USED CARS PRICE PREDICTION

DATA MINING & MACHINE LEARNING PROJECT DOCUMENTATION

FILIPPO PUCCINI **MATTEO MUGNAI**

https://github.com/mattemugno/used-car-price-prediction-dm-project

A.Y. 2021/2022

SUMMARY

1. INTRODUCTION

Nowadays used cars are very popular since we are living a moment in which buy a new vehicle is not so easy because we are experiencing a difficult historical moment due to Covid-19 pandemic and now also to the war in Ukraine. Car market is highly affected from these factors, mainly for what concerns delivery and shipping times for new cars. In fact, several people complain about this because they are forced to wait a long time between the purchase and the real delivery of the vehicle. Another factor that must be mentioned is the semiconductor crisis that we are experiencing for two years. Naturally, this is highly correlated to the problem mentioned before. Last but not least is the uncertainty that revolves around fuel prices that change day by day and it is not clear which is the more convenient choice to take.

These are the main reasons for which a lot of people are interested in purchasing used cars. In this way they can save money and the vehicle is ready to use if it is in good shape. This is a very popular market and there are plenty of websites and physical dealers for this kind of business.

Often is very difficult to establish the price at which a car has to be sold because we have to consider several factors and the final estimation is very subjective, especially if we give to users the possibility to set their self the price at which they want to sell their own car.

1.1 GOALS

The aim of our study is to build a model able to predict a fair price for selling a used car, through a regression algorithm based on supervised learning.

This project could be used to build a simple user interface in which a user can fill a form with all the features of the vehicle that he wants to put on the market. After receiving the input, the system will return the final price after running a suitable classification regression algorithm for this kind of research.

An application like this is useful for both a seller and a buyer. The first can be sure to put a car on the market with a fair and competitive price, increasing the chances of sold. On the other side, a buyer can verify that he is purchasing a vehicle that worth a certain amount as declared from the dealer, and he can be calm in concluding the negotiation.

1.2 RAW DATASET EXTRACTION

For this project we analyse AutoScout24 website since it is one of the most popular ones. We perform a scraping process in order to obtain an initial raw dataset useful as a starting point for our research. The scraping has been done using Java Jsoup library extracting data by looking at html tags of the page. We focus our study on a subset of car that are published on AutoScout24, so we apply some filter to our search to obtain the URLs that we need for scraping. Selected cars have the following features:

- Year from 1992 to 2021
- Used cars
- All European countries

In addition to this we modify dynamically the URL, using a list of selected car brands in order to scrape an equal number of vehicles for all brands.

These are the brands that we consider:

```
1 usage
public static final String[] BRANDS = { "Audi", "BMW", "Ford", "Mercedes-Benz", "Opel",
        "Renault", "Volkswagen", "Alfa Romeo", "Chevrolet", "Chrysler", "Citroen","Cupra",
        "Dacia", "Daihatsu", "Dodge", "Fiat", "Honda", "Hyundai", "Infiniti", "Jeep",
        "Kia", "Lancia", "Lexus", "Mazda", "MINI", "Mitsubishi",
        "Nissan", "Peugeot", "SEAT", "Skoda", "smart", "Suzuki",
        "Toyota", "Volvo"};
```

Figure 1 Array of brands considered

Moreover, for each brand we performed two scraping process by selecting two different filters:

- 2500 km < Mileage < 50000 km
- Mileage > 50000 km

In this way we obtain cars with heterogenous conditions, so we can train our classifier with records that represent a lot of different situations.

For each car we open the relative page and then we extract all useful attributes.

We decide to extract the following features:

```
public static final List<String> ATTRIBUTES = Arrays.asList(
        "Model", "Price", "Manufacturer", "Country", "Fuel type", "Seller", "Body type", "Upholstery colour",
        "Type", "Doors", "Warranty", "Mileage", "Power", "Gearbox", "Drivetrain", "Other fuel types",
        "Comfort & Convenience", "Fuel consumption", "First registration", "Extras", "Full service history", "Colour",
        "Previous owner", "Safety & Security", "Cylinders", "Entertainment & Media", "Empty weight",
        "Engine size", "CO2 Emissions", "Seats", "Gears", "Emission class", "Upholstery"
        );
```

Figure 2 Array of features considered

For each brand we were able to visit 40 different pages with 20 cars per page.

At the end of this process, we extract a raw CSV file with about 25k rows and 30 attributes that represents the starting point of our analysis.

1.3 FURTHER CONSIDERATIONS

Before starting with pre-processing and classification, we need to think about some aspect of our dataset.

1.3.1 MODEL PRICE PER COUNTRY

First, we need to check if, at the end, we can derive a universal model valid for all countries independently from the provenience of the car. We have to see if the price of a car is correlated with the State in which this one is going to be sold (we need to compare the market trend of different countries).

To understand this important point, we have studied the price of some interesting car models belonging to different nations. We perform an additional scraping by considering only used car with

first registration greater than 2020 (so we are sure that we have only a version of a specific model) and with mileage under 50.000 km.

We consider these European States:

- Italy
- Germany
- France
- Spain

We select this subset of car models:

```
public static final String[] BRANDS = { "Audi", "BMW", "Ford", "Volkswagen",
        "Citroen", "Renault", "Fiat", "Opel", "Toyota", "Volvo"};
```

Figure 3 Brands considered during the analysis

```
public static final String[] MODELS = {
        "A3", "A1", "X1", "X3", "Fiesta", "Focus", "Golf", "Polo",
        "C3", "C4", "Captur", "500", "Corsa", "Yaris", "XC40"
};
```

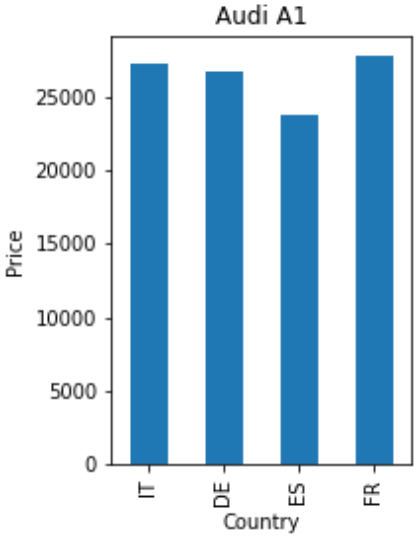
Figure 4 Models considered during the analysis

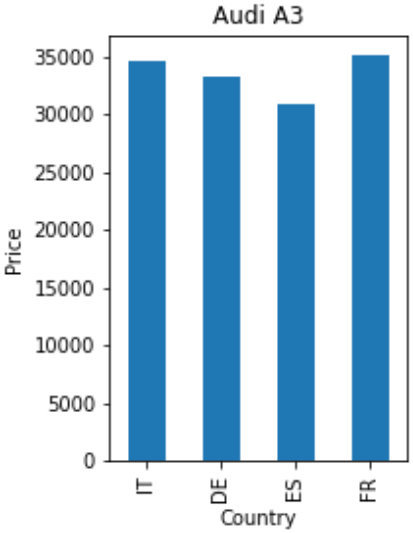
After collecting these data, we load them in some MongoDB collections (one for each country) and we perform the following aggregation:

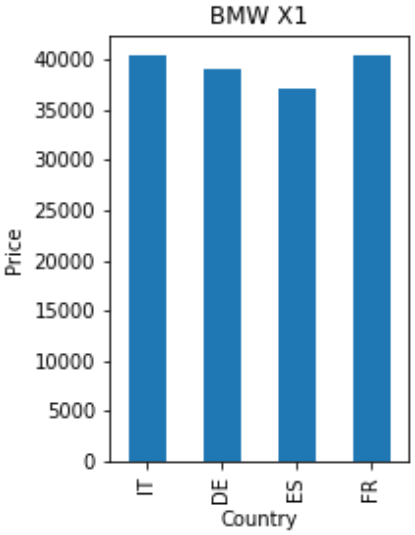
```
db.carsIT.aggregate([
{$group: { _id: {"brand" : "$Manufacturer", "model" : "$Model"}, avgPrice: {$avg: "$Price"}}},
{ $project: {_id : 0, Manufacturer: "$_id.brand", Model: "$_id.model",Price : "$avgPrice"}},
{ $out : {db: "cars", coll:"grouped_carsIT"}}
]);
```

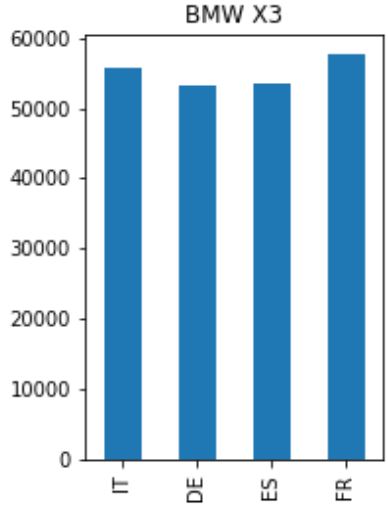
Figure 5 Aggregation exploited with MongoDB Compass

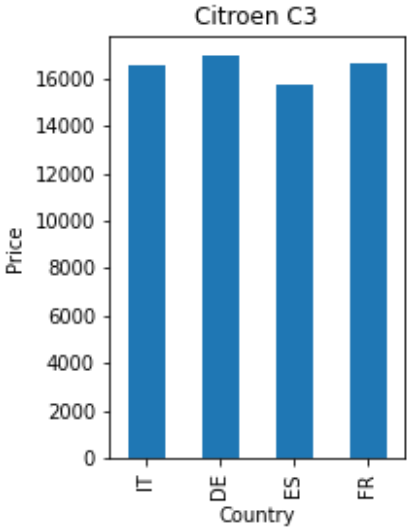
In this way we obtain four dataset (one for each country) with the brand, the model, and the average price of all cars of that specific model in that specific state. After that we can generate a bar chart for each model in Python. Here we have the resulting plots:

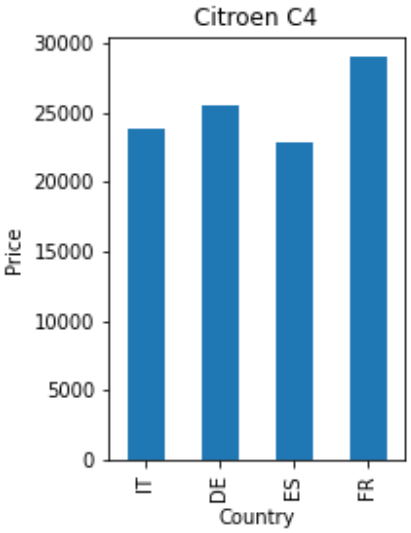

Audi A1
Price
Country
IT
DE
ES
FR
0
5000
10000
15000
20000
25000

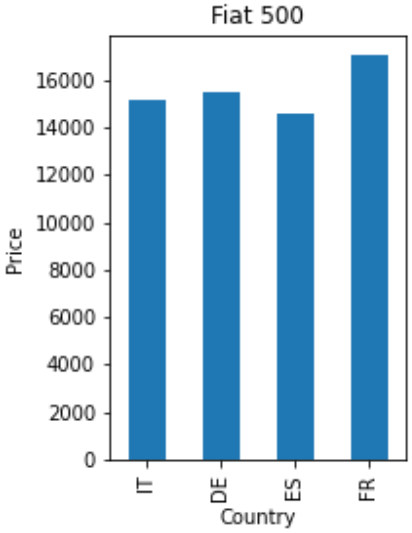

Audi A3
Price
Country
IT
DE
ES
FR
0
5000
10000
15000
20000
25000
30000
35000

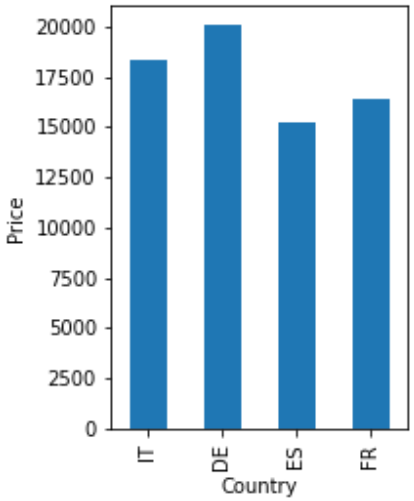

BMW X1
Price
Country
IT
DE
ES
FR
0
5000
10000
15000
20000
25000
30000
35000
40000

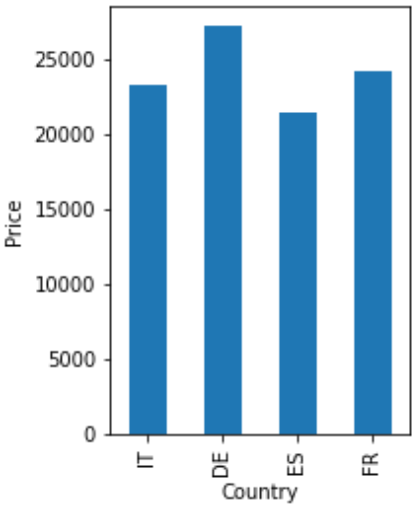

BMW X3
Price
Country
IT
DE
ES
FR
0
10000
20000
30000
40000
50000
60000


Citroen C3
Price
Country
IT
DE
ES
FR
0
2000
4000
6000
8000
10000
12000
14000
16000


Citroen C4
Price
Country
IT
DE
ES
FR
0
5000
10000
15000
20000
25000
30000


Fiat 500
Price
Country
IT
DE
ES
FR
0
2000
4000
6000
8000
10000
12000
14000
16000


Ford Fiesta
Price
Country
IT
DE
ES
FR
0
2500
5000
7500
10000
12500
15000
17500
20000


Ford Focus
Price
Country
IT
DE
ES
FR
0
5000
10000
15000
20000
25000

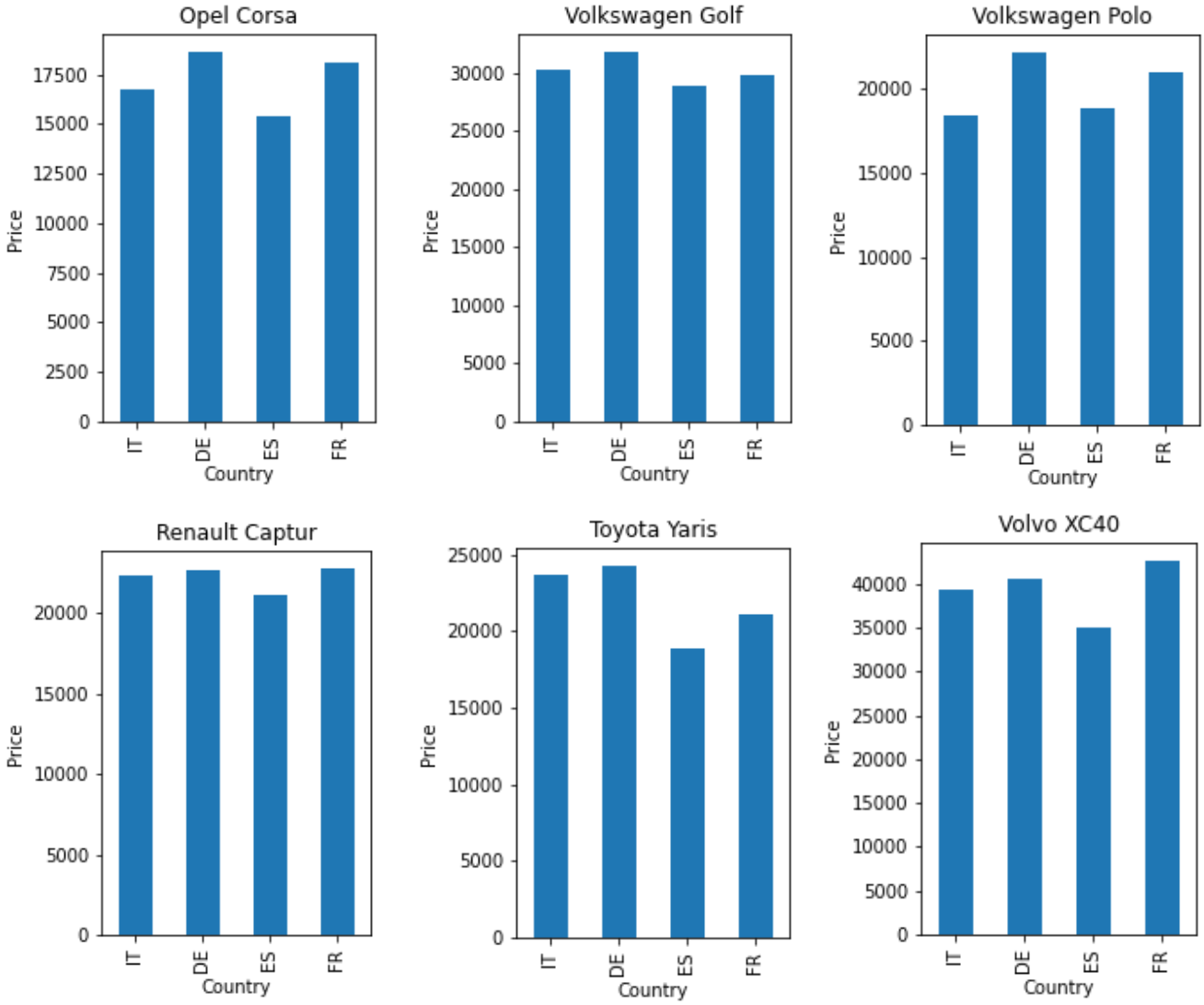


Figure 6 Bar charts

Thanks to these plots we can appreciate the fact that the Spain tend to have lower prices for most of the models took in consideration, so we decide to test the classificator with and without the cars sold in Spain; we will explain this better in the paragraph 3.3.5.

So, we can conclude that used car market is influenced by the location.

1.3.2 TEMPORAL CAR PRICE EVOLUTION

Another main aspect in this kind of study is the temporal factor. It is not difficult to understand that the price of a used car could change in time. We have to check if our prediction model could be valid and extensible to different period.

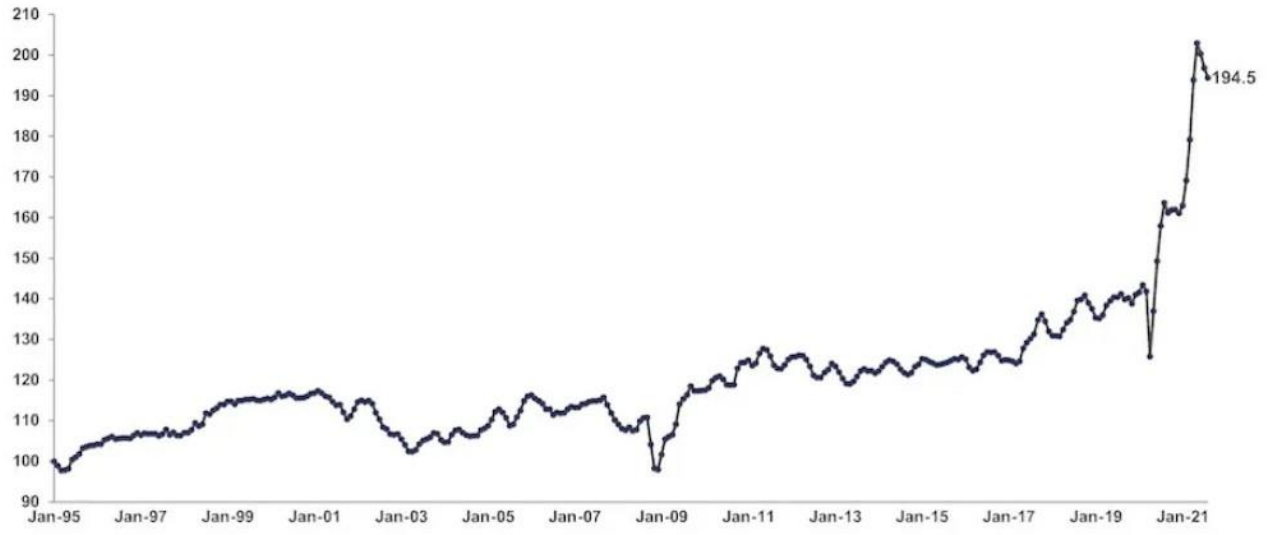


Figure 7 Used vehicle value evolution

According to the Manheim Used Vehicle Value Index (which measures pricing trends in the used car market) we can clearly see that we have several price fluctuations year by year.

Fortunately, we discover that on AutoScout24 a car post last for two months before expiring, so we can conclude that this small period is not influenced by considerable concept drift on price. Hence, if we keep fresh the training data with further scrapings every two months, we can maintain valid and usable the current model.

1.3.3 SPECIAL CAR TYPE INFLUENCE

We can find several different car categories on the market: from the common ones to some special ones (e.g., Supercars or Vintage) that could strongly condition in some way the final price. So, we have to check if our model fit also this kind of vehicles or if it's better to exclude them from our analysis.

For vintage cars we don't have this problem because we have considered only post greater than 1992. In section 3.3.3 we'll do an experiment by taking as test set a subset of vintage features to see how the classification behaves in this case.

Instead, some supercars are already stored in our dataset. As we can see from the previous list of brands that we mentioned we have: Audi, Mercedes, BMW, Nissan, etc, but most of them have been deleted during the outlier's detection phase. We perform, as for the vintage cars, experiments with the classificator with a dataset of supercar, using it as test set, as explained in paragraph 3.3.4.

2 PRE-PROCESSING

2.1 INITIAL DATASET

After performing the scraping on AutoScout24 with the filters and methods described above, we obtained the initial dataset on which we carried out the preprocessing operations; in figure 8 you can see a subset of the features scraped.

BodyType	CO2Emissi...	Colour	ComfortCo...	Country	Cylinders	Doors	Drivetrain	EmissionCl...	EmptyWei...	EngineSize	Entertainm...	Extras	FirstRegist...	FuelConsu...	FuelType	FullService...	Gearbox	Gears
Categorical	Number	Categorical	Text	Categorical	Number	Number	Number	Number	Number	Number	Text	Text	Categorical	Text	Categorical	Categorical	Categorical	Number
Body type	CO2 Emissi...	Colour	Comfort & ...	Country	Cylinders	Doors	Drivetrain	Emission cl...	Empty weig...	Engine size	Entertainm...	Extras	First registr...	Fuel consu...	Fuel type	Full service ...	Gearbox	Gears
Coupe	306 g/km (c...		Air conditio...	DE	10		Rear	Euro 6d-TE...		5204 cc	Bluetooth; ...	Alloy wheel...	10/2020	12.9 l/100 k...	Super Plus ...	Yes	Automatic	
Coupe		Black	Seat heatin...	ES		3				1984 cc			04/2021	7.3 l/100 k...	Gasoline		Automatic	
Coupe		Black	Air conditio...	ES	8		4WD		1760 kg	4163 cc	Bluetooth; ...	Alloy wheel...	02/2015	12.6 l/100 k...	Gasoline		Automatic	7
Coupe		Grey	Air conditio...	BE	6	2	4WD	Euro 6		2995 cc	Apple CarPl...	Ambient lig...	04/2017		Gasoline	Yes	Automatic	8
Coupe	306 g/km (c...	Red	Air conditio...	BE	10		4WD	Euro 6	1655 kg	5204 cc	Bluetooth; ...	Alloy wheel...	05/2018	13.4 l/100 k...	Gasoline	Yes	Automatic	7
Coupe	287 g/km (c...	Black	Air conditio...	DE	10		4WD	Euro 6	1630 kg	5204 cc	Bluetooth; ...	Alloy wheel...	04/2017	12.3 l/100 k...	Super Plus 98	Yes	Automatic	7
Coupe	222 g/km (c...	Black	360Â° cam...	DE	6	2	4WD	Euro 6d	1770 kg	2894 cc	Android Au...	Alloy wheel...	06/2021	8.8 l/100 k...	Super 95	Yes	Automatic	8
Off-Road/P...		White	Air conditio...	ES	4		Front		1460 kg	1495 cc	Bluetooth;		01/2020	6.1 l/100 k...	Gasoline		Manual	6
Coupe	145 g/km (c...	Silver	Air conditio...	DE		2	Front			1984 cc	Bluetooth; ...	Alloy wheel...	04/2021	6.3 l/100 k...	Super 95		Automatic	
Sedan		Grey	Air conditio...	DE			4WD	Euro 6d		3996 cc	Bluetooth; ...	Alloy wheel...	03/2021		Super 95		Automatic	
Coupe	144 g/km (c...	White	Air conditio...	DE			4WD	Euro 6d-TE...		1984 cc	Android Au...	Alloy wheel...	02/2020	6.3 l/100 k...	Gasoline	Yes	Automatic	
Convertible	163 g/km (c...	Yellow	Air suspensi...	DE	4		4WD	Euro 6	1525 kg	1984 cc	Apple CarPl...	Alloy wheel...	08/2017	7.1 l/100 k...	Gasoline	Yes	Automatic	6
Station wag...	288.6 g/km ...	Black	Air conditio...	DE			4WD	Euro 6d-TE...	2208 kg	3996 cc	Bluetooth; ...	Alloy wheel...	04/2020		Super 95		Automatic	
Sedan		Grey	Air conditio...	ES		5							06/2021		Gasoline		Manual	
Station wag...	290.1 g/km ...	Grey	Air conditio...	DE			4WD	Euro 6d-TE...	2233 kg	3996 cc	Bluetooth; ...	Alloy wheel...	03/2021		Super 95	Yes	Automatic	
Sedan	225.1 g/km ...	Blue	Air conditio...	DE			4WD	Euro 6d	1836 kg	2894 cc	Bluetooth; ...	Alloy wheel...	06/2021		Super 95		Automatic	
Coupe	149 g/km (c...	Black	Air conditio...	BE		2		Euro 6		1984 cc	Bluetooth; ...	Alloy wheel...	01/2015	6.4 l/100 k...	Gasoline (P...	Yes	Automatic	
Sedan		Blue	Air conditio...	DE		4	Front	Euro 6d		1498 cc	Bluetooth; ...	Alloy wheel...	02/2021		Super 95		Automatic	
Off-Road/P...	120 g/km (c...	Green	Air conditio...	DE	4				1365 kg	1498 cc	Bluetooth; ...	Alloy wheel...	04/2021	5.3 l/100 k...	Super 95	Yes	Automatic	7
Off-Road/P...	186 g/km (c...	Black	Air conditio...	DE	4	4	4WD	Euro 6	1845 kg	1968 cc	Bluetooth; ...	Alloy wheel...	02/2020	5.6 l/100 k...	Diesel (Part...	Yes	Automatic	7
Coupe	144 g/km (c...	Violet	Air conditio...	DE	4	3	Front	Euro 6d	1370 kg	1984 cc	Bluetooth; ...	Alloy wheel...	02/2021	6.3 l/100 k...	Super 95	Yes	Automatic	7
Compact	137 g/km (c...	Grey	Air conditio...	DE	4	4	Front	Euro 6		1984 cc	Bluetooth; ...	Alloy wheel...	06/2021	6 l/100 km (...	Super 95	Yes	Automatic	
Station wag...	163 g/km (c...	Black	Air conditio...	DE		4	4WD	Euro 6d-TE...		3996 cc	Android Au...	Alloy wheel...	07/2021	11.5 l/100 k...	Gasoline	Yes	Automatic	
Coupe	146 g/km (c...	Yellow	Air conditio...	BE		2		Euro 6		1984 cc	CD player; ...	Alloy wheel...	10/2017	6.3 l/100 k...	Gasoline (P...	Yes	Automatic	
Off-Road/P...	172 g/km (c...	Grey	Air conditio...	DE	4	4	4WD	Euro 6		1984 cc	Bluetooth; ...	Alloy wheel...	01/2020	7.5 l/100 k...	Super 95	Yes	Automatic	

Figure 8 Subset of the starting dataset scraped from AutoScout24

Index	Column	Non-null count	Dtype
0	Body type	25780	object
1	CO2 Emissions	18333	object
2	Colour	24240	object
3	Comfort & Convenience	23776	object
4	Country	25781	object
5	Cylinders	15554	float64
6	Doors	11451	float64
7	Drivetrain	16517	object
8	Emission class	17808	object
9	Empty weight	14612	object
10	Engine size	24708	object
11	Entertainment & Media	22401	object
12	Extras	21928	object
13	First registration	25786	object
14	Fuel consumption	20634	object
15	Fuel type	24489	object
16	Full-service history	11947	object
17	Gearbox	25576	object
18	Gears	16268	float64
19	Manufacturer	25786	object
20	Mileage	25786	object
21	Model	25529	object

22	Other fuel types	2003	object
23	Power	25257	object
24	Previous owner	13309	float64
25	Price	25786	int64
26	Safety & Security	23842	object
27	Seats	23205	float64
28	Seller	0	float64
29	Type	25786	object
30	Upholstery	18498	object
31	Upholstery colour	10748	object
32	Warranty	14420	object

As you can see, the initial dataset is full of missing values, which we have managed in order to obtain a complete dataset that can be used for classification; a more accurate description of the management of missing values will be given in the next paragraph.

We obtained 25786 records with 33 attributes, as you can see in the table above.

2.2 DATA CLEANING

This paragraph will describe in detail the techniques used for the management of missing values, outliers and for the parsing of values that were otherwise unusable.

2.2.1 MANAGE MISSING VALUES AND PARSING OF THE ATTRIBUTES

- **Body type**: we decided to delete the records with Body type equal to NaN (only 6 records),
- **CO2 Emission:** at first, we eliminated the unit of measurement of the value in order to obtain only the numerical value; we had many missing values, so we decided to replace them with average values obtained by taking into consideration the cars of the same model, manufacturer and with the same fuel type (electric cars have zero emissions); if none of these cars is present in the dataset (in the case for example in which we only have one instance of a certain model), we used the average of the emissions of the cars with the same body type.
- **Colour:** we replaced the missing values with the most common colours of the cars of the same model and manufacturer (we exploit a query on the data frame and counted the instances with different colours); in the absence of these we have set the colour to black.
- **Comfort and Convenience, Entertainment and media, Extras, Safety and Security:** these four attributes will be explored in the paragraph on data transformation.
- **Country:** we decided to drop records with this feature equal to null.
- **Cylinders:** we replaced missing values with the more common value among cars of the same model and manufacturer; alternatively, we took the most common value among cars with the same power.
- **Doors:** some cars had the number of doors equal to zero, so we decided to replace it with 5.
- **Drivetrain:** we fill missing values with “Rear” attribute.
- **Emission class:** for the management of the missing values of this attribute, we considered that the emission class is a classification that was introduced for cars registered after 1992, so the

cars belong to a specific class based on their year of registration; therefore, having only cars since 1992, we have classified the cars according to their year of registration.

- **Empty weight:** after removing the unit of measurement of the value, we have replaced the nul l values with the most popular weight of the cars of the same model and manufacturer; alterna tively, we calculated the average of the cars with the same body type.
- **Engine size:** as above, we have removed the unit of measurement and replaced the missing val ues with the most popular engine size among cars with the same model and manufacturer; alt ernatively, we calculated the average engine size of the cars with the same cylinders.
- **First registration:** records with this feature null were dropped.
- **Fuel consumption:** if is a null, we took the most common fuel consumption among cars of the same model, manufacturer and fuel type; alternatively, we calculated the average fuel consum ption among cars with the same body type.
- **Fuel type, other fuel types:** we merged the values of this column and the other fuel type colu mn (electric cars) so as not to have null values.
- **Full-service history, Seller, Type:** these three attributes will be explored in the paragraph on data reduction.
- **Gearbox:** if value is missing, we have set the gearbox to manual.
- **Gears:** if value is missing, we have set the gear to 5 if the gearbox is manual, to 7 if the gearbox is automatic.
- **Manufacturer, Mileage, Price:** no missing values detected.
- **Model:** records with this feature null were dropped.
- **Power:** to replace the missing values we have taken the most common power among the cars of the same model and manufacturer; alternatively, we took the average of the power of cars with the same cylinders.
- **Previous owner:** we replace missing values with zeros.
- **Seats:** we fill NaN values with zeros, then we set those to 5.
- **Upholstery:** to replace the missing values we have taken the most common upholstery type a mong the cars of the same model and manufacturer; alternatively, we set that attribute to “clo th”.
- **Upholstery colour:** if the value is missing, we set that to “black”.
- **Warranty:** if the value is NaN, we set this attribute to zero.

2.2.2 MANAGE OUTLIERS

After managing missing values, we performed an analysis, taking into consideration a subset of attributes, to look for any outliers, i.e., records with feature values very far from the average.

Using boxplots, we found that most outliers match records with values too high or too low on attributes “CO2 Emissions”, “Engine size”, “Empty weight” and “Power”; these outliers found, in most cases, correspond to cars with extreme characteristics that are not in line with the cars in our dataset, such as supercars (for which we will do a more in-depth analysis in the section 3.3.4) or particularly powerful pickups.

To remove outliers, we used the quantile filter, implemented through python, that eliminates the bottom and top 1 % of data. In the end, more than 1000 records were eliminated, which does not correspond to the total number of outliers present: if we had used the IQR method, we would have had many more records flagged as outliers, but we decided to not eliminate them from the dataset;

we also manually eliminated some records with extreme values on other attributes, such as "Mileage," "Gears," or "Seats".

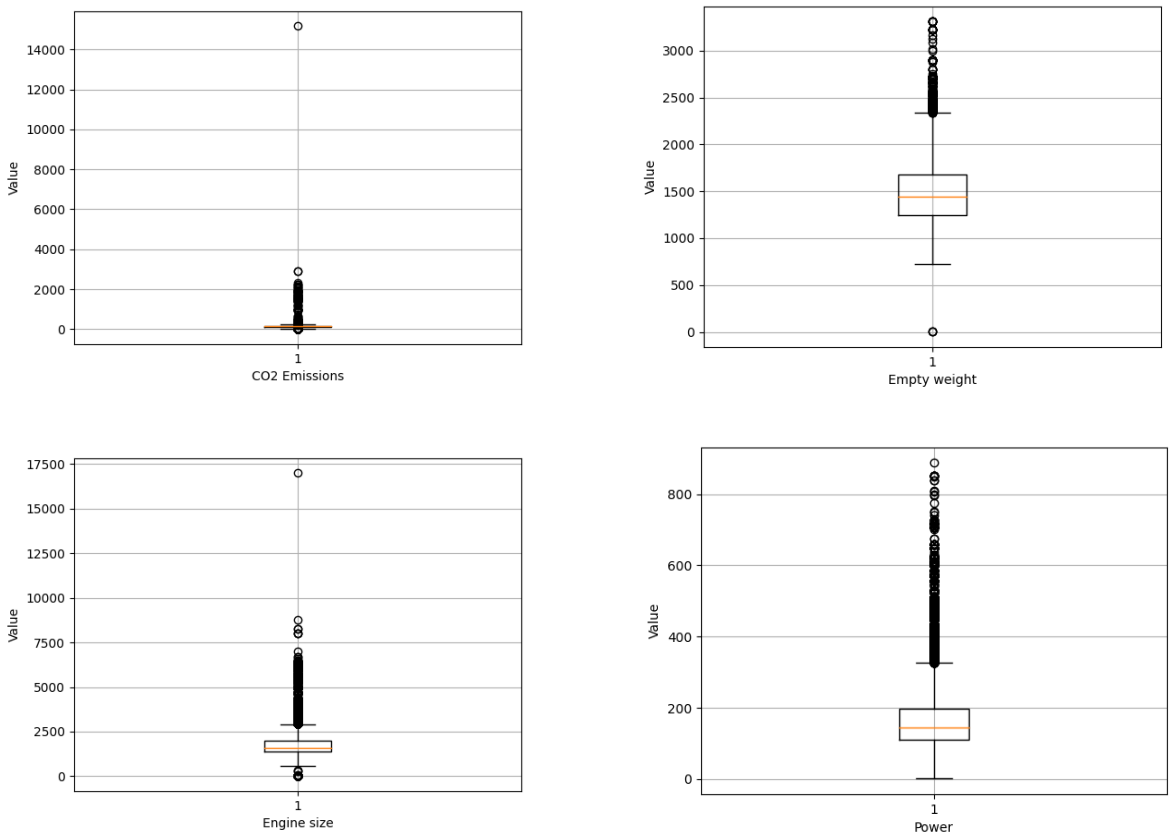


Figure 9 Boxplots of CO2 Emissions, Power, Engine Size and Empty Weight

2.3 DATA REDUCTION

As mentioned above, we have deleted some records because they did not have values for attributes that we considered fundamental, as in the case of the manufacturer, model, mileage and price; in addition, we deleted some records with too many missing values, since the information would not have been reliable. The number of records dropped is very small compared to the total number of records, in the order of a few hundred.

- **Number of records at start**: 25786
- **Number of records after pre-processing**: 23080

We have also eliminated some columns that we did not consider necessary for the purpose of classification, as in the case of the column seller, type (all cars are used), full-service history and other fuel type (merged with the fuel type column).

Index	Column	Non-null count	Dtype
0	Body type	23080	object
1	CO2 Emissions	23080	float64
2	Colour	23080	object
3	Comfort & Convenience	23080	int64

4	Country	23080	object
5	Cylinders	23080	int64
6	Doors	23080	int64
7	4WD	23080	int64
8	Emission class	23080	int64
9	Empty weight	23080	int64
10	Engine size	23080	int64
11	Entertainment & Media	23080	int64
12	Extras	23080	int64
13	First registration	23080	int64
14	Fuel consumption	23080	float64
15	Fuel type	23080	int64
16	Gearbox	23080	int64
17	Gears	23080	int64
18	Manufacturer	23080	object
19	Mileage	23080	int64
20	Model	23080	object
21	Power	23080	int64
22	Previous owner	23080	int64
23	Price	23080	int64
24	Safety & Security	23080	int64
25	Seats	23080	int64
26	Upholstery	23080	int64
27	Upholstery colour	23080	object
28	Warranty	23080	int64

After the data reduction phase, the dataset has a number of attributes equal to 29, without null values, as can be seen in the table above.

2.4 DATA TRANSFORMATION

We performed some data transformation operations, some of which have a great impact on the final state of the dataset.

- **Comfort and Convenience, Entertainment and media, Extras, Safety and Security:** after scraping, for each of these features, we had a list of strings divided by the ";" separator.

We had three ways of handling these attributes: binarizing them all, counting them, or binarizing only part of them and counting the rest. We therefore decided for the last option, since the first would lead us to have too many features (about 250), the second would have flattened the feature to a single numerical value. To select the features to binarize, we took the 5 most popular features for each attribute, which correspond to almost indispensable accessories inside a car; we counted the other ones and set the value in the corresponding attribute column.

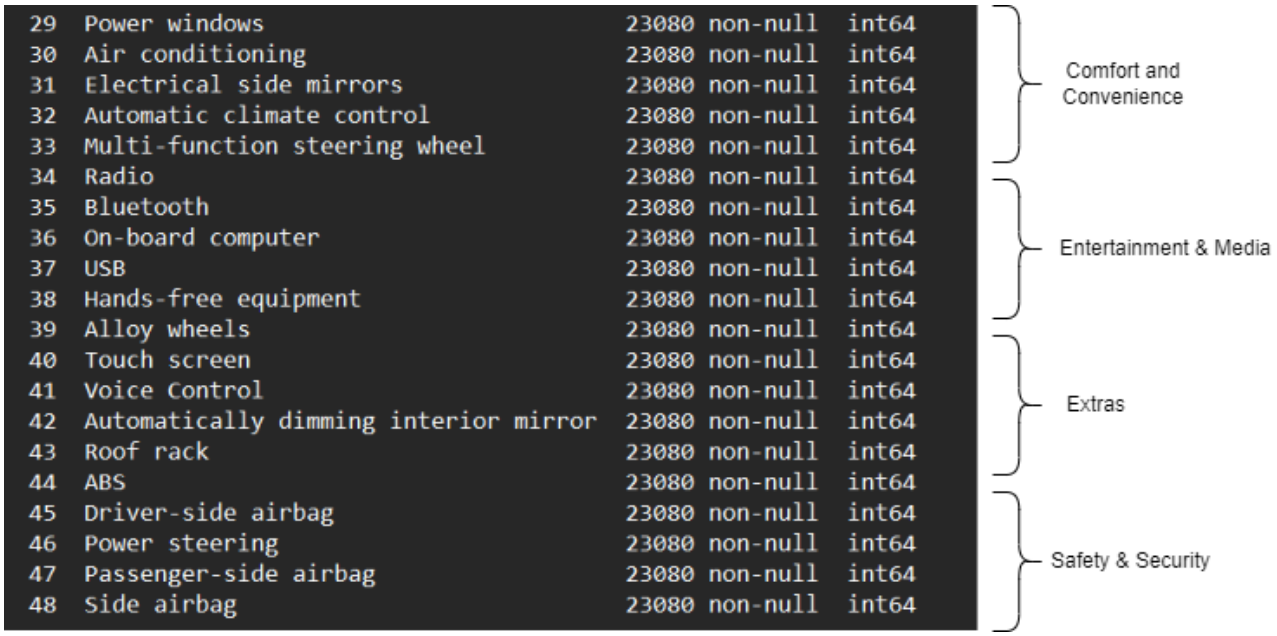


Figure 10 Features binarized

- **Power:** this feature was composed by two values, horsepower and power: we decided to keep only the first one.
- **Fuel type:** we had a lot of different values, so we parse these and traced them back to one of these 6 values: “electric”, “hybrid”, “lpg”, “methane”, “diesel”, “gasoline”; after that we transformed these in numeric values.
- **Emission class:** we had only six values, so we perform a transformation from nominal to numeric.
- **Gearbox, Upholstery:** we have few different values, so we transform these in numeric values.
- **Drivetrain:** we keep only the value “4WD”, binarizing it, because other values not affecting the classification; we rename this attribute in 4WD.
- **Fuel consumption:** initially for this feature we had three values corresponding to the type of use of the car: city, country and combined. We have replaced the values (eliminating the units of measurement) with the average of the previous three and set it to zero in the case of electric cars.
- **Doors:** if value was 2, we set it to 3; if value was 4, we set it to 5.

2.5 DATA INTEGRATION

In addition to the dataset that we created through scraping, we decided to introduce two attributes, the brand score and the reliability score, so that the manufacturer of a car had even more weight in the classification, and the reliability was also considered.

2.5.1 BRAND SCORE

The brand score we utilize is taken from *https://www.carwow.co.uk/blog/most-and-least-reliable-car-brands-revealed#gref* and is based on data taken from the extended car warranty data held by Warranty Wise (UK warranty provider) and the repair claims they have received. Warranty Wise considers a car's age, repair cost, the time taken for it to be repaired, and the frequency of repairs when giving it a reliability score – and this applies to brands, too. The value is included from 0 to 100.

Brand	Reliability Score
Audi	53
BMW	53
Ford	67
Mercedes-Benz	56
Opel	62
Renault	67
Volkswagen	62
Alfa Romeo	53
Chevrolet	53
Chrysler	71
Citroen	62
Cupra	62
Dacia	85
Daihatsu	80
Dodge	70
Fiat	71
Hyundai	71
Honda	87
Infiniti	64
Jeep	71
Kia	69
Lancia	71
Lexus	87
Mazda	67
Mini	60
Mitsubishi	58
Nissan	64
Peugeot	64
Seat	62
Skoda	64
Smart	73
Suzuki	71

Toyota	80
Volvo	62

2.5.2 RELIABILITY SCORE

The reliability score is an attribute introduced by us using the features available in our dataset and the brand score illustrated above.

- At first, we create an array with the brand scores
- Then we transform the warranty values from months to years
- We created an array with the "age" of the cars (current year - first registration)
- We assigned a score from 0 to 10 based on the values on different attributes (mileage, warranty, safety and security), one for each attribute taken in consideration.

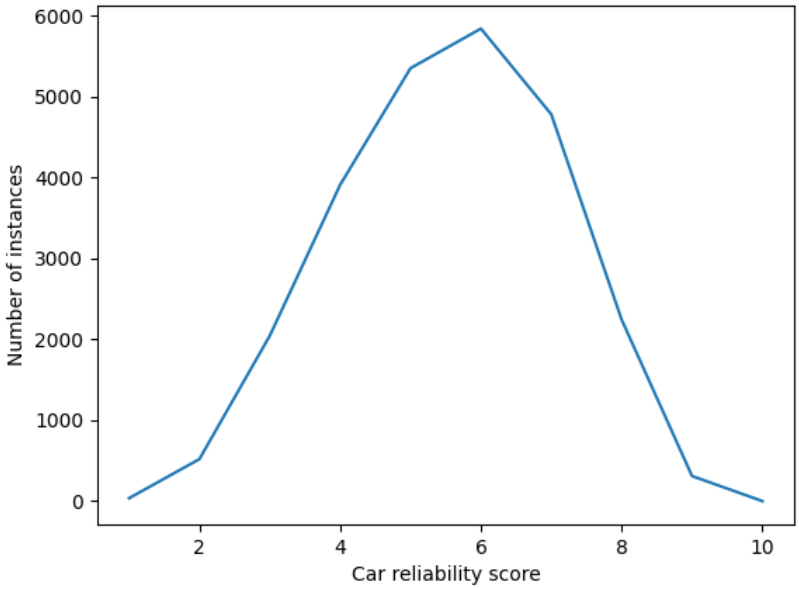


Figure 11 Value distribution for reliability score

- We divided the brand score per 10
- We normalize those attributes from 1 to 10

Finally, we obtain the reliability score:

$$Reliability\ Score = \frac{BrandScore * SafeSecurity * Warranty}{Mileage * PreviousOwner * CarAge}$$

We applied to this feature a logarithmic transformation, because the distribution of values was skewed, and after this, we normalize it in a range from 1 to 10 with min-max normalization.

So, the reliability score of a specific car indicates its present and future reliability. As can be seen in Figure 13 the distribution of values is quite realistic, with most cars scoring between 4 and 8, peaking at 6.

2.5.3 POSSIBLE FUTURE INTEGRATION

Given the large price variations in the car market, as shown in paragraph 1.3.2, in the future it will be necessary to periodically obtain new data, in order to obtain a reliable classification; in our case, we got the data from a source that makes only the most recent data accessible, and we couldn't find a dataset with announces of the past years, so it was not possible to test the classifier with older data.

2.6 FINAL DATASET

The final dataset, including binarized attributes and brand score and reliability score attributes, includes 50 features.

Index	Column	Dtype
0	Body type	object
1	CO2 Emissions	float64
2	Colour	object
3	Comfort & Convenience	int64
4	Country	object
5	Cylinders	int64
6	Doors	int64
7	4WD	int64
8	Emission class	int64
9	Empty weight	int64
10	Engine size	int64
11	Entertainment & Media	int64
12	Extras	int64
13	First registration	int64
14	Fuel consumption	float64
15	Fuel type	int64
16	Gearbox	int64
17	Gears	int64
18	Manufacturer	object
19	Mileage	int64
20	Model	object
21	Power	int64
22	Previous owner	int64
23	Price	int64
24	Safety & Security	int64
25	Seats	int64
26	Upholstery	int64
27	Upholstery colour	object
28	Warranty	int64
29	Power windows	int64
30	Air conditioning	int64
31	Electrical side mirrors	int64
32	Automatic climate control	int64
33	Multi-function steering wheel	int64
34	Radio	int64
35	Bluetooth	int64
36	On-board computer	int64

37	USB	int64
38	Hands-free equipment	int64
39	Alloy wheels	int64
40	Touch screen	int64
41	Voice Control	int64
42	Automatically dimming interior mirror	int64
43	Roof rack	int64
44	ABS	int64
45	Driver-side airbag	int64
46	Power steering	int64
47	Passenger-side airbag	int64
48	Side airbag	int64
49	Car reliability score	int64

At this point we can calculate the correlation matrix to see which features are most related to the price, showed in figure 15. As expected, the features most related to price are power, engine size and cylinders, which indicates that an auto will be priced higher the higher the values of these attributes; we also find that the reliability score attribute, introduced by us as explained in paragraph 2.5.2, is strongly correlated to price. The negative values are those inversely

	Price
Power	0.679775
Empty weight	0.524184
Gears	0.520177
Car reliability score	0.478688
Gearbox	0.455307
First registration	0.449459
Comfort & Convenience	0.423377
Engine size	0.402226
Safety & Security	0.351583
Emission class	0.346662
Upholstery	0.346175
Entertainment & Media	0.339812
4WD	0.338036
Cylinders	0.331068
Extras	0.321377
Voice Control	0.293532
Automatic climate control	0.280594
Hands-free equipment	0.262498
Automatically dimming interior mirror	0.236797
Multi-function steering wheel	0.236592
Bluetooth	0.233093
USB	0.230613
Touch screen	0.195156
Fuel type	0.157799
Roof rack	0.141700
On-board computer	0.107170
Alloy wheels	0.100292
CO2 Emissions	0.089333
Warranty	0.075698
Electrical side mirrors	0.074771
Fuel consumption	0.063368
Radio	0.061440
Side airbag	0.057916
Seats	0.045735
Passenger-side airbag	0.032301
Previous owner	0.014895
ABS	0.003918
Power steering	0.000850
Doors	-0.006415
Power windows	-0.024004
Driver-side airbag	-0.026870
Air conditioning	-0.042171
Mileage	-0.438882

Figure 12 Correlation Matrix

proportional to the price: mileage is the most correlated feature, followed by some additional features binarized during pre-processing.

2.6.1 PRE-PROCESSING IMPLEMENTATION

All the pre-processing phases described in the previous paragraphs were performed using Python, through the Pandas library: once the scraping was done (using the Jsoup library of java), we obtained the csv file, then we created the data frame and performed the operations independently, using Jupyter Notebook; in many scripts we also used the NumPy library, while the graphs were obtained using the pandas plot function.

After all the operations we obtained another csv file, ready to be used on Weka.

3 CLASSIFICATION

After the pre-processing phase, the dataset is ready to be used to learn regression models that will be used to compute the Price feature of the cars. In the following chapter we will discuss all chosen approaches and all experiments done with relative results.

3.1 SELECTED ALGORITHMS

All the classifiers have been evaluated with the same strategy: *10-fold cross validation*. This means that a model is learned from the folds of the training set (90%) and they are evaluated against the corresponding test fold (10%); then the operation is repeated 10 times, and at each iteration a different fold acts as test set.

According to our application domain, the classifiers needed are actually regression algorithms capable of handling our non-nominal class. The selected algorithms have been tested both on the full dimensionality of the dataset (50 features) and on the reduced subspaces identified by the supervised attribute selection methods. Once again, we underline the fact that those features selection algorithms have been modelled exclusively on the bases of the training set. They are *CfsSubsetEval + BestFirst* and *CfsSubsetEval + GreedyStepwise*. We couldn't exploit the *InfoGain* evaluator since it is not capable of working with numerical classes; we discarded *PCA* since it works transforming dimensions, thus it would be difficult to understand if we could have afforded to ask only for a limited number of parameters to input from the user in a hypothetical real application (e.g., if *CfsSubsetEval + BestFirst* selects only 5 attributes, we can create an application that requires only 5 parameters as input. With *PCA* we don't know which original features has been chosen in order to generate the new space, so we are forced to ask the user to input alle the 50 parameters).

To summarize, the tested classifiers are:

- *Linear Regression* with and w/o attribute selection
- *5-NN* with and w/o attribute selection
- *M5Rules* with and w/o attribute selection
- *Random Forest* with w/o attribute selection
- *Rep tree* with w/o attribute selection

We tested the above classifiers in Weka's Classify section by selecting the Attribute Selected Classifier, which allow us to implement the correct approach seen during labs.

3.2 PERFORMANCE EVALUATION

Once the classifiers have been built, as anticipated in the previous paragraph, we tested them using a *10-fold cross validation*. In the following table we report the average values respectively of *correlation coefficient* (CC), *mean absolute error* (MAE), *root mean squared error* (RMSE), *relative absolute error* (RAE) and *root relative squared error* (RRSE) for each tested classifier.

	NO FEATURE SELECTION	**CFS SUBSET EVAL + GREEDY STEPWISE**	CFS SUBSET EVAL + BEST FIRST
LINEAR REGRESSION	OUT OF MEMORY (EVEN WITH 50 % OF SAMPLES)	CC = 0.8524 MAE = 4819.6601 RMSE = 8129.42899 RAE = 43.641 % RRSE = 52.2844 %	CC = 0.8503 MAE = 4854.37 RMSE = 8183.0349 RAE = 43.9553 % RRSE = 52.6291 %
5-NN	CC = 0.844 MAE = 4731.6987 RMSE = 8367.3542 RAE = 42.8446 % RRSE = 53.8146 %	CC = 0.8404 MAE = 4705.6345 RMSE = 8435.5513 RAE = 42.6086 % RRSE = 54.2532 %	CC = 0.8399 MAE = 4700.002 RMSE = 8449.9122 RAE = 42.5576 % RRSE = 54.3456 %
M5-RULES	OUT OF MEMORY (EVEN WITH 50 % OF SAMPLES)	CC = 0.9068 MAE = 3563.7948 RMSE = 6556.9792 RAE = 32.2694 % RRSE = 42.1712 %	CC = 0.9055 MAE = 3594.5742 RMSE = 6600.8324 RAE = 32.5481 % RRSE = 42.4532 %
REP TREE	CC = 0.8902 MAE = 3820.3797 RMSE = 7105.9627 RAE = 34.5928 % RRSE = 45.702 %	CC = 0.8967 MAE = 3702.4068 RMSE = 6907.7698 RAE = 33.5245 % RRSE = 44.4273 %	CC = 0.8942 MAE = 3741.6653 RMSE = 6982.81 RAE = 33.88 % RRSE = 44.9099 %
RANDOM FOREST	DONE WITH 50 % OF SAMPLES --> CC = 0.955 MAE = 2306.9286 RMSE = 4763.2676 RAE = 21.2634 % RRSE = 31.3194 %	**CC = 0.9383** **MAE = 2727.4627** **RMSE = 5456.9642** **RAE = 24.6966 %** **RRSE = 35.0964 %**	CC = 0.9368 MAE = 2759.6865 RMSE = 5518.5499 RAE = 24.9884 % RRSE = 35.4925 %

From these results we can conclude that the best model for this kind of analysis is for sure Random Forest with a slightly better improvement with the adoption of *Greedy Stepwise* as search method in attribute selection.

All the classifiers without attribute selection are, unfortunately, not very suitable for the usage of the application: they would require the user to input 50 parameters, which is a huge amount of work to do. On the contrary, all the attribute selected classifier needs about 20 features.
Here we show an example of selected features during an execution of *CfsSubsetEval + GreedyStepwise*:

Body type, Comfort & Convenience, 4WD, Emission class, Entertainment & Media, First registration, Fuel type, Gearbox, Gears, Manufacturer, Power, Safety & Security, Seats, Upholstery colour, Warranty, Bluetooth, USB, Hands-free equipment, Touch screen, Voice Control, Car reliability score

Linear Regression shows good but not awesome results, it requires few memory for the model loading, and it is very fast at prediction time. However, it showed very long times and high memory consumption at training time.

K-NN is not very suitable because it has the lowest performances, although it requires very little time and memory for the model to be built (and then also to be loaded), the computational effort is concentrated at prediction time (thus increasing too much the application response time).

M5Rules has very good performances and it needs very little time and memory to save and load the model. The prediction is quite fast, but the time required to train the classifier is very long. This algorithm could be chosen as an alternative to *Random Forest*.

REP Tree has also very good performances, but it wastes some memory to save the final decision tree. The response is very fast and also the training time is fine, so it is a good algorithm to be considered as an alternative to *Random Forest*.

Random Forest is the one with the best results in absolute, both with and without attribute selection. The training is very slow, but the prediction time it's quite good, anyway it needs a lot of time and memory for the model to be saved and loaded. Since the fact that there is an increase in response time and the fact that a user is forced to enter all 50 parameters, we discarded the pure *Random Forest,* and we chose *Random Forest* with *CfsSubsetEval + GreedyStepwise* as the default regression model.

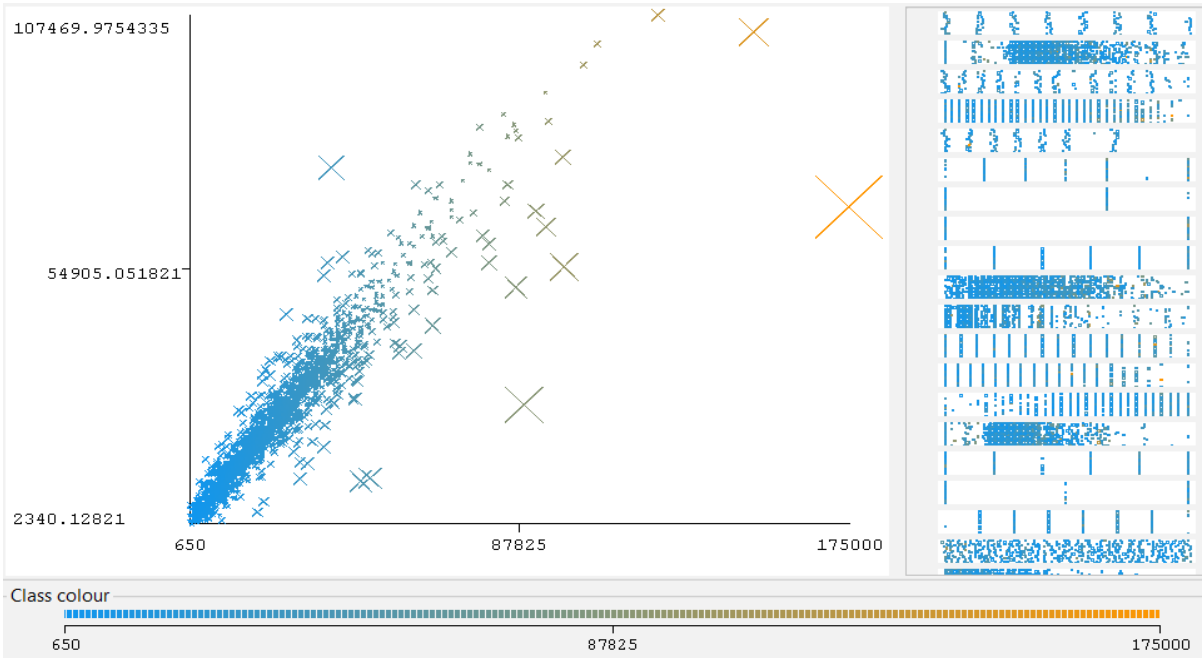


Figure 13 Plot of the classifier errors using Random Forest

To save time and memory a possibility is to decrease the "numIterations" parameter (e.g., from 100 to 10). In this way we get worst but still good results:

- Correlation coefficient = 0.9237
- Mean absolute error = 3016.2248
- Root mean squared error = 5979.3205
- Relative absolute error = 27.3113 %
- Root relative squared error = 38.456 %

3.3 EXPERIMENTS AND RESULTS

Once discovered which is the most suitable classification methods, we can do further analysis with this algorithm to evaluate the goodness and the precision of our model in different cases. We run the following experiments:

- Final dataset (50 features) vs Final dataset - optional binarization (30 features) in 3.3.1
- Final dataset (50 features) vs Final dataset - Car Reliability Score (49 features) in 3.3.2
- Final dataset as Training Set and Vintage Cars dataset as Test Set in 3.3.3
- Final dataset as Training Set and Supercars dataset as Test Set in 3.3.4
- Final dataset vs Final dataset – records with Country =" Spain" in 3.3.5
- Final dataset vs Final dataset with different pre-processing techniques in 3.3.6

3.3.1 CAR OPTIONAL BINARIZATION RESULTS

In this one we try to drop all columns relative to car optional binarization (20) to see whether our classification is worst or better than before. We keep only the total count for each comfort category. We run a 10-fold cross validation with *CfsSubsetEval + GreedyStepwise* as attribute selection and *Random Forest* as classification. Here we show the result of the comparison:

DATASET WITH OPTIONAL BINARIZATION	DATASET WITHOUT OPTIONAL BINARIZATION
Correlation coefficient = 0.9383	Correlation coefficient = 0.9291
Mean absolute error = 2727.4627	Mean absolute error = 3144.1751
Root mean squared error = 5456.9642	Root mean squared error = 5748.0347
Relative absolute error = 24.6966 %	Relative absolute error = 27.3678 %
Root relative squared error = 35.0964 %	Root relative squared error = 39.315 %

From the above table we can appreciate the fact that the dataset with best-5 optional binarization is slightly better than the other case, so from our point of view it has been a good choice to adopt this transformation for our research.

3.3.2 CAR RELIABILITY SCORE RESULTS

In this one we try to drop the car reliability score column to see whether our classification is worst or better than before. We run a *10-fold cross validation* with *CfsSubsetEval + GreedyStepwise* as attribute selection and *Random Forest* as classification. Here we show the result of the comparison:

DATASET WITH CAR REL. SCORE	DATASET WITHOUT CAR REL. SCORE
Correlation coefficient = 0.9383	Correlation coefficient = 0.9353
Mean absolute error = 2727.4627	Mean absolute error = 2827.8752
Root mean squared error = 5456.9642	Root mean squared error = 5582.8502
Relative absolute error = 24.6966 %	Relative absolute error = 25.6058 %
Root relative squared error = 35.0964 %	Root relative squared error = 35.9061 %

From the above table we can appreciate the fact that the dataset with the car reliability score is slightly better than the other case, so from our point of view it has been a good choice to consider this attribute in our research.

3.3.3 VINTAGE CARS RESULTS

In this experiment we apply *CfsSubsetEval + GreedyStepwise* attribute selection to our training set, i.e., our final dataset with 29 attributes (without comfort binarization and car reliability score). Then we run *Random Forest* classification algorithm on Vintage car dataset as test set. The results that we got are the following:

- Correlation coefficient = 0.7266
- Mean absolute error = 9252.6056
- Root mean squared error = 16585.3462
- Relative absolute error = 52.6837 %
- Root relative squared error = 78.7437 %

Hence, we can conclude that our model is not suitable for this kind of cars because we obtained poor performances if we look at the above measures, so, excluding them from our analysis has been a good choice.

3.3.4 SUPERCARS RESULTS

In this experiment we apply *CfsSubsetEval + GreedyStepwise* attribute selection to our training set, i.e., our final dataset with 29 attributes (without comfort binarization and car reliability score). Then we run *Random Forest* classification algorithm on Supercar dataset as test set. The results that we got are the following:

- Correlation coefficient = 0.5411
- Mean absolute error = 97902.0542
- Root mean squared error = 141641.0652
- Relative absolute error = 86.0528 %
- Root relative squared error = 89.5059 %

Hence, we can conclude that our model is not suitable for this kind of cars because we obtained poor performances if we look at the above measures, so, excluding them from our analysis has been a good choice.

3.3.5 INFLUENCE OF SPANISH CARS

In this one we try to drop all records with Country = “Spain” to see whether our classification is worst or better than before. We run a *10-fold cross validation* with *CfsSubsetEval + GreedyStepwise* as attribute selection and *Random Forest* as classification. Here we show the result of the comparison:

DATASET WITH SPANISH RECORDS	DATASET WITHOUT SPANISH RECORDS
Correlation coefficient = 0.9383	Correlation coefficient = 0.9368
Mean absolute error = 2727.4627	Mean absolute error = 2885.7304
Root mean squared error = 5456.9642	Root mean squared error = 5710.5958
Relative absolute error = 24.6966 %	Relative absolute error = 25.3488 %
Root relative squared error = 35.0964 %	Root relative squared error = 35.7756 %

From the above table we can appreciate the fact that the dataset with Spanish cars is quite similar than the other case. We didn't expect this, since in a previous chapter we had seen that Spain tend to have a lower price than other countries. In our dataset we have 4035 records from Spain which is ~17% of the total. A possible explanation of this fact could be that since Spanish cars represent a considerable number of rows, so deleting them could complicate the task of the classificator because it would lose a big amount of training samples.

3.3.6 FURTHER EXPERIMENTS ON PRE-PROCESSING MODIFICATIONS

In this chapter we have analysed the behaviour of our Random Forest model by changing something during pre-processing phase.

For example, we apply nominal to binary transformation to all non-numerical attributes except the model (which would have result in a huge amount of column). So, we applied binarization to Body type, Colour, Country, Manufacturer and Upholstery colour and we obtained a dataset of 114 columns. Then we run regression by using Random Forest and we obtained the following results:

DATASET WITH 50 ATTRIBUTES	DATASET WITH 114 ATTRIBUTES
Correlation coefficient = 0.9383	Correlation coefficient = 0.925
Mean absolute error = 2727.4627	Mean absolute error = 3115.2237
Root mean squared error = 5456.9642	Root mean squared error = 6069.3776
Relative absolute error = 24.6966 %	Relative absolute error = 28.2077 %
Root relative squared error = 35.0964 %	Root relative squared error = 39.0352 %

In this case we got a little worsening, maybe because attribute selection pick more or less the same columns than before but now we have several binary attributes and by chance some of this could have very high correlation without having a strong meaning (e.g., Colour=Yellow, Manufacturer=Mercedes-Benz).

Another test that we consider was the one carried out by applying normalization (between 0 and 1) to all our features, in order to see if performances are affected by this factor or not.

FINAL DATASET	FINAL DATASET WITH 0-1 NORMALIZATION
Correlation coefficient = 0.9383	Correlation coefficient = 0.9079
Mean absolute error = 2727.4627	Mean absolute error = 0.0186
Root mean squared error = 5456.9642	Root mean squared error = 0.0321
Relative absolute error = 24.6966 %	Relative absolute error = 34.2094 %
Root relative squared error = 35.0964 %	Root relative squared error = 41.9479 %

Even in this case the final dataset with standard pre-processing phase is better than the other case.

It is sharper if we use 5-NN because it does a more accurate measurement with linear search as distance method, since all attributes are numeric and in the same range. Here we show results with 5-NN:

FINAL DATASET 5-NN	**FINAL DATASET 5-NN 0-1 NORMALIZATION**
Correlation coefficient = 0.8404	**Correlation coefficient = 0.8699**
Mean absolute error = 4705.6345	**Mean absolute error = 0.0221**
Root mean squared error = 8435.5513	**Root mean squared error = 0.0377**
Relative absolute error = 42.6086 %	**Relative absolute error = 40.6411 %**
Root relative squared error = 54.2532 %	**Root relative squared error = 49.3924 %**

3.4 T-TEST

In order to determine if Random Forest with CfsSubsetEval + GreedyStepwise is really the best solution for this kind of problem, we perform some t-test with respect to different metrics to verify if there are some significant differences between all the algorithms that we have seen before.

	RANDOM FOREST	LINEAR REGRESSION	5-NN	M5 RULES	REP TREE
CORRELATION COEFF.	**0.94**	0.87 *	0.93 *	0.94 *	0.94 *
MAE	**0.37**	0.59 v	0.42 v	0.40 v	0.40 v
RMSE	**0.50**	0.75 v	0.56 v	0.51 v	0.53 v
ELAPSED TIME TRAINING	**9.90**	0.53 *	0.69 *	10.73	1.43 *
ELAPSED TIME TESTING	**0.23**	0.00 *	1.61 v	0.04 *	0.00 *

From this table we can see that *Random Forest* is the best algorithm from the point of view of correlation coefficient, MAE and RMSE but is significantly slower than other algorithms took in consideration. So, we can conclude that *REP tree* could be a valid alternative because it has similar (even if slightly worst) performances, but it is faster than Random Forest in time and test training. These could be important metrics in a real-life application from the point of view of time response for the user.

4 CONCLUSIONS

From this project we have understood that is possible to build an accurate model with the aim of determining the price of a used car. This task could be very useful in a real context because it can help a user to decide at what price to sell his or her car, or whether a car is being sold at a fair price.

As we have seen, it is not possible to classify every type of car using the same classifier: much more data on cars with particular characteristics would be needed for an even more comprehensive work, so as to create an even more reliable and universal model.

For what concern the maintenance of our model, we have to keep the data up to date in order to obtain reliable results, that reflects the car market price variation.

One possible idea for improving the model would be the introduction of a car's accident history, so that a buyer could have a detailed account of the car they would like to buy.