.

6.2.9 BaggingRegressors

This model can be used to combine predictions from multiple Random Forest structures trained on random subsets.

Model	Hyperparams	Feature Extraction	MAE	RMSE
Bagging + RF	N_est = 5, n_bag = 5	All features	2740.97	8091.71
Bagging + RF	N_est = 10, N_bag = 10	All features	2639.07	7563.98
Bagging + RF	N_est = 50, N_bag = 10	All features	2644.60	7532.39
Bagging + RF	N_est = 100, N_bag = 100	All features	**2591.10**	**7241.26**

Table 13: The results obtained by the BaggingRegressors model on our dataset.

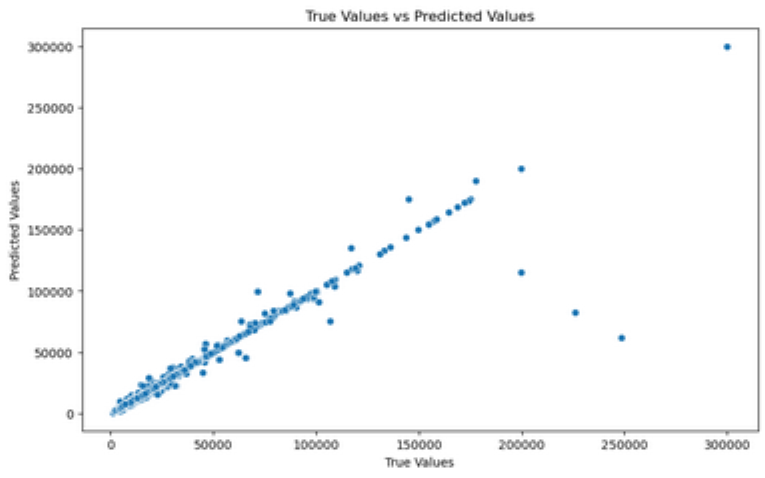


Figure 27: The true vs predicted values obtained by the best BaggingRegressors model.

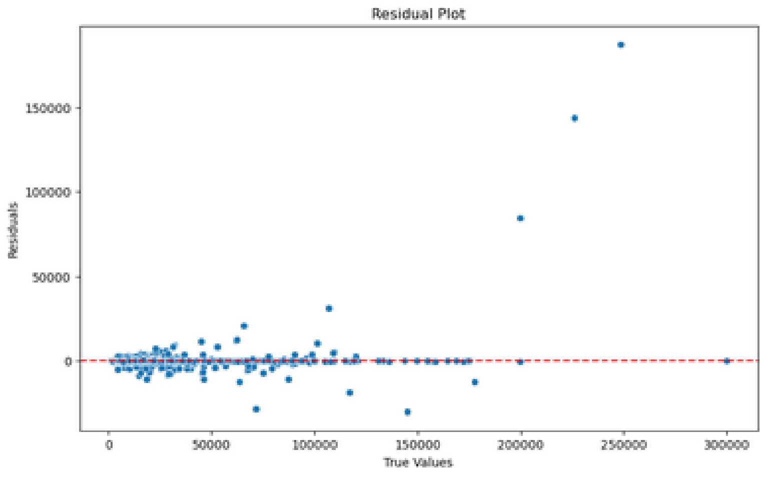


Figure 28: The residual values obtained by the best BaggingRegressors model.

6.2.10 SVR+KNN

We combined the predictions of the best-trained SVR and KNN models and obtained the following results:

Model	Hyperparams	Feature Extraction	MAE	RMSE
SVR + KNN	SVR = 50%, KNN = 50%	All features	1488.91	9497.79
SVR + KNN	SVR = 67%, KNN = 33%	All features	1456.78	9492.35
SVR + KNN	SVR = 52%, KNN = 48%	SVR: All features, KNN: Km, Age, Power, Brand, Model, Version, Fuel	**1329.61**	**8718.52**

Table 14: The results obtained by the SVR + KNN model combination on our dataset.

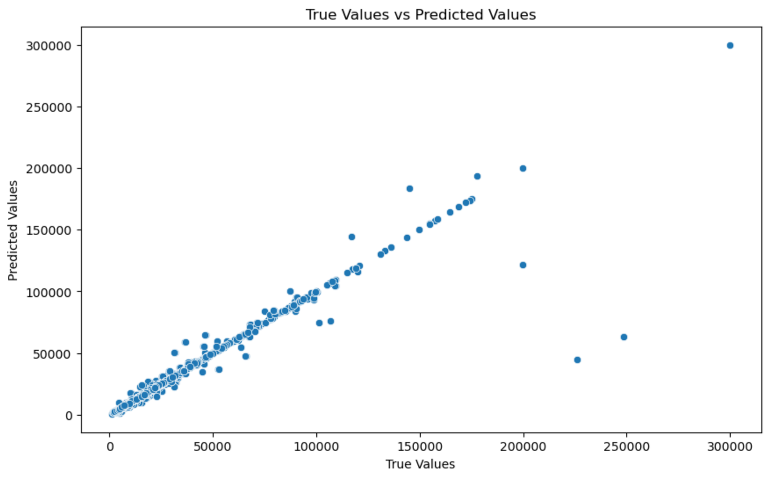


Figure 29: The true vs predicted values obtained by the best SVR + KNN model combination.

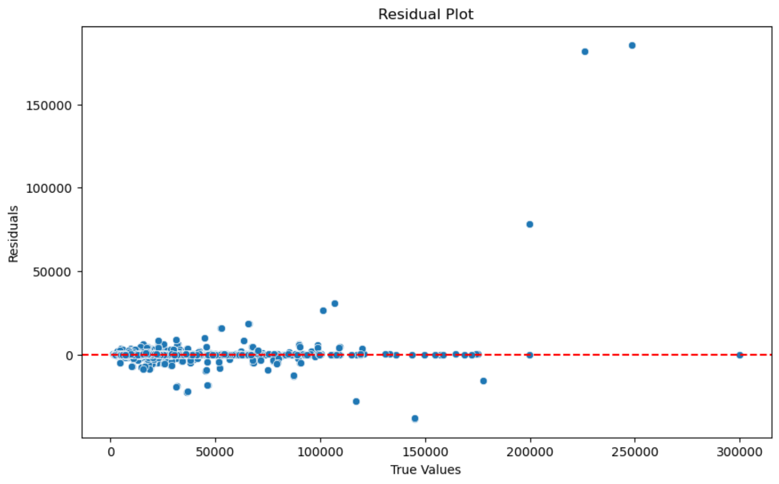


Figure 30: The residual values obtained by the best SVR + KNN model combination.

7 Results

Model	Best MAE	Best RMSE
SVR / LinearSVR	1504.84	8434.16
KNN	1492.11	8467.35
NNreg(MLPRegressor)	1801.52	6684.32
Linear Regression	3446.10	8549.60
Ridge Regression	3502.92	9260.69
Lasso Regression	3577.92	9879.44
RandomForest	2007.28	**6483.13**
XGBRegressor	1691.21	7967.24
BaggingRegressors	2591.10	7241.26
SVR+KNN	**1329.61**	8718.52

Table 15: The best results over all the tested models.

8 Conclusions

The model that showed the best results in regards to the MAE (Mean Absolute Error) metric is the combination of SVR+KNN models with weights of 52% and 48%, resulting in MAE of 1329.61 and RMSE of 8718.52. In practice, the model can predict the price of a car in the Romanian market with an average precision of ±1329€. Considering the average price of a car is 26793€, the estimator has an average error of 4.96%.