ISYE 6740 Group Project Final Report

DRIVEN INSIGHTS: PREDICTIVE MODELING FOR CAR PRICE ESTIMATION

**Team 4**

Hassan Abdrabo (GTID: 90364994)

Fuad Jannatov (GTID: 903847334)

Duc Nguyen Hong (GTID: 903943817)

# Abstract

Known for its rich history of automotive innovation, Europe has established itself as a global hub for both car manufacturing and consumption. This project primarily focuses on the booming used car market in Germany and the Czech Republic. We aim to build machine learning models capable of estimating the price of a pre-owned car using historical consumer data and specific features. Additionally, we assess and compare the prediction accuracy of various models to identify the optimal one. This tool will serve as a recommendation for any new car buyer looking for a good deal on a secondhand vehicle.

# I. Introduction

## Background

Demand for automobiles had been increasing for years in Europe. In 2017, the passenger car market experienced a 3% growth, reaching almost 15.5 million in sales. This marks the fourth consecutive year of expansion in Europe and represents the highest sales volume since 2007. Among the prominent markets, Italy, Spain, and France stood out with growth rates of 7.9%, 7.7%, and 4.7%, respectively, surpassing the overall market performance, while Germany also experienced a solid growth rate of 2.7% ([[1]](#footnote-2)). The secondhand market also experienced a similar trajectory. The economic boom, growing urbanization, and a shift in consumer preferences towards more budget-friendly transportation were ascribed to the rising interest in the used car market.

## Problem Statement

As opposed to purchasing a brand-new car from dealerships, buying used ones can be tricky and risky. Estimating prices is a crucial task for both buyers and sellers to facilitate informed decisions in terms of trading opportunities. Furthermore, meaningful analyses contribute to market transparency, fair pricing practices, and a more efficient and equitable automotive marketplace.

## Project Goals

A used car's price depends on an innumerable number of factors. However, whether it be internal factors such as age/mileage, make, model ... or external ones like market demand, location ..., we believe a close estimation can be derived as an aggregation of some vital variables. We aim to find the optimal algorithm from a variety of methods and hopefully the model can be deployed into a recommendation tool for car price suggestions for new and inexperienced buyers. Even though the data is not up to date, this report can be treated as a blueprint that can be applied to future data.

**Primary Research Problem:** Car price estimation based on data scraped from advertisements.

**Optional Complementary Questions:**

* Which factors determine the price of a car?
* With what accuracy can the price be predicted?
* Can a model trained on all cars be used to accurately predict prices with only a few samples?

# II. Data Source

The [data](https://www.kaggle.com/datasets/mirosval/personal-cars-classifieds/data) was scraped by Kaggle user MIROSLAV ZORICAK from several websites in the Czech Republic and Germany from 2015 to 2017. There are 3.5 million raw data points in the dataset, each data point representing a scraped ad of a car, having 15 independent and 1 dependent (price\_eur) features. The data types are mixed, containing strings, floats, integers, and datetime types.

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Description** |
| maker | String | Brand |
| model | String | Model |
| mileage | Float | Mileage in km |
| manufacture\_year | Integer | Manufacture year |
| engine\_displacement | Float | Total volume of cylinders in ccm |
| engine\_power | Float | Engine power in kW |
| body\_type | String | Body type |
| color\_slug | String | Main color |
| stk\_year | String | Year of the last emission control |
| transmission | String | Transmission: automatic or manual |
| door\_count | Integer | Door count |
| seat\_count | Integer | Seat count |
| fuel\_type | String | Fuel type: gasoline, diesel, cng, lpg, electric |
| Date\_created | Datetime | When the ad was scraped |
| date\_last\_seen | Datetime | When the ad was last seen |
| price\_eur | Float | Price of the car in EUR listed in the ad |

The scrapers were tuned slowly over the course of the year and some of the sources were completely unstructured, so as a result the data is dirty, there are missing values, and some values are obviously wrong.

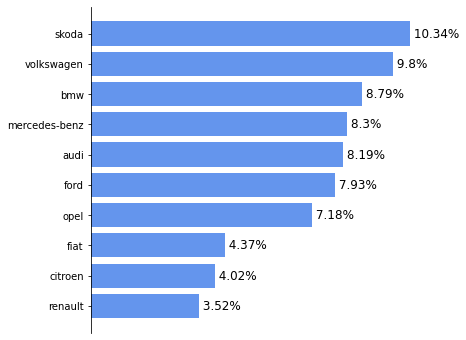
# III. Methodologies

We follow a typical process pipeline of a data science project:

* Data Exploratory Analysis
* Data Cleaning
* Data Modeling
* Model Evaluation

## Data Exploratory Analysis

The dataset contains 3,552,912 records with each row indicating the proposed price of a car and its specifications. The set is rather sparse. The missing value percentage on each column can go from 10% to 90% of all observations. This is expected considering not every advertisement follows the same format regarding information provision. We can drop the columns with an overwhelming number of null values and attempt to compute the rest.

The most notable categorical features include maker/model. We observe that the distribution of car brands reflects the Central European car market well as the most popular brands like Volkswagen or BMW take a large portion of the data. Each maker can have multiple models but some models are exclusive to one particular maker and we detect some outliers as well. We use the maker-model pair as the base to impute other features like body\_type, fuel\_type or transmission.

Numerical variables like mileage, engine\_power... contain a considerable number of extreme values. This might be due to incorrect inputting or failed scraping. For instance, mileage numbers on some rows can reach over 1,000,000 miles which is unlikely. We rely on car expertise to weed out the outliers. When we filter mileage under 400,000 miles, we achieve a more sensible distribution.

Figure 1: top present brands

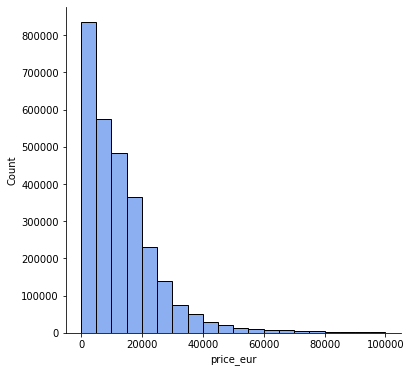
Other numerical variables like door\_count, seat\_count contain a mixture of erroneous data types. We must correct them while removing outliers as well. Both features center around 4 or 5 units. Based on real-life knowledge and data distribution, we accept door count to go up to 6 and seat count to go up to 8.

Figure 2: distribution of price

Date variables including manufacture\_year and stk\_year also suffer from some illogical data. The year can go back several centuries.

With the target variable, we detect an abnormal number of 1295.34 values, especially across different combinations of maker, model and mileage... We suspect this is due to incorrect inputting. In order to maintain the scope of passenger car assessment only, we plan on keeping all the data with the price under €100,000 (which eliminates most of the outliers as well).

## Data Cleaning

As aforementioned, the dataset is riddled with missing values, extreme values and outliers. The pre-processing step is essential to do for effective training. From the insights gained in the EDA, subsequent actions are taken:

* Drop columns with a large portion of missing values: We exclude color\_slug for it consists of mostly null values.
* Fix outliers on the model column: Since we utilize the maker-model pair as the base to impute missing values on other columns. Some records have an abnormal model associated with its maker. We update those values according to the most frequent brand of the model type.
* Update extreme values on numerical features: We gather information about each car metric and its plausible range of value. For instance, a commercial car can last 400,000 miles (about 643737.6 km) at maximum. Therefore, we change all values above it to null. We then substitute missing values with the mean or median aggregated by each maker-model pair. A similar approach can be applied to the other date and numerical variables.
* Impute missing values on categorical features: Same as the step above, we replace null values with the most frequent value aggregated by each maker-model pair.
* Drop all extreme values on the target variable: We keep the range of price\_eur from €50 to €100,000 and drop the rows with the price of 1295.34.

In the end, we preserve more than 2 million rows.

## Data Modeling

We have implemented these model types to explore our research questions:

* Regression models (Linear & Non-linear Models)
* K-nearest Neighbors Regression
* Tree models (Decision Tree Regressor, Random Forest Regressor)
* Boosting models (AdaBoost, XGBoost)
* Artificial Neural Network

1. **Regression models:** Regression models, as potent statistical tools employed in predictive analysis, offer numerous advantages. Firstly, they furnish a systematic and quantitative methodology for comprehending and scrutinizing intricate relationships between variables. Secondly, these models facilitate precise predictions, enabling the forecasting of future outcomes grounded in historical data and patterns. Thirdly, regression models empower the identification of the most influential factors impacting the dependent variable, providing valuable insights into the fundamental drivers of the studied phenomenon.
   1. Feature engineering and pre-processing: We calculate the manufacture period and stk period by subtracting date\_created with the respective columns to create two new features. With null values in numerical features, we fill them with zero and create indicator columns to signify missing values. After dummy variables are created from the categorical features, we drop all the redundant ones. We can assess the linear correlation between numerical variables with the heatmap below:

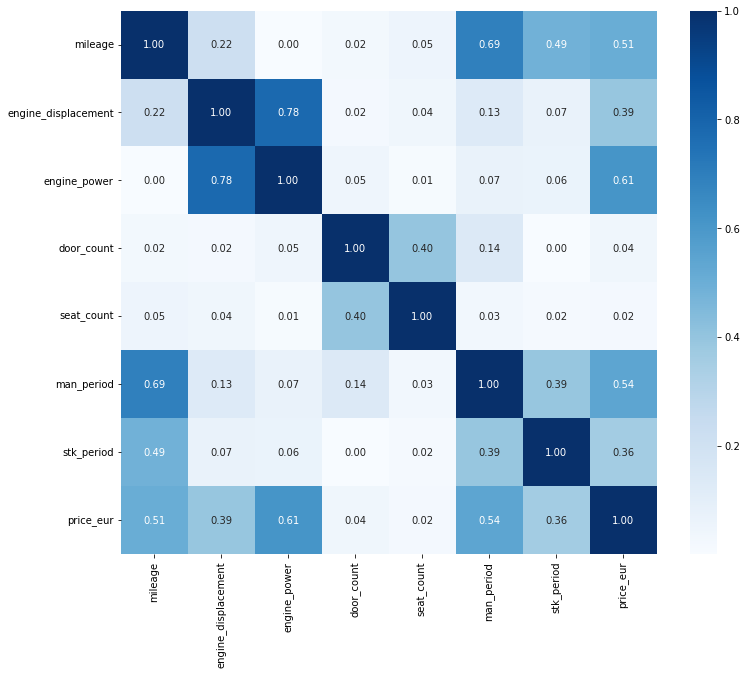


Figure : Correlation Heatmap

* 1. Fit model: We fit a regression model using ‘sklearn’ package. Positive points of the model include: R-squared is relatively high (0.74 on the test set) along with a high F-statistic score signifying that the model is significant; There is no autoregression (Durbin-Watson close to 2) or heavy heteroscedasticity. On the other hand, we also detect multicollinearity (high Cond. No. : there are a few heavily correlated columns), insignificant variables (by considering the p-value) and negative predicted values. From ANOVA, we can also conclude some features like engine\_power or mileage have large influence on the dependent variable.

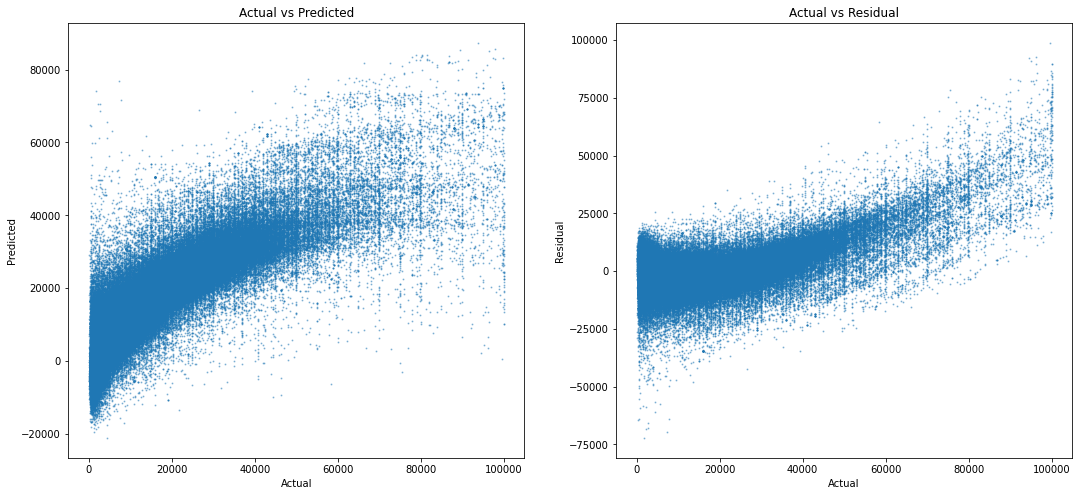
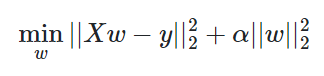


Figure : actual vs. predicted values for linear regression

* 1. Other regression variations: We implement several tuning methods to refine the model, followed by an assessment of each technique. Furthermore, we incorporate a Cross Validation approach using 5 folds to mitigate bias in our selection process and obtain the average for each evaluation metric.
* Regularization with LASSO Regression: Lasso R egression introduces a penalty term to the linear regression objective function, which is based on the absolute values of the regression coefficients. This penalty term encourages sparsity in the coefficient values, effectively shrinking some coefficients to zero. Therefore, the least significant variables will be eradicated.



* Regularization with Ridge Regression: Ridge Regression introduces an additional term, known as the ridge penalty or L2 penalty, to the loss function. This penalty term is proportional to the squared magnitude of the regression coefficients, effectively shrinking them toward zero. Unlike LASSO, the coefficients will not reach zero, but the model can still mitigate the multicollinearity problem effectively.



* Linear Regression with Truncated SVD: Truncated SVD is similar to Principal Component Analysis (PCA) in that it is a statistical method employed to reduce the complexity of data and explore patterns, but it is more effective for a sparse dataset than traditional PCA. It enables the transformation of a dataset with many dimensions into a simpler form while retaining most of the essential information. Truncated SVD is useful for reducing the correlation between variables and getting rid of less important ones in the process. We scale the data and apply the transformation. We choose the number of components to be 10 as they retain 90% of the original data then apply a linear regression model.

1. **K Nearest Neighbors Regressor (KNR)**: The K-Nearest Neighbors Regressor is very similar to K Nearest Neighbors Classifier, except that it performs regression based on each data point Neighbors.
   1. Feature engineering and pre-processing: Maker and model are concatenated into one feature. Numerical variables such as engine\_power, engine\_displacement and mileage are standardized. The rest are treated as categorical and one hot encoded. The response variable price\_eur is log-scaled to prevent the prediction of negative values. We then implement Truncated SVD with 6 components to the sparse matrix to produce a dimensionality-reduced dataset.
   2. Fit model: We start with a model created with 35 nearest neighbors, Euclidean distance measure, and weighted distance prediction (the further away the points are from a new point, the less of an impact these points will have on the price prediction). The model yields extremely very good results on the testing set with an R2 score of 0.88. We then tune the model by using the cross-validation method. The examined hyperparameters are distance metric (Manhattan & Euclidean), number of neighbors (from 2 to 50) and weight assessment (uniform & distance). R2 metric of the optimal model is similar to the previous model but this model contains a lower number of neighbors.

Below we visualize some of the model outputs. First, we show the price actual price vs the predicted price. Then we show an example model output on one test point. The output includes the test point, and all of the neighbors used to evaluate the price of the car sorted by weight/distance to the test point.

A blue and black dotted pattern

Description automatically generated with medium confidence

Figure : actual vs predicted values for kneighborsregressor

1. **Tree Models**: A decision tree is a fundamental machine learning algorithm used for both classification and regression tasks. It represents a flowchart-like structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node holds the predicted outcome. Decision trees are transparent and interpretable, making them valuable for understanding decision-making processes in a model. On the other hand, a random forest is an ensemble learning technique that builds multiple decision trees and combines their predictions. It introduces randomness during the training process by considering a random subset of features for each tree and aggregating their outputs.

A diagram of a algorithm

Description automatically generated

Figure : decision tree sample

* 1. Feature engineering and pre-processing: The data preprocessing steps are the same as for Linear regression except we set door\_count and seat\_count as categorical features instead. `DecisionTreeRegressor` and `RandomForestRegressor` can only process numerical input. Therefore, we must one hot encode all object variables.
  2. Fit model: After splitting the data, ranges of potential hyperparameters were stored in a dictionary and used in Randomized Search CV with 5-fold cross-validation, evaluation criteria being R-squared. Using cross-validation is crucial for tuning a model because it ensures the correct parameter choice. The parameter choices include number of trees, maximum tree depth, minimum samples split, splitting method… To save time, we train on a sample of the training set to derive the optimal combination then train it again on the whole set. The following plot is an example of choosing the maximum number of estimators based on R2:

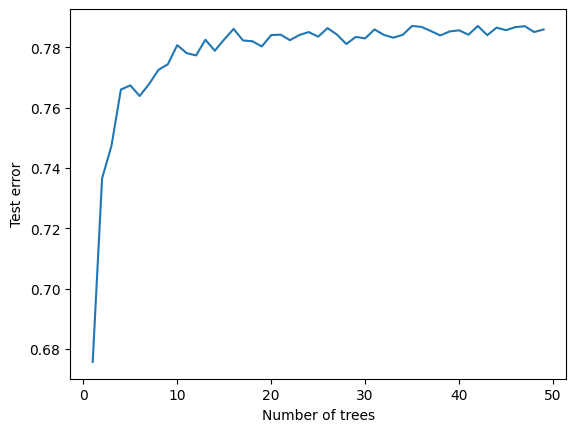


Figure : test error vs no. of trees

Ultimately, when we run the models on the test set, they return relatively high accuracies (0.62 for decision tree and 0.83 for random forest).

1. **Boost models:** AdaBoost (Adaptive Boosting) and XGBoost (Extreme Gradient Boosting) are powerful ensemble learning techniques widely used in machine learning. AdaBoost operates by sequentially combining weak learners, such as decision trees, assigning higher weights to misclassified data points to improve overall predictive accuracy. It adapts iteratively, focusing on the weaknesses of previous models. In contrast, XGBoost enhances the gradient-boosting framework by incorporating regularization techniques, parallel processing, and tree pruning. It is renowned for its speed, efficiency, and performance, making it a popular choice for both regression and classification tasks.
   1. Feature engineering and pre-processing: The data preprocessing steps are the same as for tree models.
   2. Fit model: Both models were optimized for the number of learners. AdaBoost performed poorly with an R2 Score of just 0.4. XGBoost, however, performed much better with R2 Score of 0.77. Adding more learners. The best model selected by cross-validation for AdaBoost had 50 learners. Unfortunately, adding more learners does not have any significant improvements on the model performance. The best model selected by cross validation for XGBoost had 90 learners. We notice that adding more learners (around a 30) more does have a slight performance increase of around 0.02 or 2% on the testing set.
2. **Artificial Neural Network**: Taking inspiration from the human brain, an Artificial Neural Network (ANN) is comprised of interconnected layers of weighted nodes that can capture complex non-linear relationships between input features. For regression tasks, ANNs utilize activation functions that allow the network to produce continuous output values. The training process involves feeding input data into the network, computing predictions, comparing them to the actual values, and adjusting the weights to minimize the difference between predicted and actual outcomes. This is typically achieved through optimization algorithms like gradient descent.

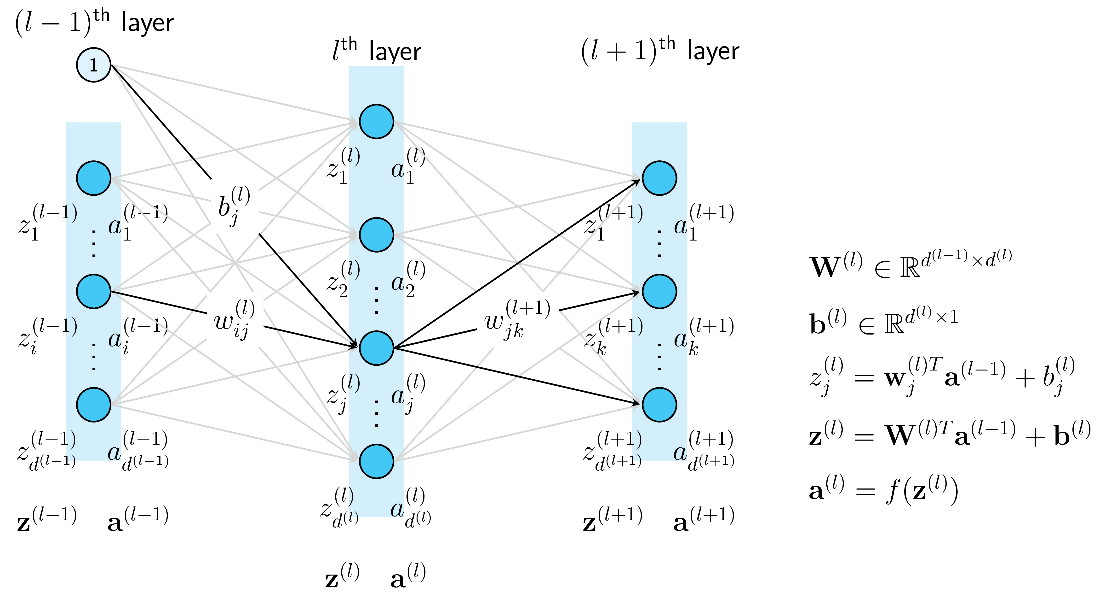


Figure : generic neural network structure (machinelearningcoban.com)

* 1. Feature engineering and pre-processing: We apply the same steps as in the regression modeling process.
  2. Fit model: We implement ‘keras’ solution to train our ANN. The network consists of three fully connected layers with a descending number of neurons on each layer (600, 200 and 50). We set the loss metric as ‘mean\_squared\_error’ and the optimizer as ‘adam’. Each fed batch contains 32 data points. We also set the maximum epoch to 100 with early stopping after 10 attempts. It takes approximately 3 hours to derive the optimal parameters, stops at 37th epoch. The inference on the test set also takes longer than other models. However, it results in high accuracy with robustness.

## Evaluation metrics:

Please refer to the metric table below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **RMSE** | **MAE** | **R2** |
| Linear Regression | 6418.69 | 4005.50 | 0.74 |
| LASSO Regression | 6366.59 | 3967.09 | 0.74 |
| Ridge Regression | 9436.60 | 6095.57 | 0.44 |
| Linear Regression with Truncated SVD | 7384.20 | 4661.31 | 0.66 |
| K-Nearest Neighbors Regressor | 4352.26 | 2042.00 | 0.88 |
| Decision Tree | 7751.07 | 4893.33 | 0.62 |
| Random Forest | 5164.35 | 2872.68 | 0.83 |
| AdaBoost | 9783.04 | 5847.60 | 0.40 |
| XGBoost | 6093.04 | 3551.03 | 0.77 |
| Artificial Neural Network | 4999.39 | 2784.87 | 0.84 |

# IV. Model Evaluation

Combined with the metric table above, we derive the conclusion for each method:

|  |  |  |
| --- | --- | --- |
| **Models** | **Advantages** | **Disadvantages** |
| Linear Regression models | - Relatively fast to train and run  - Return acceptable precisions  - Able to identify the most significant variables  - Easy to interpret and implement | - Not the most accurate method |
| K-Nearest Neighbors Regressor | - Return high accuracies  - Can show similar cars in the dataset. | - Extremely slow and memory inefficient on the dataset without dimensionality reduction. |
| Tree models | - Return high accuracies  - Easy to train, doesn’t need much tuning (especially with Random Forest)  - With Decision Tree, it’s possible to point out the most significant variables  - Able to handle non-linear relationship | - Relatively slow to train  - Difficult to interpret with Random Forest  - Predicted values have a lower variance. |
| Boost models | - XGBoost returns high accuracies  - XGBoost has much better performance and can be trained extremely fast on the reduced dataset. | - Slow and memory inefficient on the dataset without dimensionality reduction. |
| Artificial Neural Network | - Returns high accuracies  - Robust to unseen data  - Able to continue training with new data | - Costly to train and infer  - Require a large amount of data to train |

# V. Conclusion

**Research question: Can car prices be estimated based on data scraped from advertisements?**

Throughout our analysis, we explore a diverse range of models: from linear to non-linear models; from base to ensemble models; from traditional machine learning to deep learning models. Although complex non-linear algorithms return more accurate predictions (based on the R2 scores), a simple regression model can serve the purpose if we are to find the most average prices for recommending. New car buyers can take the predicted value as a threshold to determine whether they’ve got a good or bad deal or whether what they are looking for is within their budget. For business’ purposes, it is worth noting that the models remain capable of making accurate predictions within a confident interval, nonetheless. The business can utilize the results for price determination (ANOVA can detect significant variables contributing to the value of the car) or building marketing strategies. There can be so many possibilities.

We believe the models can further be tuned to achieve even higher precisions. However, it requires collecting additional data, conducting further research on hidden patterns and trends, and exploring alternative approaches.

# References

Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong. "Linear Regression" *Mathematics for Machine Learning,* edited by Lauren Cowles, Cambridge University Press, 2020, pp. 289-315.

Aurélien Géron.” Decision Trees”. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, edited by Nicole Butterfield, Nicole Taché, Michele Cronin, Beth Kelly, O'Reilly Media; 3rd edition, 2022, pp. 195-207.

-- “Ensemble Learning and Random Forests”. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, edited by Nicole Butterfield, Nicole Taché, Michele Cronin, Beth Kelly, O'Reilly Media; 3rd edition, 2022, pp. 211 -232.

Ganesh, Mukkesh & Venkatasubbu, Pattabiraman. (2019). Used Cars Price Prediction using Supervised Learning Techniques. International Journal of Engineering and Advanced Technology. 9. 216-223. 10.35940/ijeat.A1042.1291S319.

Sanchith Venkatesan, Shivam Handa, and M Baskar\*. 2022. Car Price Prediction. In International Conference on Information Management & Machine Intelligence (ICIMMI 2022), December 23, 24, 2022, Jaipur, India. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3590837.3590864>

Shaprapawad, Snehit & Borugadda, Premkumar & Koshiga, Nirmala. (2023). Car Price Prediction:An Application of Machine Learning.

AlShared, Abdulla, "Used Cars Price Prediction and Valuation using Data Mining Techniques" (2021). Thesis. Rochester Institute of Technology

1. https://www.goodcarbadcar.net/2017-europe-sales-figures-by-brand/ [↑](#footnote-ref-2)