Data Mining Project - CARVANA DATASET - Report

Thizirie Ould Amer, Alessandro Bonini, Francesco Salerno, and Malick Jobe

University of Pisa, Tuscany, Italy

Abstract. This project tries to highlight the important criteria from the Carvana dataset, that will help predicting if a car deal is good or not and notice some behaviours that can help in this process. We tried different methods to enhance the distinctions of the recorded observations that will lead to good predictions and groupings. Following guidelines of our data mining course, we will apply and try to improve the results we get from several studied algorithms.
In the first part of this report, the raw data properties will be discussed and analyzed. Then we will show how we pre-processed it and how we carried out the analysis and transformations of variables. Thus, feature engineering process will be described in this part, as well as the outliers detection and correction and data visualization.
The next parts describe the work we have completed in clustering. K-means, Hierarchical Cluster Analysis (HCA) were performed to try getting a correct and smooth separation of the observations we have. DBSCAN has also been used in the same purpose, to try a density-based method and compare the results.
The penultimate step is not the least important of course, since we will try different **classification** (the target is a binary variable) algorithms and compare them to select the most efficient one for the classification. The nature of the dataset makes it hard since we have to deal with an imbalance situation here.
Finally, we will conclude and analyse our results and describe sum up the results and our learning outcomes from this project.

Keywords: Data Mining · Feature Engineering · Supervised Learning · Classification · Pattern Mining · Clustering · Carvana Data Set · ML

1 Data Understanding

The Carvana dataset is composed by two different sets. One is the training set, *training.csv*, which contains 58386 records and 34 features (including the target, ***IsBadBuy***). This will be the dataset which we will work on in all the project except Classification which will need and use the second one.
Our second set, *test.csv*, is composed by 14597 observations and 34 columns (including the target variable). The test set will be used to evaluate the classification models before a final one that is most likely going to perform better than the others on a blind test.
In both sets, we have 13 categorical variables and 21 numerical variables.

1.1 Description of the variables

Table 1: Information about variables, their types and if they have been used or not in the following analysis.

Variable	Type (Nb classes or mean)	Description	Used
RefID	Numerical (ID)	Unique (sequential) number assigned to vehicles	No (ID)
PurchDate	Date	The Date the vehicle was Purchased at Auction	No (VehicleAge corr.)
Auction	Categorical	Auction provider at which the vehicle was purchased	Yes
VehYear	Numerical	The manufacturer's year of the vehicle	No (VehicleAge corr.)
VehicleAge	Numerical	Years elapsed since the manufacturer's year	Yes
Make	Categorical	Vehicle Manufacturer	Yes
Model	Categorical	Vehicle Model	No (Too many)
Trim	Categorical	Vehicle Trim Level	Yes
SubModel	Categorical	Vehicle Submodel	No (Too many).
Color	Categorical	Vehicle Color	No

Transmission	Binary	Vehicles transmission type (Automatic, Manual)	Yes
WheelTypeID	Numerical (ID)	The type id of the vehicle wheel	No
WheelType	Categorical	The vehicle wheel type description (Alloy, Covers)	Yes
VehOdo	Numerical	The vehicles odometer reading	Yes
Nationality	Categorical	The Manufacturer's country	Yes
Size	Categorical	The size category of the vehicle.	Yes
TopThree AmericanName	Categorical	Identifies if the manufacturer is one of the top three American manufacturers	Yes
MMRAcquisition AuctionAveragePrice	Numerical	Acquisition price for this vehicle in average condition at time of purchase	Yes
MMRAcquisition AuctionCleanPrice	Numerical	Acquisition price for this vehicle in the above Average condition at time of purchase	Yes
MMRAcquisition RetailAveragePrice	Numerical	Acquisition price for this vehicle in the retail market in average condition at time of purchase	Yes
MMRAcquisiton RetailCleanPrice	Numerical	Acquisition price for this vehicle in the retail market in above average condition at time of purchase	Yes
MMRCurrent AuctionAveragePrice	Numerical	Acquisition price for this vehicle in average condition as of current day	Yes
MMRCurrent AuctionCleanPrice	Numerical	Acquisition price for this vehicle in the above condition as of current day	Yes
MMRCurrent RetailAveragePrice	Numerical	Acquisition price for this vehicle in the retail market in average condition as of current day	Yes
MMRCurrent RetailCleanPrice	Numerical	Acquisition price for this vehicle in the retail market in above average condition as of current day	Yes
PRIMEUNIT	Categorical	Identifies if the vehicle would have a higher demand than a standard purchase	No
AUCGUART	Categorical	The level guarantee provided by auction for the vehicle.	No
BYRNO	Numerical (ID)	Unique number assigned to the buyer that purchased the vehicle	No
VNZIP	Categorical	Zipcode where the car was purchased	Yes
VNST	Categorical	State where the the car was purchased	No
VehBCost	Numerical	Acquisition cost paid for the vehicle at time of purchase. Numerical	Yes
IsOnlineSale	Binary	Identifies if the vehicle was originally purchased online	Yes
WarrantyCost	Numerical	Warranty price (term=36month and millage=36K)	Yes
IsBadBuy	Binary	Identifies if the kicked vehicle was an avoidable purchase	Yes (Target)

1.2 Data visualisation and understanding

In our initial data visualisation we used bar chart and histograms. We tested different attribute parameters to understand the distribution of our variables. We considered here all of the original attributes. We will then delete the three ID variables (*BYRNO, WheelTypeID, RefID*) and the two variables with too many missing values (*PRIMEUNIT, AUCGUART*).

Assessing data quality Looking at the training dataset we have found the following attributes containing missing values:

Table 2: Features with missing values

Feature name	**Number of Missing values**	**Corrected by**
TRIM	1911	Mode (BAS)
SubModel	7	Mode (4D SEDAN)
Color	7	Mode (SILVER)
Trasmission	8	Mode (AUTO)
WheelTypeID	2573	Deleted variable
WheelType	2577	Mode (ALLOY)
Nationality	4	Mode (AMERICAN)
Size	4	Mode (MEDIUM)
TopThreeAmericanName	4	Mode (GM)
MMRAcquisitionAuctionAveragePrice	13	Median (6128.12)
MMRAcquisitionAuctionCleanPrice	13	Median (7372.91)
MMRAcquisitionRetailAveragePrice	13	Median (8497.28)
MMRAcquisitonRetailCleanPrice	13	Median (9851.76)
MMRCurrentAuctionAveragePrice	245	Median (6131.66)
MMRCurrentAuctionCleanPrice	245	Median (7389.95)
MMRCurrentRetailAveragePrice	245	Median (8776.06)
MMRCurrentRetailCleanPrice	245	Median (10145.22)
PRIMEUNIT	55703	Deleted variable
AUCGUART	55703	Deleted variable

The other variables do not have any missing values.

We need to specify that when we will present **pattern mining**, we replaced the missing values in a proportional manner with respect to the distribution of the variables, for *Trim* and *WheelType*. Regarding *Color* and *Size*, we decided to drop their missing values (in pattern mining).

Correlations The most important correlations in the presented figure:

As we can see, 8 major variables *MMR[]* are strongly correlated with one another (Between 0.85 and 0.99). We will have to take this into account in the feature engineering to avoid redundancy and to make the analysis easier and limit the number of features in the next steps.

Moreover, from this matrix and others we have done (not in the report, due to space limitation) a high correlation appears between *VehBcost* and all the *MMR* factor. In particular, *VehBcost* and the *MMR* variables have a correlations around 0.8 with each one. Finally, we notice a slight negative linear correlation between *VehAge* and *VehCost*. We noticed also strongly negative correlation between *VehAge* and *VehYear* which obviously expected.

Feature Engineering:

Within the group of numerical value there are *VehYear* and *VehAge* which have the same meaning behind (they both refer to the manufacturer's year of the vehicle). We will keep *VehAge* since it is more natural and easier to analyse.

The *MMR[...]* are all strongly correlated. This is due to some measures that are either done in a very short time space, or values that rarely have surprising slumps or drastic increase.

Regarding to categorical variables, the ones which are related are mostly *Make*, *Model*, *SubModel* and *TopThreeAmerican*, because they all refer to the manufacturers of the vehicle. We will then work with some and not all of them. (The ones with a more limited number of classes).

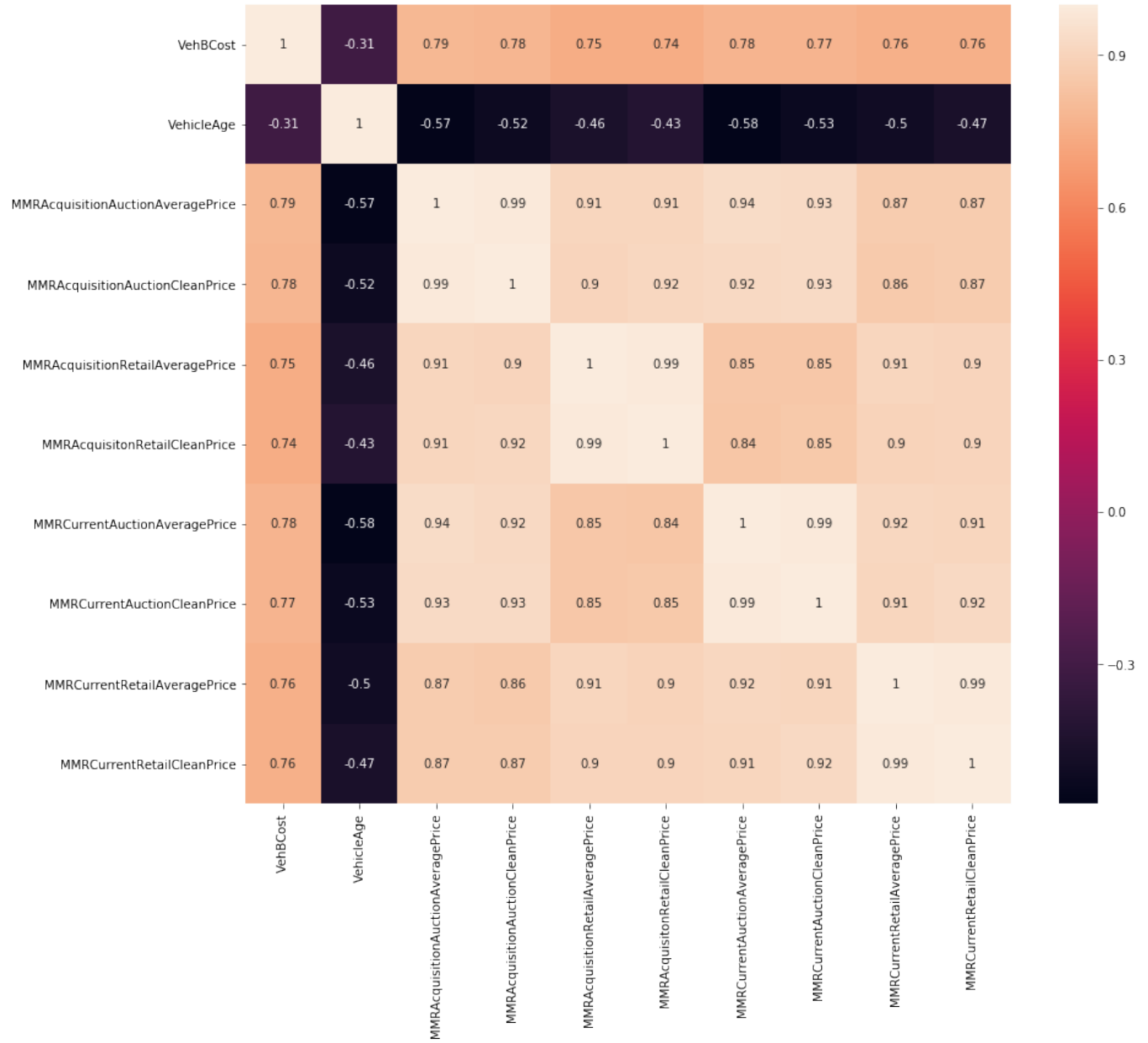

Fig. 1: Correlation matrix of few variables.

At this stage, we have decided which variables we will combine to create a new variable representative. The most important one is the combination of: *MMRAcquisitionAuctionAveragePrice*, *MMRAcquisitionAuctionCleanPrice*, *MMRAcquisitionRetailAveragePrice*, *MMRAcquisitonRetailCleanPrice*, *MMRCurrentAuctionAveragePrice*, *MMRCurrentAuctionCleanPrice*, *MMRCurrentRetailAveragePrice*, *MMRCurrentRetailCleanPrice* into one called ***MMR_factor***.

Since we know that they are all closely related and correlated therefore we choose only four of them because they give out the same results and find the mean: *MMRAcquisitionAuctionAveragePrice*, *MMRAcquisitonRetailCleanPrice*, *MMRCurrentAuctionCleanPrice*, *MMRCurrentRetailAveragePrice*.

Normalizations Another variables' transformation that we have done is regarding normalization of variables to have a better control of the numbers and their evolution, thus a better vision of correlations and the distribution of the variables. The used scalers were *MinMaxScaler* and *StandardScaler*. Some of the scaled variables were *MMR_factor* and *VehBCost*.

Fig. 2: New correlation matrix after normalisation. A correlation to highlight here is the one between *VehBCost* and *VehOdo* that is equal to 0,41, that appeared only after the normalisation has been done.

Detection of outliers:

We are going to present only the detection of outliers for some features. A similar process has been applied for the other ones.

The results displayed have been obtained after data cleaning and normalisation. The method to evaluate the presence or absence of outliers is **IQR** scores. See Figures 3 and 4.

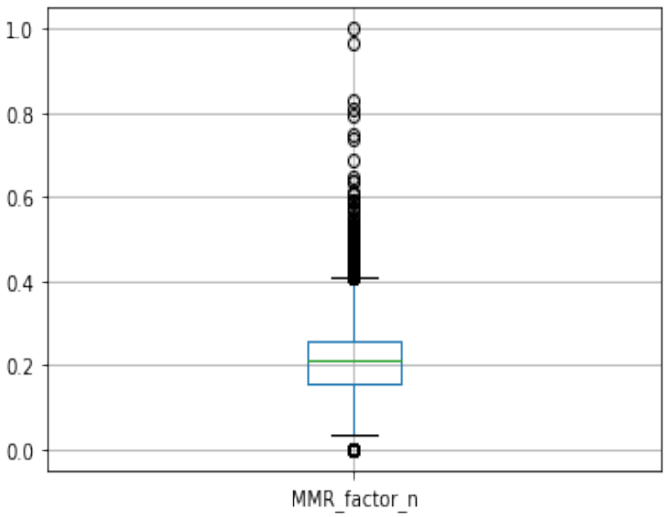

Fig. 3: Boxplot for outliers detection in the *MMR_factor* values. – Superior limit = 0.410827, up-outliers = 406. – Inferior limit = 0.0059550, down-outliers = 355.

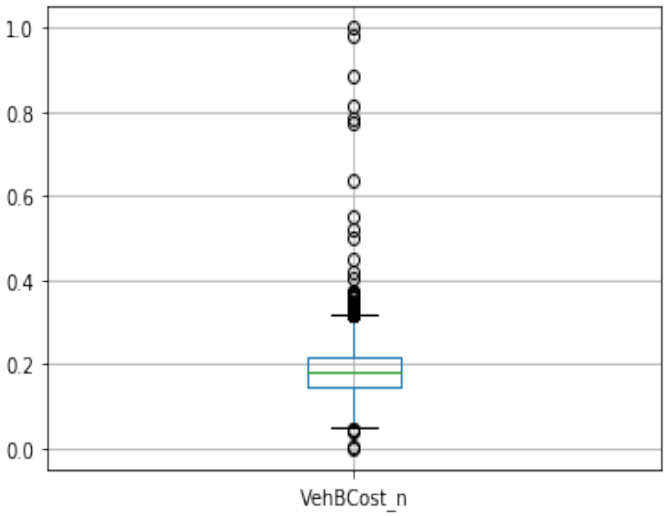

Fig. 4: Boxplot for outliers detection in the *VehBCost* values. – Superior limit = 0,3180575 outliers = 141. – Inferior limit = 0,04725349, outliers = 5

2 Clustering

Cluster analysis requires to build a dataset which has a particular composition. Due to this, we selected only few of the all variables as described above. We selected a dataset composed by *VehicleAge*, *WarrantyCost*, *VehBCost* and *Vehodo* because they are the numerical variables that fit better in clustering. Another variable that we could have taken into account is *MMR_factor* but its redundant with *VehBCost* (they have 0.8 correlation) and *VehBCost* gives us better results in clustering than *MMR_factor*, due to this we removed this last one.

2.1 K-means

In our analysis the best attributes for K-means were: *VehicleAge*, *WarrantyCost* and *VehBCost*. Using these attributes we obtained the best combination of **SSE** and **silhouette** score.

The distance function that we preferred is the **euclidean distance** and we normalize our dataframe with **MinMaxScaler** because it improves the results.

When we evaluated the different values of K, we understood that the ideal number was between 3 and 6. The final choice is subjective as there is a trade-off between *silhouette* and *SSE*. We set the number of time the k-means algorithm will be run with different centroids = 10 and a maximum number of iterations of the k-means algorithm for a single run = 300. The clusters we obtained were not easy to visualize because of the number of dimensions, so we preferred displaying the results and values instead of the plots in this part.

Table 3: K-means Algorithm results (Silhouette and SSE) and composition of clusters.

K	SSE	Silhouette	Composition of clusters
2	1228.8	0.46	0: 22739, 1: 35647
3	833.8	0.39	0: 21983, 1: 23862, 2: 12541
4	692.5	0.37	0: 26386, 1: 15933, 2: 9281, 3: 6786
5	601.5	0.34	0: 15998, 1: 16926, 2: 6575, 3: 9620, 4: 9267
6	517.7	0.34	0: 5856, 1: 9280, 2: 13626, 3: 12701, 4: 16391, 5: 532
7	487.5	0.33	0: 8172, 1: 16391, 2: 5856, 3: 8521, 4: 12192, 5: 6725, 6: 529

We selected K = 4 for our analysis, since after K = 4 the value of SSE starts decreasing with a slower pace while the silhouette still has a good result and decreases with an even more slow rhythm.

We then end up with four clusters; **Cluster 0** contains 26386 records. **Cluster 1**: 15933 records. **Cluster 2**: 9281 records and **Cluster 3** is composed by 6786 records. We can see the comparison of the distribution of the variables within the clusters dataset, in the four following figures (whearas in the whole dataset we have 58386 records for each

attribute, the typical values for VehicleAge are between 3 and 5 with a standard deviation of 1.71; regarding VehBCost, the values are typically included between 5430 and 7900 with a standard deviation of 1762.07; and WarrantyCost is usually above 837 and below 1632, with a standard deviation of 598.88):

	VehicleAge	VehBCost	WarrantyCost
count	26386.000000	26386.000000	26386.000000
mean	3.518608	6847.712488	1205.518267
std	0.499663	1580.656913	498.024280
min	3.000000	225.000000	462.000000
25%	3.000000	5690.000000	825.000000
50%	4.000000	6795.000000	1119.500000
75%	4.000000	7850.000000	1503.000000
max	4.000000	20100.000000	5908.000000

Fig. 5: In cluster 0 we have 26386 records for each attribute, the typical values for VehicleAge are between 3 and 4 with a standard deviation of 0.49; regarding VehBCost between 5690 and 7850 with a standard deviation of 1580.65; regarding WarrantyCost between 825 and 1503 with a standard deviation of 498.02.

	VehicleAge	VehBCost	WarrantyCost
count	15933.000000	15933.000000	15933.000000
mean	5.358815	6546.500932	1236.953493
std	0.479668	1949.261888	473.187852
min	5.000000	1720.000000	462.000000
25%	5.000000	4995.000000	869.000000
50%	5.000000	6230.000000	1215.000000
75%	6.000000	7925.000000	1506.000000
max	6.000000	13535.000000	4032.000000

Fig. 6: In cluster 1 we have 15933 records for each attribute, the typical values for VehicleAge are between 5 and 6 with a standard deviation of 0.47; regarding VehBCost between 4995 and 7925 with a standard deviation of 1949.26; regarding WarrantyCost between 869 and 1506 with a standard deviation of 473.18.

	VehicleAge	VehBCost	WarrantyCost
count	9281.000000	9281.000000	9281.000000
mean	1.729555	7606.076400	1115.775886
std	0.444699	1325.320617	492.272561
min	0.000000	3465.000000	462.000000
25%	1.000000	6815.000000	754.000000
50%	2.000000	7550.000000	920.000000
75%	2.000000	8280.000000	1389.000000
max	3.000000	36485.000000	4032.000000

Fig. 7: In cluster 2 we have 9281 records for each attribute, the typical values for VehicleAge are between 1 and 2 with a standard deviation of 0.44; regarding VehBCost between 6815 and 8280 with a standard deviation of 1325.32; regarding WarrantyCost between 754 and 1389 with a standard deviation of 492.27.

	VehicleAge	VehBCost	WarrantyCost
count	6786.000000	6786.000000	6786.000000
mean	7.291925	5505.029546	1861.769525
std	0.793739	1730.307818	929.191936
min	5.000000	1.000000	762.000000
25%	7.000000	4175.000000	1275.000000
50%	7.000000	5152.500000	1641.000000
75%	8.000000	6535.000000	2218.000000
max	9.000000	11955.000000	7498.000000

Fig. 8: In cluster 3 we have 6786 records for each attribute, the typical values for VehicleAge are between 7 and 8 with a standard deviation of 0.79; regarding VehBCost between 4175 and 6535 with a standard deviation of 1730.30; regarding WarrantyCost between 1275 and 2218 with a standard deviation of 929.19.

As we can see from the the figures above, **Cluster 2** represents better the distribution of variables according to the attributes *VehicleAge* and *VehBCost* while **Cluster 1** represents better the distribution of variables according to *WarrantyCost*. The whole dataset have a better standard deviation only respect to Cluster 1 in *VehBCost* and compared to Cluster 3 in *WarrantyCost*.

2.2 DBSCAN

After trying many different combinations of variables we concluded that the best group was: *VehOdo*, *WarrantyCost* and *VehBCost* because they adapt better in our different clustering algorithms and we obtain the best silhouette and SSE scores, especially when we apply Density based Clustering (DBSCAN) algorithm.

After trial and in order to improve our performance, we kept the **Euclidian Distance** as our distance function and we normalized our dataframe with *StandardScaler*.

The best silhouette score we recorded was a result of an analysis with the following parameters' values: $eps = 0.9$ and $min\_samples = 5$. The value of the *silhouette score* is equal to **0.77**.

We obtained two clusters -1 which contains 24 elements and 0 which contains 58362 elements. It is thus, not a good clustering when we check the number of elements in each one.

Other useful parameters were: $eps = \mathbf{0.8}$ and $min\_samples = \mathbf{9}$ that gives us a silhouette of 0.74. The clusters obtained are -1 which contains 33 elements and 0 which contains 58353 elements. Again, the results were not satisfying, and any trade-off gave us very low score results.

Fig. 9: DBScan results. Parameters: Eps=0.9 ; min_samples= 7. Attributes used [VehBcost - WarrantyCost]

We went further to increase the number of elements in the cluster -1 since we interpret this one as outliers points and we want to remove more outliers than before. So the parameters that we try are with *eps* = 0.8 or 0.9 and *min_samples* in a range of 150-250, in this way we obtained a *silhouette score* of **0.63**.

Fig. 10: Parameters: Eps=0.8 ; min_samples= 150. Attributes used [VehBcost - WarrantyCost]

In a DBSCAN analysis with parameters *eps* = 0.8 *min_samples* = 200, we obtain a silhouette score of **0.63** and two clusters -1: 597 and 0: 57789. Interpreting these data like cluster -1 is the cluster with outliers, we compared the distribution of variables between the cluster 0 and the whole dataset. We decide to not use the clusters obtained with the best silhouette because if not we have too few difference between the description of the distribution of variables. (See Fig. 11 and 12)

	VehOdo	VehBCost	WarrantyCost
count	57789.000000	57789.000000	57789.000000
mean	71469.475679	6737.245505	1245.201492
std	14549.431918	1734.707042	503.338481
min	12628.000000	1400.000000	462.000000
25%	61756.000000	5450.000000	837.000000
50%	73339.000000	6725.000000	1155.000000
75%	82416.000000	7900.000000	1583.000000
max	115717.000000	13195.000000	3298.000000

Fig. 11: In cluster 0 we have 57789 records for each attribute, the typical values for VehOdo are between 61756 and 82416 with a standard deviation of 14549.43; regarding VehBCost between 5450 and 7900 with a standard deviation of 1734.7; regarding WarrantyCost between 837 and 1583 with a standard deviation of 503.33.

	VehOdo	VehBCost	WarrantyCost
count	58386.000000	58386.000000	58386.000000
mean	71478.090518	6730.008335	1276.105042
std	14591.224550	1762.075265	598.885423
min	4825.000000	1.000000	462.000000
25%	61785.000000	5430.000000	837.000000
50%	73359.000000	6700.000000	1155.000000
75%	82427.000000	7900.000000	1623.000000
max	115717.000000	36485.000000	7498.000000

Fig. 12: In the whole dataset we have 58386 records for each attribute, the typical values for VehOdo are between 61785 and 82427 with a standard deviation of 14591.22; regarding VehBCost between 5430 and 7900 with a standard deviation of 1762.07; regarding WarrantyCost between 837 and 1632 with a standard deviation of 598.88.

2.3 Hierarchical Clustering (HCA)

The variables we decided to use to conduct the hierarchical clustering are the ones which gave us the best results in the previous points of clustering, i.e DBSCAN and K-MEANS.

The data we used for the clustering has been normalized using *MinMaxScaler* as we did earlier in DBScan. We then observe how our sample is by analysing the mean and we observe that it has a very similar distribution as the original one.

After that, we run our algorithm with different possible combinations using different techniques and distance metrics.

Some of the hierarchical clustering techniques we used are

A. SINGLE link

B. Complete Link

C. Average

The distance metrics formulas we applied are:

A. Manhattan B. Euclidean

After setting up our parameters using the best variable combinations in K-MEANS i.e VehBCost, VehAge and WarrantyCost with respect to the **Euclidean** method we obtain these results. The dendrogram shows the result we obtained using the euclidean distance algorithm.

Table 4: Results of our euclidean method with different algorithms

Algorithm Method	Number of Cluster	Cut points
Single	3	0..056
Complete	4	0.9
Average	3	0.35

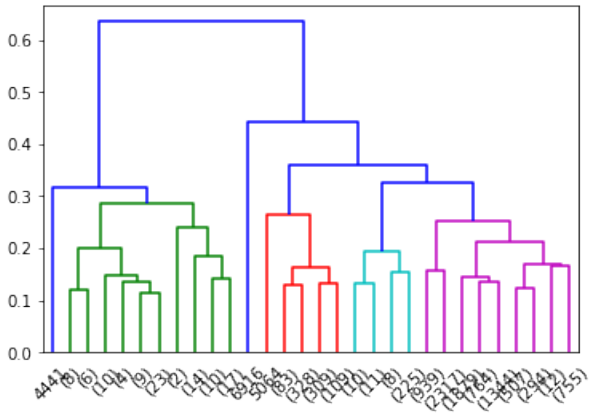

Fig. 13: The graphs represent a dendrogram plotted with Euclidean distance and Average algorithm. The cut has been done at the 0.35 level and it bring us to obtain 3 clusters

Table 5: Results of Manhattan distance with different algorithms

Algorithm Method	Number of Cluster	Cut points
Single	3	0.065
Complete	3	1.2
Average	3	0.4

These are the variables we obtained best result in DBSCAN aand they are VehBCost, WarrantyCost and VehOdo. Below are the results we get after performing hierarchical clustering on them with both the euclidean and manhattan methods.

Table 6: Results of our euclidean method – Variables: *VehBCost, WarrantyCost and VehOdo*

Algorithm Method	**Number of Cluster**	**Cut points**
Single	4	0..05
Complete	3	1
Average	4	0.3

Table 7: Results of our Manhattan method – Variables: *VehBCost, WarrantyCost and VehOdo*

Algorithm Method	**Number of Cluster**	**Cut points**
Single	4	0..06
Complete	3	1
Average	4	0.35

Evaluation of the dendrograms obtained: The decision of where to cut the dendrograms and consequently the number of cluster obtained by them has been taken under two different prospectives.

First, the distances we obtained from the graphs. In other words we decided to cut the dendrograms where the distances from the next point became too high to be comparable with the next one and evaluating also the type of distance metrics we were using for each code. Precisely, you can see that the cut in the single linkage assumes the lowest value due to the reasoning of the lower distance and in the row where we find average and complete. This has been respected also using different distance metrics.

Second, we tried to keep a little difference between the same dendrograms which refer to the two dataframes, although in few cases we find the necessity to increase it slightly to obtain a better result. In the majority of the cases we obtained almost the same numbers of clusters, cutting the graphs around the same values even if they refer to different datasets.

2.4 Final evaluation

The best silhouette score that we obtained was with **DBSCAN** algorithm, the score that we obtained is 0.77 setting eps = 0.9 and min_samples = 5; this kind of algorithm removes outliers. We tried to use the dataset without the outliers removed by dbscan in kmeans but it did not improve the results. We also tried to delete the outliers in the classic way (using quantiles) but also in this way our results didn't improve w.r.t when we keept the outliers in kmeans. But as we explained above, getting two clusters with one containing only 27 elements is not good at all.

Thus our best results, got in K-means, are those obtained setting the k = 4 giving an *SSE score* = 692 and *silhouette* = 0.37 while keeping the outliers.

Furthermore we used two different datasets for DBSCAN and K-means because some attributes fit better in a density based clustering and others do not. However they have been resumed both when we worked on hierarchical clustering. In the last clustering group analysis we found useful to maintain them because of the fact it might be useful to extract rules or conclusions.

One useful evidence we found is that the optimal number of cluster we obtained in K-MEANS which was between 3 and 6 and ideally 4, resulted in getting the best result and was confirmed by HCA where the number of cluster obtained is between 3 and 4.

3 Association rules mining

In pattern mining, we used a dataset containing only the variables that we considered more adapted for this task which are: *Auction*, *VehicleAge*, *Make*, *Trim*, *Color*, *WheelType*, *Size*, *VNST*, *VehBCost_bin*, *VehOdo_bin*, *WarrantyCost_bin* because the others are irrelevant or redundant. In the last three variables we use discretization so we could use them in pattern mining. We binned the variables in 10 bins instead of 17 as the *Sturge's rule* say because in this way we

obtain more rules. We cleaned the dataset dropping the missing values in the variable *Color* and *Size* because they are few and we replace the missing values in a proportional way in *WheelType* and *Trim*.

Setting the *apriori algorithm* in a way that we can find all the frequent itemsets and the closed frequent itemsets in the dataset we obtained:

Table 8: Values of support and extracted itemsets in the case of closed frequent itemsets and all frequent itemsets

MinSupport	**Itemsets**
5	945
10	214
15	80
20	43
25	26
30	14
35	6
40	5
45	2
50	1

We then set the apriori algorithm in a way that will find us the maximal frequent itemsets in the dataset, the obtained itemsets are:

Table 9: Values of support and extracted itemsets in the case of maximal frequent itemsets

MinSupport	**Itemsets**
5	347
10	94
15	36
20	24
25	14
30	14
35	6
40	5
45	2
50	1

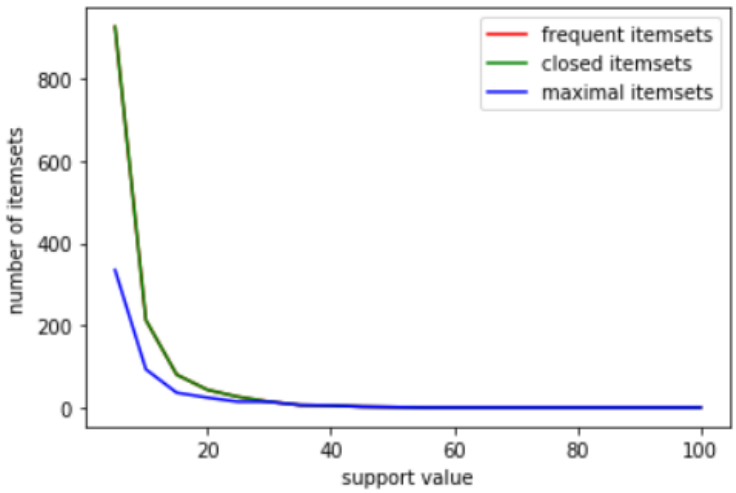

Fig. 14: Comparison between different kind of itemsets extraction

We set the minimum number of items to 2 to avoid itemsets being composed by only one item.
We studied the association rules extracted by using different level of minimum support:

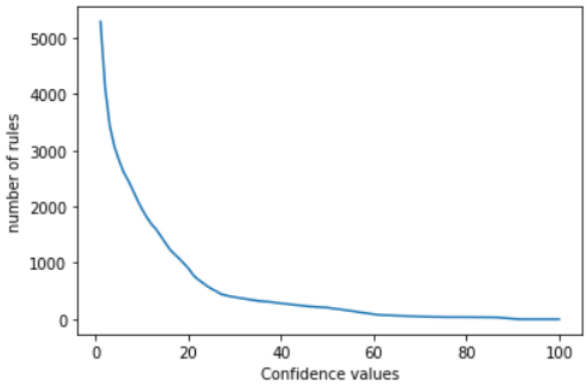

Fig. 15: **Support = 20** – The number of rules goes until 5000

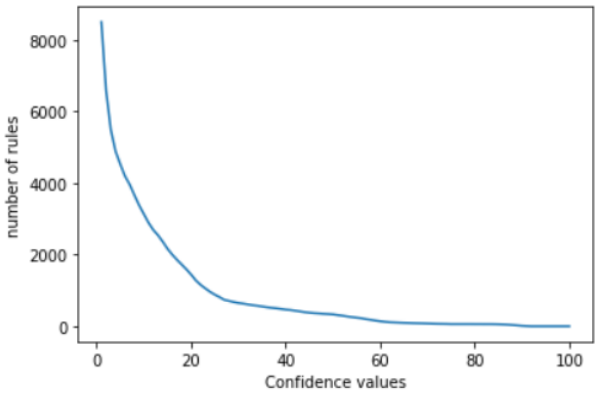

Fig. 16: **Support = 15** – The number of rules goes until 8000

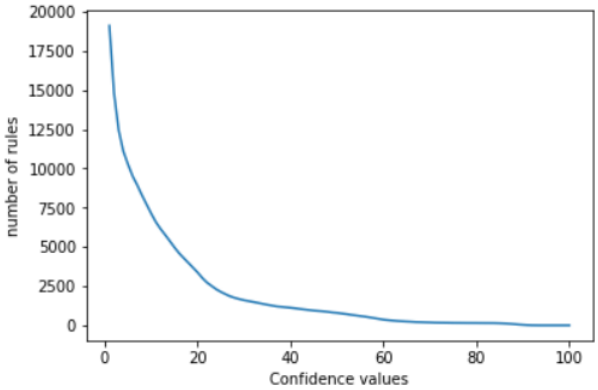

Fig. 17: **Support = 10** – The number of rules goes until 20000

Our plotted graph above explain how many rules we extract for level of confidence between 0 and 100.

The most interesting rules are the ones that have a positive level of *lift*, i.e higher lift, more interesting rules are found in them.

Taking into account the granularity of each attribute in the dataset we used for pattern mining, we decided to choose a support = 10. The frequency histogram of association rules based on confidence and lift values so obtained are:

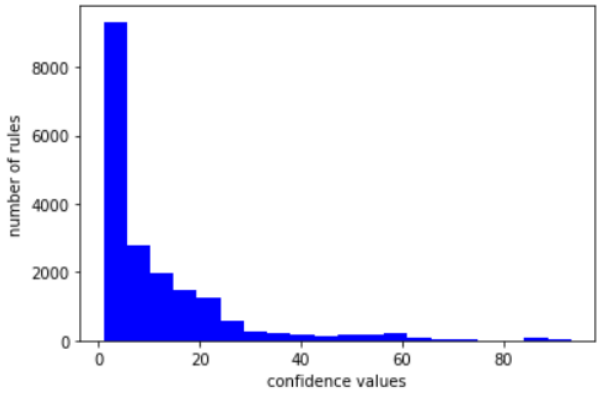

Fig. 18: **Frequency of rules according to confidence values** – We can see that the majority of rules extracted have a confidence lower than 20% but we are interested in those with an high level of confidence.

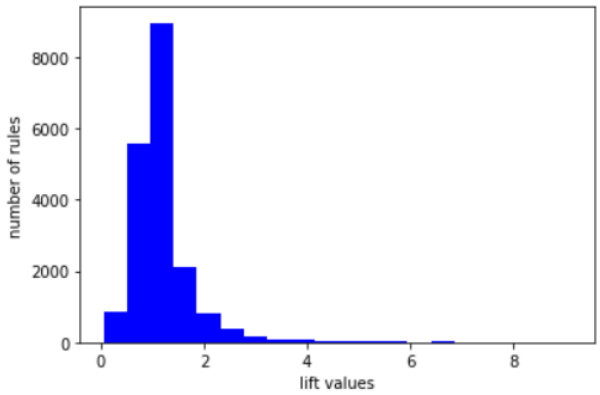

Fig. 19: **Frequency of rules according to lift levels for all confidence values** – We can see that the majority of rules are around a lift value of 1.

Here, we have to take the rules that interested us and applied them to the dataset and saved the predictions in a new column, and then we have to use the classification functions to find the accuracy.

The missing values to replace in the dataset used for pattern mining are:

Table 10: Number of missing values per attribute.

Attribute	**Missing values**
Trim	1911
Color	7
WheelType	2577
Size	4

4 Classification

4.1 Handling the imbalanced target classes

Our target variable, *IsBadBuy*, is very imbalanced. This kind of situation could lead to very biased results and induce lot of errors in the classification of the elements that are actually included in the minority class.

Table 11: Number of elements in each class of *IsBadBuy*

0	1
51178	7208

We used three different methods to improve the learning process of our models. The first one was beneficial for the decision tree (DT) algorithm. The second and third consists of the use of respectively ensemble methods and SVM classifier that perform better under these conditions than a simple DT.

Method 1: Up-sampling the minor class

This method re-samples the data, using *Sickit-learn.utils*'s *resample* function. The idea behind is to duplicate the few observations available of the minority class (here, 1 in *IsBadBuy*) to make them more present. We chose to create as many samples of the minor class as the major one. So the 7208 cases were duplicated until we reached 51178 observations of $IsBadBuy = 1$.

Naturally, this approach have some serious drawbacks. The fact that the observations labelled **1** have now very similar properties. (Ideally, each original observation labelled 1 would have been duplicated at least 7 times, which would make, even in this perfect case, still 7208 cases that have at least 7 similar cases each). This can (and more likely will) trick our model to believe and strongly consider those specific values as very reliable, inducing the omission of relevant values that can be near those repeated ones.
Nevertheless, as we will show in the DT part, it has improved slightly our global scores (*Accuracy and F1*).

Method 2: Using bagging and boosting

The algorithms and their results will be more detailed in the next parts (4.4). We used the ensemble techniques, which rely on decision trees and link them either in a serial (XGBoost) or a parallel (Random forests) way to improve the results (the focus is and will still be on the F1 scores, with a view on the accuracy as well to notice the modifications.)

Method 3: Using SVM classifier Support Vector Machines allow us to consider the imbalanced case, it uses the values of the target variable to automatically adjust weights inversely proportional to class frequencies in the input data. Unfortunately, even though we tried this method, we couldn't get any result because of the computation time it was taking in our limited hardware resources. Thus, this part will unfortunately not be exploited correctly.

4.2 Dummification of the categorical variables

The following was done, to be able to take into account the categorical data, in the classification:

All relevant categorical data, that had a very limited number of classes (three to six) generated directly the same number of new columns with the *[name_of_variable]_class_name*.

The variables that had a higher and non-controllable number of classes were divided using the same approach as the aforementioned method, but only for the classes that counted more than 1000 elements. The other ones were grouped in *[name_of_variable]_Other*. Thus, we had a more limited number of possibilities.

4.3 First algorithm: Decision Trees

a. Direct approach In this case, we directly worked with our (cleaned and pre-processed) data. We did not operate any change other than leaving out the categorical variables, since *sklearn*'s DT classifier does not support them.
We trained our model with a first bunch of parameters' values chosen by ourselves. We obtained a poor result.

b. Hyperparameter Tuning (HT) Here, we went deeper on the hyperparameter tuning, using the same data, but with the best estimator obtained by grid search.

The following hyperparameters were tested in this approach and the next one as well. They are listed below.

Table 12: Hyperparameter values tried in DT

Hyperparameter	**Tested values**
Criterion	Gini, Entropy
MaxDepth	2-10
Min_samples_split	2, 5, 10, 20
Min_samples_leaf	1, 2, 5, 10

c. Dummy variables

We included the categorical variables by creating dummy columns for the most frequent/present classes of one feature, and grouped the rare ones in a column *[name_of_initial_feature]_Other*

Below, we show the best parameters obtained in the different models:

Table 13: Best parameters after grid search in DT without and with Dummy columns (DV)

Parameters	**DT without DV**	**DT with DV**
Criterion	Entropy	Gini
MaxDepth	4	10
Min_samples_split	2	2
Min_samples_leaf	5	1

d. Upsampling the data

This step's goal is to improve the predictions of the elements where $IsBadBuy = 1$. We decided to upsample, as described above, to have a balanced set and accepted the drawbacks and risks of this method. (See 4.1).

The results of the four approaches are listed in Table 14

Table 14: Results of the three methods tried in DT (on **internal** test set)

Try	**Accuracy**	**F1 score (macro)**
Direct approach	0.869	0.501
HT	0.785	0.526
Dummy variables	0.888	0.585
Upsampled	0.781	0.652

What is interesting to notice in our DT models is the scores we get each time. Obviously we get some improvements but we are still far from the expected result or the one we hoped for. However, we managed to get a better score for one of the most interpretable classification algorithms, so it is still useful to have it at least to get an idea of which features can be more relevant to classify our data, and perhaps an insight on why other tree-based ensemble methods performed well or poorly as well. We can highlight the improvement in F1 score we got in the case of upsampling even if we got the worst accuracy of all. When we checked the details, we noticed that we had less True Positives but more True Negatives than we ever got with the other models!

Table 15: Four most important features in the last DT (with DV and Upsampled)

Feature	**Importance**
VehicleAge	0.2757
VehBCost_n	0.1709
MMR_factor_n	0.1455
VehOdo	0.1285

Interpretation What can be extracted from both the DT in 20 and Tables 14 and 15 is the most important features that lead to the classification of our data set. In our best DT model (the one that included the dummy columns with upsamled data), the most important features were the *VehBCost* (normalized) and *VehicleAge*. This

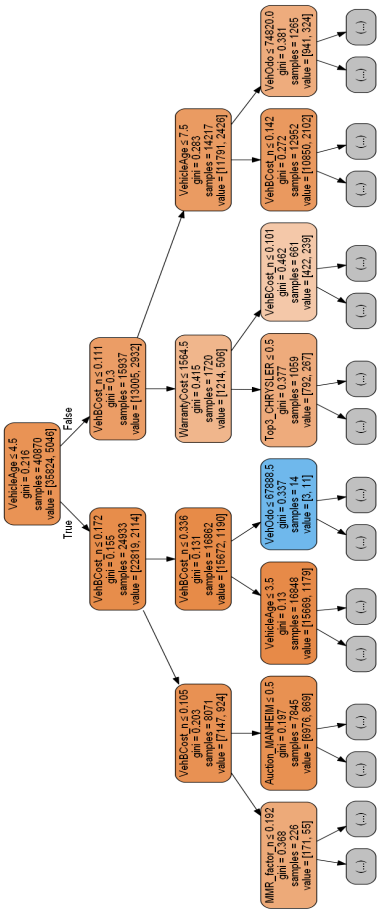

Fig. 20: Final (rotated for a 'clearer' vision) DT obtained after adding the dummy columns

seems natural when we think about it since one of the most important parameters people check before buying a car and considering if it is interesting or not... is the price and when was it built. We can notice then the decrease of the Gini index in one of the leaves (second level of the tree) is important enough to confirm this importance of *VehBCost*, for example.

The gini index on the apparent leaves is not surprising when we check the accuracy and the -not so high- F1 score we get. It gets better of course in the last leaves, which leads to a 0.65 F1 score, but it is still important and shows how big the effect of such an imbalanced target class on the analysis.

Final validation The results shown above were all obtained by teaching the models with **70%** of the training data, then the scores were recorded based on the predictions made in **30% of the left training set**. The final validation now will be made using the real test set. We chose the fourth described model since the evaluation metric is F1 score and it had the best one.

Before applying the model on the provided test set, we naturally applied all the pre-processing steps we performed on the training set: Normalizations, removing some features, creation of dummy categories.

The results are recorded in the table 16

Table 16: Results of the validation process - Decision Tree model with Dummy columns, based on the up-sampled dataset. From our researches, it seems that it shouldn't be very surprising to get some similar values of the different values in cases with a similar kind of dataset and especially with a binary target. Multi target classifications should show different results on the different metrics.

Metric	**Training set (70% TR)**	**Internal test set (30% TR)**	**Real test set**
Accuracy	0.673	0.653	0.4
F1 score macro	0.672	0.652	0.369
F1 score micro	0.673	0.653	0.4
AUC	0.673	0.653	0.549

4.4 Second algorithms: Ensemble methods

The ensemble methods we tested are tree-based. The goal was to enhance the scores using different features of boosting and bagging algorithms and see how the behaviour of our metrics change. We expect both to be get better results since the algorithm of Boosting (XGBoost in this case) focuses on the mis-classified elements to give them a higher wait and tries to give them more "attention" in the following step's tree, while bagging (Random Forests) here combines the performance of multiple trees to enhance the score. We will see if our expectations are true or wrong.

a. Bagging: Random Forests

In this case, we tried a first test with common values then applied directly a Grid Search with different values for the Hyperparameters tuning. The tested values will be displayed in Table 17.

Regarding the feature importance, the order was exactly the same as in the DT's features importance shown in Table 15.

Table 17: Hyperparameter tuning in random forest. Best ones **in bold**

Parameters	**values**
Criterion	Entropy, **Gini**
n_estimators	10, 30, 60, 90, **100**, 120, 150
max_features	None, **auto**
max_depth	**None**, 2, 4, 6, 8, 10, 12, 14
min_samples_split	**2**, 5, 10
min_samples_leaf	**1**, 2, 5
boostrap	**True**, False

Results and analysis of the RF results

The model was tested on both training, internal test, and real test sets. The results were satisfactory compared to what we have got before, with the DT.

Table 18: Results of the best Random Forest model on the training set (70% of TR), internal test set (30% of TR) and provided test set.

Metric	**Training**	**Internal test**	**Real test**
Accuracy	0.999	0.964	0.825
F1 score (macro)	0.999	0.965	0.54

We can easily notice an important overfitting situation here. This is not contradictory even if we consider the results obtained in the internal test set. That good performance in that set is very likely due to the fact that we can find many identical observations (that have $IsBadBuy = 1$) in the training and the test after the up-sampling. Again, here is the best example of the drawback we could get. Although now we realize that we should have resampled only the training set, but we can guess that the performance on the internal test set in that case would be as poor as what we got on the provided test set here.

b. Boosting: XGBoost (Enhanced Gradient Boosting Decision Trees) This was a quick try of one popular boosting algorithm. Another choice would have been, for example, AdaBoost. We chose XGBoost because of the computation time that is probably going to be less important in this one than in AdaBoost, since one of XGBoost's main features is the very highly parallel computations it does.

The results were not as good as expected and are as follows:

Table 19

Metric	**Value training set**	**Value internal test set**	**Value Provided test set**
F1 Macro	val1	val2	val3
F1 micro	0.631	0.627	0.447

4.5 Selection of the final model

Surprisingly, the two ensemble models did not outperform the decision tree as we expected. Instead, we can safely say that Random Forest was better if we consider that it is important to detect more accurately the cars that would be labelled as *IsBadBuy* = 1 (it is better if the goal is to enhance our chances that, for a car that is labelled 0, we are more sure that it is indeed very likely to be true). That was the challenging part in the imbalanced dataset. If the goal is, without caring about the False negatives, to be sure that if a care is labelled as 1, it has higher chances to be true, then a simple DT or Random Forest will be enough.

In our case, since F1 score considers **precision** and **recall**, and the competition's score seem to be averaged according to the micro criteria, we will take the one that got a higher score which was the Random Forest (on final test set; F1 macro = 0.54, F1 micro = **0.83**). It is also as we explained the one that had the most True Negatives of all tested models.

5 Further researches, perspectives

Our results evaluated our choice of algorithms and we could see the limitations of each one. Besides the problems due to the nature of our dataset, some of these limitations are undoubtedly due to our own choices.

In order to be more assured of the quality of our work, more pattern mining techniques should have been employed. We should have went deeper in our own work and measured the accuracy in that part to evaluate our result.

Regarding classification and supervised learning, we could certainly have gone further to apply Nave Bayes, K-Nearest Neighbor, or AdaBoost for instance. Trying Neural Networks would also have been a plus for our own learning purpose but also perhaps would have improved the scores we got.

Another perspective or improvement could have been a deeper approach with DBScan, further than the simple separation of the data from some outliers as clusters.

We therefore look at these possibilities as a basis for a future work independent work.

6 Conclusion

The project's goal, as stated in the beginning, was to be able to determine or predict whether a dealer buying a used car bought a good product or not.

The cost of buying cars depends on the condition of the car and this was well reflected in the correlations and the analysis we performed as well.

Our work performed different knowledge discovery techniques to have an insightful information of our data-set. Dealing with data cleaning, data visualization, we selected attributes that fitted better than others in the different stages of the project.

After finishing the Data Cleaning and visualization we applied the different Clustering techniques such as KMeans, DBSCan, and Hierarchical clustering (HCA). During the process we observed and analyzed the different clusters formed by each algorithm and determined which is the best defined clustering.

We went further and applied association rules in the data-set to see what important rules we can obtain to explain the target. We performed the task by using the apriori algorithm. We observed that rules with high lift or positive values provide the best rules.

In our Classification process we applied different classification techniques such as Decision Tree and Ensemble Methods and tried to handle the imbalance nature of the dataset to extract useful information of the data. The results we obtain in random forest gives us the most useful extracted information that we learned. We tested all our models to the training and test data and therefore we conclude that Random forest is best among the algorithms we used.

Analyzing all the above data mining techniques from Clustering, Association rules and Classification, we can determine its effectiveness and feasibility. The type of information to be obtain depends on the different type of mining techniques and analyses used as well as the dataset's nature. So therefore to retrieved very meaningful information one needs to try different techniques in order to see which one have the best performance.

References

Carvana dataset : https://www.kaggle.com/c/data-mining-20192020-unipi
Data Mining Course page: http://didawiki.di.unipi.it/doku.php/dm/start