Author: Nico Herrig

Executive Report
Machine Learning for car price predictions

Introduction

Assignment 1 deals with a machine-learning-based approach for car price predictions. The problem itself can be classified as a univariate, multiple regression problem as its aim is to predict one continuous variable (*price*) using several signals. As the data comes in a labelled format, a supervised machine learning approach is appropriate.

Data Exploration, Feature Selection and Data Pre-Processing

The full data set comes in a size of 10GB, containing 3 million rows and over 60 features. As the machine used for developing and running the script is not able to process large amounts of data due to a lack of RAM, a 500 MB subset of the full data is used by randomly sampling 50% of the instances from a 1GB subset file. In a first pre-cleaning process, the data is cleaned from features containing more than 50% of NA values and duplicated rows. Further, the numerical variable *year* is transformed into a string for using it as a categorical variable rather than a numerical feature. The exploratory analysis shows that *price* is heavily right skewed, and car-specific features, such as the manufacturer and the car type, tend to heavily affect the dependent variable. A visualisation of geographical information, plotting longitude and latitude against price, does not show a distinct pattern, indicating that the region does not affect a car's price. A correlation analysis, both using the pearson coefficient and scatterplots, shows that especially *horsepower* and *engine_displacement* (car-specific information*)* as well as *mileage* and *owners_count* (car condition) are related to *price*.

Feature selection is performed as an iterative process. Attributes which are highly correlated with the dependent variable are included as well as certain categorical variables due to logical reasoning. This process is iterated multiple times and the selection is altered due to changes in model performance and impurity-based feature importance. Next, the data is split up 80/20 into a train set and a test set using a stratified sampling approach ($p < 30k$, $p < 60k$, $p < 90k$ and $p \geq 90k$) due to the skewness if *price*. After the split, a data pipeline is used for processing the train data to an appropriate input matrix for model training. The pipeline contains two main parts, called wrangling and transformation. The first wrangling part manipulates specific string variables and converts them into numeric. The transformation step deals with imputing the median value for missing numerical variables and scaling them using *sklearn.RobustScaler* (due to outliers in the data set), as well as imputing the most frequent value for missing categorical features and transforming them into binary variables using *sklearn.OneHotEncoder*. The resulting matrix is now used for model training.

Model Training, Parameter Tuning and Model Assessment

Three different supervised regression models are compared: **kNN-Regression**, **Random Forest Regression** and **Ridge Regression**. All have their advantages and disadvantages, which are explained in the Notebook. A grid-search-based parameter optimisation method is used to evaluate the ideal hyperparameters for each model. Once evaluated, all models are then tested using 5-fold cross-validation. Model performance is measured with the root-mean-square-error scores, as this cost function is on the same scale as the dependent variable and therefore easy to interpret. The results show that the optimized Random Forest Regressor offers the best average performance with a mean RMSE of 8683.

Random Forest is an algorithm which uses several decision-trees ("the forest") which are fitted on bootstrapped samples. As an ensemble learning technique, it then averages the results from each tree as its prediction. This method generally performs well and is one of the most-used ML algorithms for solving modern problems. The model assessment for the Random Forest regressor shows that considering a larger number of features in order to make the best split (square root of total parameters) is preferable over a smaller number of features (log of parameters). Further, the average RMSE decreases if the number of splits each decision tree is allowed to make increases, while the maximum number of fitted trees tends to not affect the average score. The assessment also shows that, based on the Gini index, horsepower and mileage are by far the most important features in the model, while car-specific measures generally tend to show a high level of importance.

System Implementation, Conclusion and Discussion

Using the fitted data pipeline and the optimized regressor on the test set results in a RMSE of 7691 and a R^2 value of 0.85. Although the error on the test set is even lower than the average RMSE from the 5-fold cross-validation, a model with this accuracy does not have much of a practical use.

Indicated both by the relatively large standard deviation of cross-validation scores as well as the outlier score in fold 4 (see corresponding plot in notebook), the final model suffers from a large amount of variance. A possible explanation might be the outliers in the data: While over 97% of instances lie in the price range of up to 70k, the data contains certain cars with prices of up to USD 7 Mio. Removing the outliers would not be an adequate approach as they are not based on measurement errors. On the other hand, increasing the amount of data did not fix the problem, the same model fitted on 100MB of data did not result in a significantly lower average accuracy. Possible approaches could be adding more outlier instances (highly expensive cars) to the training data, as the model could learn certain patterns for recognizing such outliers and price them correctly, or trying different models or hyperparameters for trading a decrease in variance against an increase in bias.