
SAPIENZA
UNIVERSITÀ DI ROMA

# Prediction prices of used cars

Giovanni Pica
1816394

December 2021

## Contents

# 1 Introduction

In the developing of this free topic project I developed a *Regression* model for predicting prices of Bmw used cars. Dataset url: https://www.kaggle.com/adityadesai13/used-car-dataset-ford-and-mercedes

## 1.1 Dataset

I take the dataset on Kaggle and it is called "100,000 UK Used Car Data set" and it contains 100,000 scraped used car listings, cleaned and split into car make. This dataset is composed by this features:

| Feature Name | Description |
|---|---|
| model | model name of the bmw |
| year | manufacture year |
| price | price of the car (target) |
| transmission | type of transmision of the car |
| mileage | number of miles that the car have |
| fuelType | type of the fuel |
| tax | is the vehicle tax |
| mgl | miles per gallon |
| engineSize | liters |

## 1.2 Tools Used

I used *Python 3.7* and *Jupyter Notebook* for document plots and some experiments, which contains computer code and text. The library used for all of those experiments is *Scikit-Learn* that is an open source machine learning library for Python and also I use *Numpy* for collect data in arrays. The library used for manipulate and analyze data is *Pandas*. The libraries used for the plots are *Seaborn* and *Matplotlib*.

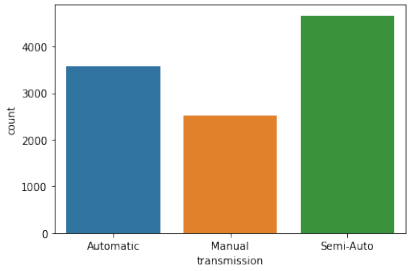
# 2 Load and Clean Dataset

I load the dataset using the Pandas library function *read_csv_file()* and also I check if there are some NaN values, but the dataset is already cleaned.

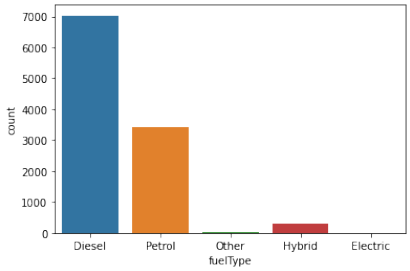
# 3 Data Analysis

I analyze some particular kind of features such as year, transmission and the fuel type and I've decided to plot those analyses. The 1a is the plot of the transmission and Semi Automatic cars are the most common in this dataset. In 1b there is an analysis of the fuel feature and Diesel is the most frequent. In 1c there is a plot of the year feature and the 2019 is the most frequent manufacture year. Another interest plot is the 1d that illustrates the prices target feature

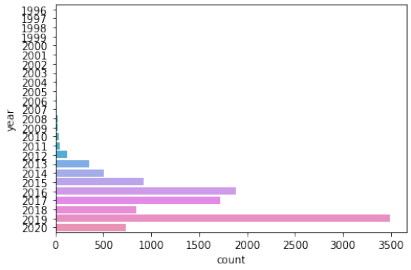
distribution and this feature is skewed on the left so before performing the model I scale this feature. Finally I decide to plot at what prices and at what year cars are selled in 2 and cars are selled in average with an highest price if their manufacture year is 2020.



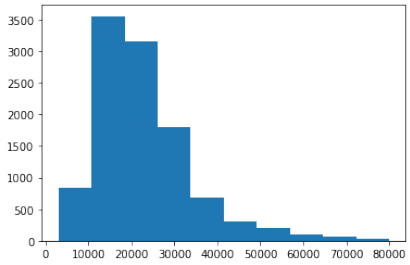

(a) Transmission Histogram




(b) Fuel Histogram




(c) Year Histogram




(d) Prices Distribution

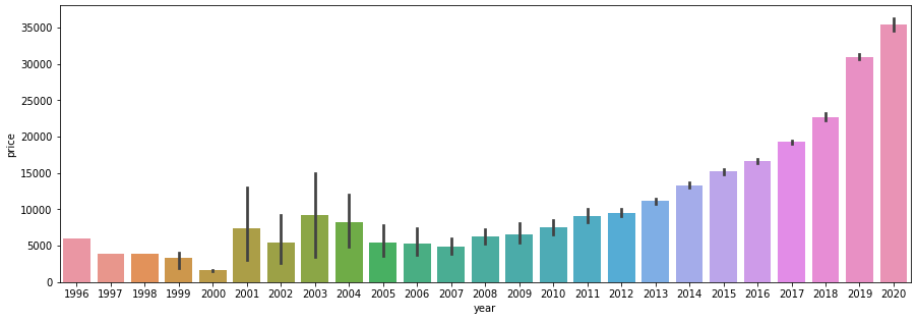



Figure 2: Year and Prices

# 4 Preprocessing

First of all I encode with the function *get_dummies()* of the Pandas library that performs OneHotEncoding on the categorical feature model. After that I scale the data using the library *preprocessing* of Scikit-Learn. After that I

perform *Feature Importance* with a *Random Forest Regressor* and *Permutation Importance* and in this figure 3 illustrates that the feature year is the most important. And also I split the dataset in train and test set and I decide to divide test in 33% and training in 77%.

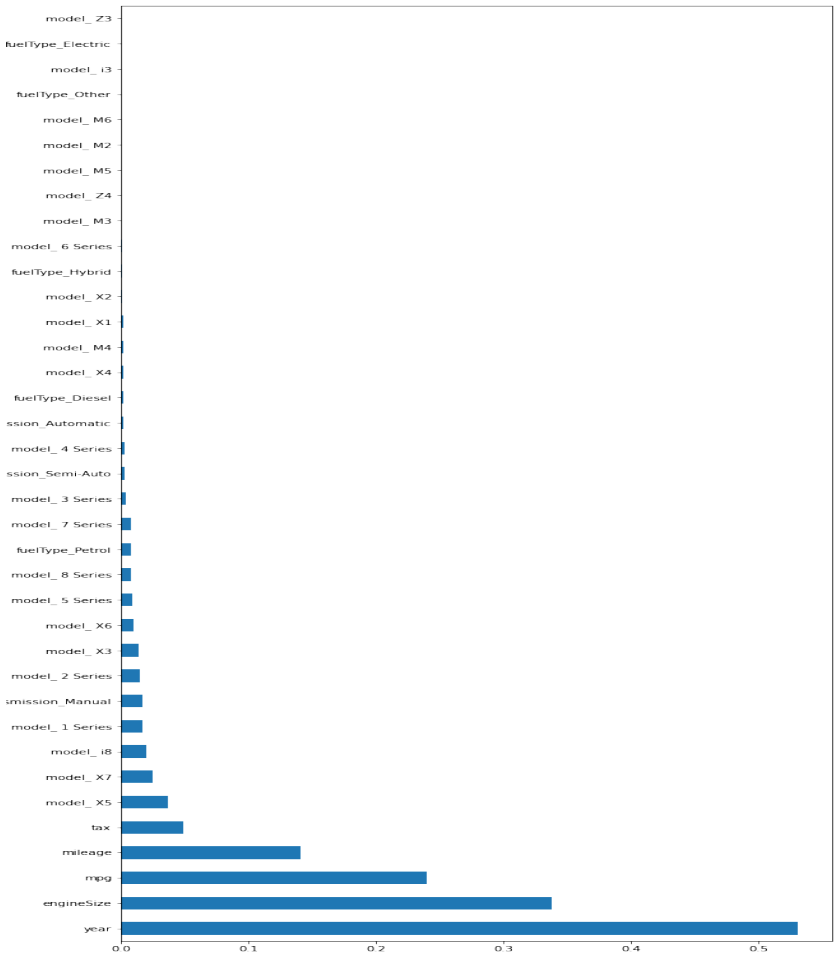



Figure 3: Feature Importance

# 5 Modeling

For the selection of the model I consider 4 options:

1. *Linear Regressor*: fits a linear model with coefficients $w = (w_1, \ldots, w_p)$ to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

2. *Support Vector Regressor*: that is a type of Support Vector Machine that is based on an *hyperplane*. Given labeled training data the algorithm outputs an optimal hyperplane which categorizes new examples.

3. *Decision Tree Regressor*: The model output is a tree structure. Nodes are tests on features values, there is one branch for each value of the feature, and leaf nodes are "decisions", they specify the class label.

4. *Random Forest Regressor*: Ensemble method designed for decision/regression tree classifiers, each tree is generated based on a bootstrap sample of training data and a vector of randomly chosen attributes.

5. *Multi Layer Perceptron Regressor*: NN is a class of ML algorithms belonging to both the categories of supervised and semi-supervised models. The learned model is an algebraic function (or a set of functions), rather than a boolean function, as for DTrees. The learned function is linear for Perceptron algorithm, non-linear for the majority of other NN algorithms. In general, both features and output function(s) are allowed to be real-valued.

## 5.1 Hyperparameter tuning

I decide to use *Cross-Validation* for performs the tuning and I use *GridSearchCV* and the hyperparameters that I tune are:

- For SVR C and $\gamma$;
- For DTR splitter, max depth and min samples leaf;
- For RFR max depth, min samples leaf and number of estimators;
- For MLPR $\alpha$, hidden layer sizes and the solver function.

The score that I use for choose the best hyperparameter is the *MAE*.

## 5.2 Evaluation

For the Evaluation I use as metrics the $R^2$ score which the best possible is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a score of 0.0. So I got this result:

| Model | Score |
|---|---|
| Linear Regression | 0.86 |
| SVR | 0.91 |
| DTR | 0.90 |
| RFR | 0.91 |
| MLPR | 0.92 |

So the best model for this kind of problem is the Multi Layer Perceptron Regressor.