IN SUPREMÆ DIGNITATIS 1343

UNIVERSITÀ DI PISA

Data Mining Report

Group 12

Analysis of car auctions

Lorenzo Bellomo
Andrea Bruno
Marta Scalisi
Chiara Spampinato

January 2020

1 Data Understanding

The data set contains information about car auctions. The majority of attributes give us information about vehicles. In this section, we'll explain how we studied our data, evaluated the quality of them, how we fixed any missing values (invalid or misleading), in order to gain general insights about the data that would potentially be helpful for the complete analysis.

1.1 Data Semantics

The semantic meaning of the most important (in our opinion) variables in the dataset are described in the following, while the remaining columns are not described either because their meaning is too obvious or because its discussion is delayed to the following sections.

- *RefId*: attribute which defines an unique ID number assigned to each vehicle;
- *Auction*: categorical attribute which defines three different auction providers (Manheim, Adesa, other);
- *Make*: categorical attribute which gives us the name of the production company of the vehicle;
- *VehOdo*: numerical discrete attribute that points out the mileage;
- *VehBCost*: continuous numerical attribute indicating the purchase price auction;
- *MMRA*: numerical attributes that define acquisition price of the vehicles, divided into:
  - average condition price at time of purchase
  - above average condition price at time of purchase
  - average condition price at time of purchase in the retail market
  - above average condition price at time of purchase in the retail market
- *MMRC*: numerical attributes that define the current price of the vehicles, divided into:
  - average condition price as of current day
  - above condition price as of current day
  - above average condition price as of current day
  - above average condition price as of current day in the retail market
- *WarrantyCost*: variable expressing cost of the warranty for the car at the time of the auction;
- *IsBadBuy*: this is our target variable, and it expresses if the car was a good deal or not.

Furthermore, analyzing all the attributes there are two of them, Acquisition Type and Kickdate, which aren't in the dataset even though they are present in the dictionary.

1.2 Variables Transformation and elimination of redundant variables

We decided to split the attributes *Model* and *SubModel*, because, within them, too much information was contained. As a result, we use this information to create five different attributes:

- *Engine Liters*: this information was in both the Model and SubModel columns, usually in the form of a float number and a L (i.e. 2.7L);
- *NumCylinders*: this information was also present in both variables (i.e. V4, V-8, I4...);
- *4X4*, four-wheel drive;
- *WheelDrive*, other drive wheel configurations (i.e. 2WD, 4WD...);
- *NumDoors*, the number of the doors in a car (i.e. 5D);

The information contained in the five new attributes is not very interesting in itself, but with this approach we were able to "clean" the both columns from their meta-information. Taking as example the Model column, we were able to go from over 1000 unique models in the original data set to just around 200 without loss of information.

We also created four different attributes in order to separate the information contained in the variable *PurchDate* into PurchYear, PurchMonth, PurchDay and PurchWeekDay (which is the working day of acquisition, like Monday, Tuesday. . .).

For the sake of simplicity, from now on, the 8 MMR variables will be addressed with an abbreviation. For example, MMRAcquisitionRetailAveragePrice becomes ARAP, MMRCurrentAuctionCleanPrice becomes CACP, and so on.

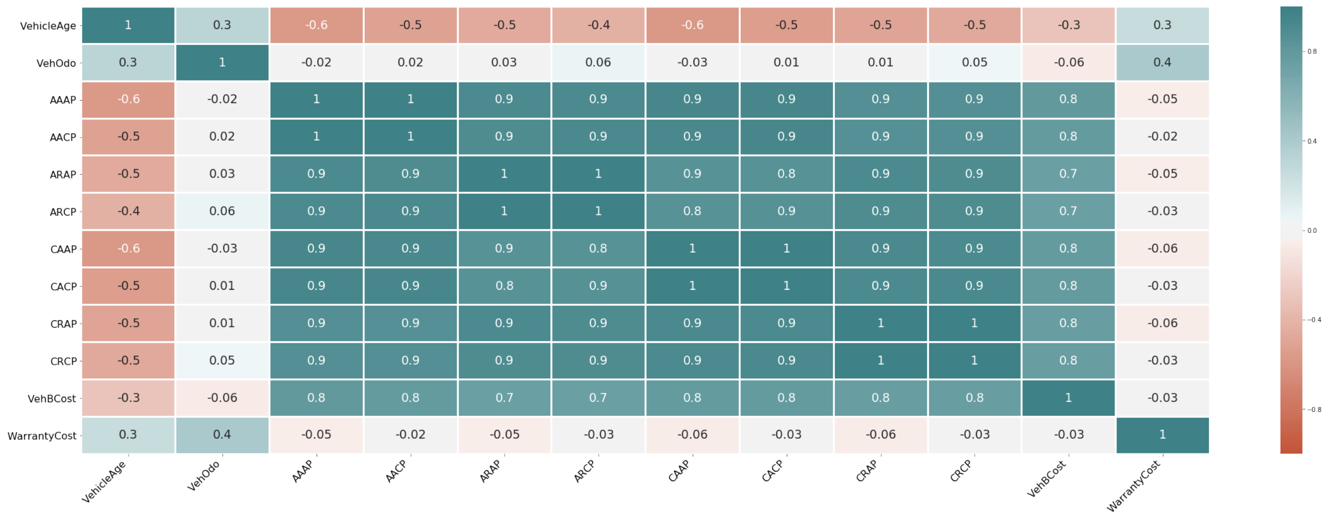


Figure 1: Correlation matrix

Finally, analyzing the correlation matrix (shown in the figure1,) of all the continuous attributes we can see that the 8 MMR attributes are strongly correlated (close to 0.9). Therefore, we reduced them into two attributes by using the algorithm PCA used to reduce the dimensionality of the data. We created 2 variables from the original 8, namely PCA1 and PCA2. The first one became the most correlated to the original prices, while the second one had a lower correlation and mainly expressed the variance between different attributes.
However, for the sake of clarity, we used PCA1 and PCA2 only in section 3, when dealing with classification. We instead decided to use one candidate price (MMRAcquisitionAuctionAveragePrice, or AAAP for short) in section 4 and section 2 when describing Pattern Mining and Clustering, in order to have clearer semantics regarding car prices. The discussion regarding the distribution of MMR attributes and PCA ones is delayed to section 1.4.

1.3 Assessing data quality

In order to assess the quality of data, we performed some variable wise consistency checks (for example on VehYear, VehicleAge, VehOdo, WarrantyCost), but we did not find any notable error. However, we found a lot of 0 values for the 8 MMR prices (in addition to its missing values), and we also identified one clear outlier in the VehBCost column, with this car having been sold at 10$. We then decided to drop this row.
Outlier wise, we found that the MMR prices were usually coherent with respect to the VehBCost attribute, so we decided to keep in the data set all the other rows. In this, we also kept a lot of very high-cost cars, but we could not label these cars as outliers, as we thought that this behavior is to be expected.

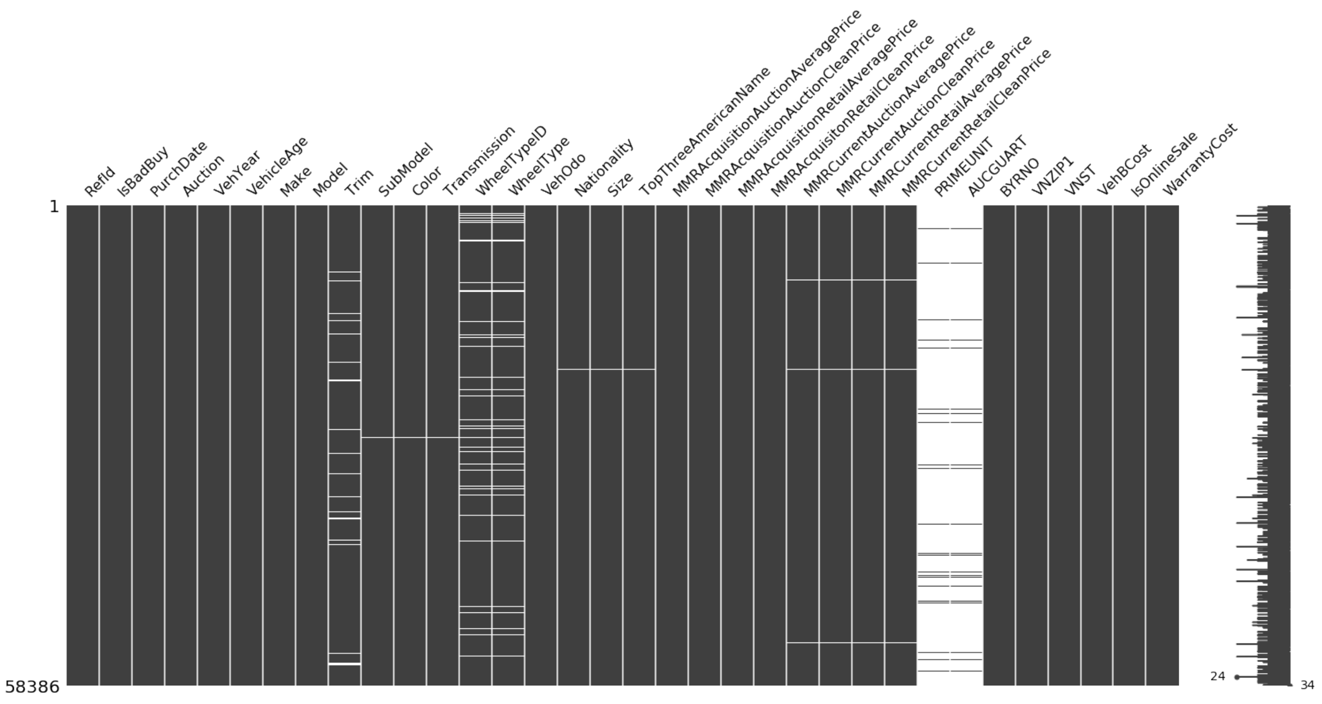


Figure 2: Missing values

By looking at figure 2 it is possible to see the missing value of our dataset (identified by the white lines). The attributes PRIMEUNIT and AUCGUART have too many missing values, so we decide to discard and not use them.

Analyzing the others, we chose to correct the variables Trim, Liters, Cylinders, Transmission with the algorithm MICE (Multiple Imputation by Chained Equations).

1.4 Attributes Distribution

In this section, we will analyze the distribution of some particular attributes, showing interesting statistic plots.

The first thing to note is that the target variable (IsBadBuy), is highly imbalanced. Good buys are $87 \sim 88\%$ of the population, while the remaining $12 \sim 13\%$ are bad buys.

In the plot shown in figure 3, we show the distribution of the 8 numerical attributes which point out the different prices of Vehicle as we explained in the first part of Data Semantics (section 1.1).

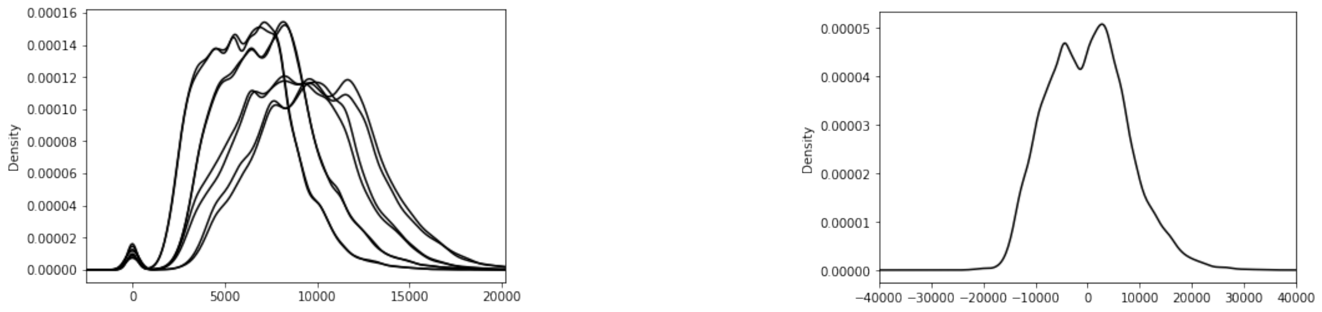


Figure 3: Distribution of 8 MMR prices and of PCA1

We are able to notice that MMR attributes are very correlated, but the correlation is ever higher when considering them pairwise (when considering clean price and average price together).
We are also able to notice that, before variable cleaning, there is a very high peak corresponding to the 0. This peak is no longer present in PCA1, as before applying the algorithm we considered those 0s as missing values and imputed them.

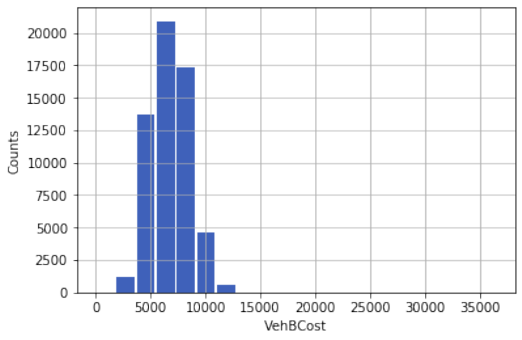


Figure 4: Distribution of the attribute VehBCost

Analyzing the attribute *VehBCost*, we can observe that Vehicles are usually sold for a price between 6000 and 7000, and a very low percentage of cars is sold above 11000 or below 5000 USD.

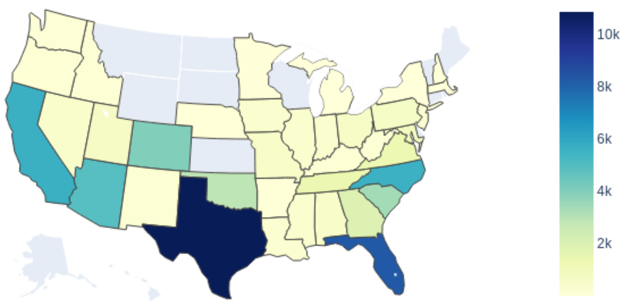


Figure 5: Distribution of the attribute VSNT

For the attribute *VSNT* we decide to plot the distribution on a map of the United States. By looking at the figure, it is possible to see that the major number of auctions is done in Texas (18800), followed by Florida (8317) and California (5673). On the other hand, the state with lower auctions is New York (4). Furthermore, there are no auctions in Montana, Wyoming, North Dakota, South Dakota, Kansas, Wisconsin, Maine, Vermont, Rhode Island and Connecticut.

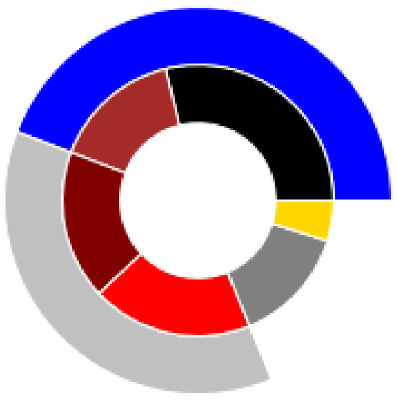


Figure 6: Distribution of the attribute Color

As far as it concerns the attribute *Color* (figure 6, it is not surprising that the most common colors are blue, silver and white.

By analyzing also the distribution of the variable IsBadBuy in figure 7, plotted with respect to WarrantyCost and VehOdo on the left, and with respect to WheelType on the right, we can notice a few different things. Regarding the left image, bad purchases are more frequent in less dense areas. Regarding the image on the right, we are able to notice that, whenever WheelType information is lacking, the number of bad buys is huge with respect to the variable distribution (corresponding to the column labeled as *Unknown*).

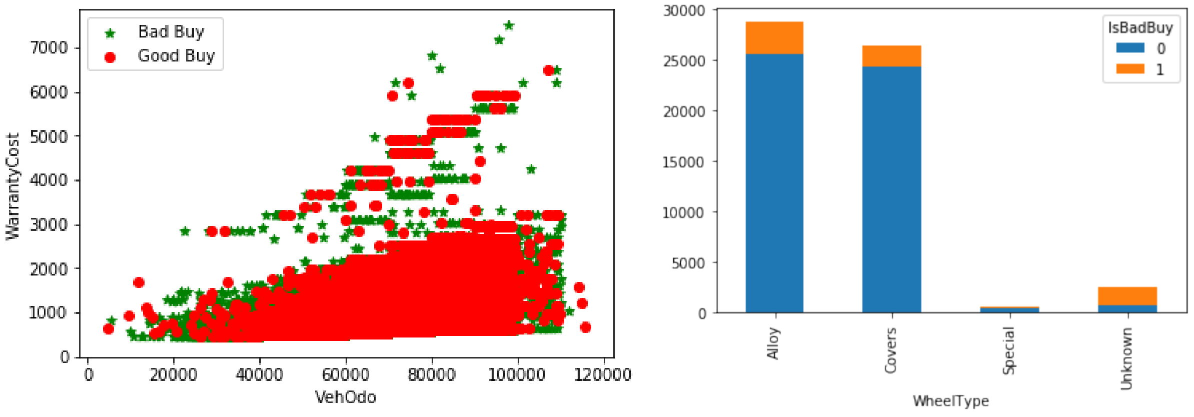


Figure 7: IsBadBuy distribution with respect to VehOdo and VehBCost

2 Clustering

In this section we describe the three Clustering algorithms applied to the data set (KMeans, DBScan and Hierarchical), and we describe the results

2.1 Function Selection

In K-means we tried to use both the *MinMax* scaler and the standard z scaler, observing that the clustering results were very similar. In the end, we decided to use the MinMax one. In DBScan and Hierarchical we decided to adopt the Standard one because it gave us slightly better results.

2.2 KMeans

The following sections discuss the analysis of the results of KMeans clustering algorithm.

2.3 Attributes' selection

Considering that our dataset contains information about car auctions, we opt to study characteristics about cars, like how many kilometers the car has done (**VehOdo**), the auction selling price for the car (**VehBCost**), the cost of repairing or replacing previously sold products (**WarrantyCost**) and some samples of the different prices (**AAAP, ARAP**).

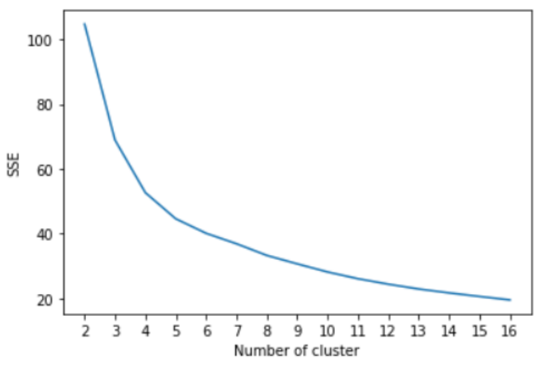
We created 5 Dataframes with these attributes in order to study which was the best combination of them.

	Attributes set
1	'VehOdo' , 'VehBCost', 'AAAP'
2	'WarrantyCost' , 'VehBCost' , 'AAAP'
3	'AAAP' , 'ARAP' , 'VehBCost'
4	'WarrantyCost' , 'VehOdo' , 'VehBCost'
5	'WarrantyCost' , 'AAAP' , 'VehOdo'

2.3.1 Identification of best k

In order to pick the best parameter k for K-Means, we made use of the Knee method by computing the SSE for K $\in$ [2,16]. The best SSE was obtained in the Data Frame 3, which was originally chosen for his highest correlation among the attributes.

However, when we tried plotting the data, we did not obtain any interesting information in order to better interpreter the data behaviour. We then decided to give up the best SSE by choosing the Data Frame 4 which has a lower SSE but semantically more interesting results.



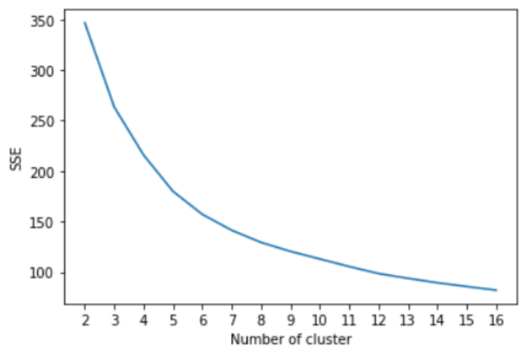


Figure 8: Plot SSE of Attribute set 3 (left) and 4 (right).

We noticed that the SSE curves for the 5 data frames share a strong similarity (same curvature, but different SSE). Their behaviour is very similar to the ones shown in figure 8, and we found, using the *knee method*, that the best attribute k for all the data frames was 6. All the values collected for the the different data frames are shown in table 1. In particular, the lowest SSE is by far the one of data frame 3.
However, we decided to discard this result since the 6 clusters found by the algorithm were basically groups of cars in different price ranges. We decided that the best clustering, both semantically and parameter wise, was the DF4 one (taking in consideration 'WarrantyCost', 'VehOdo' and 'VehBCost').

	Best k	SSE	Silhouette
DF1	6	106.0	0.309
DF2	6	90.0	0.307
DF3	6	40.0	0.294
DF4	6	157.0	0.287
DF5	6	184.0	0.278

Table 1: Summary of the SSE, Silhouette and k values obtained for all the Attribute sets with K-means

2.3.2 Description of the best clustering

The following descriptions refer to the results of the clustering that were proposed as the best in previous section. Every result is presented with its centroid, that describes the core point of the cluster, and a textual interpretation of the kind of cars that are present in those clusters. The centroids coordinates are expressed like this:

$$\mathbf{centroid} = (\mathbf{VehBCost}, \mathbf{WarrantyCost}, \mathbf{VehOdo})$$

In figure 9 the clustering results are shown plotted in 2 dimensions (WarrantyCost and VehOdo). The plot does not show the third dimension (VehBCost) because, from the results, this latter one results the least important dimension (as it will be evident from the centroids shown when describing the clusters). The figure also shows the cluster centroids with a star on the plot.

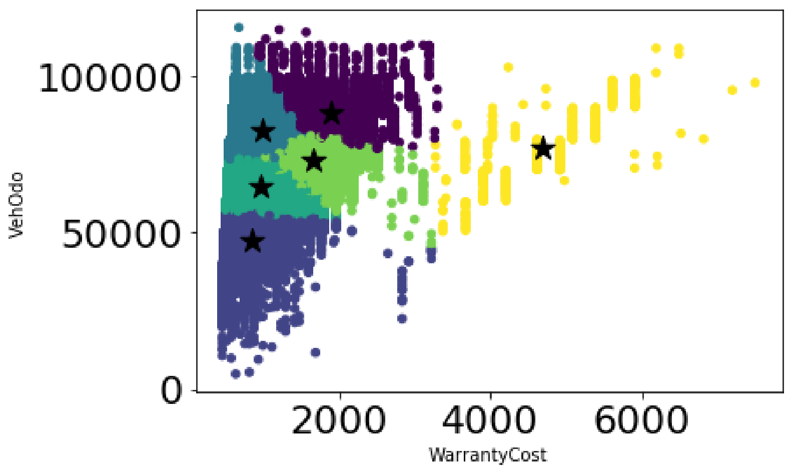


Figure 9: Clusters plotted with WarrantyCost and VehOdo

The clusters descriptions for figure 9 are:

1. **centroid** : (6 500, 1 900, 88 000): cars with very high odometer reading and pretty high warranty cost (purple cluster). Those cars are sold for a price which is in line with the mean of the prices.
2. **centroid** : (6 800, 850, 48 000): cars which are pretty new , with low reading and low warranty cost (light blue cluster). As expected their cost is slightly above average.
3. **centroid** : (6 400, 1 000, 83 000): cars with high odometer reading but low warranty cost (azure cluster). Those cars are probably considered to be solid (low outage risk) even after years of use, and are sold at a normal price.
4. **centroid** : (6 700, 1 000, 65 000): cars with low warranty cost and average odometer (aquamarine cluster). There is not much to say about this cluster, as it represents the average car.
5. **centroid** : (7 300, 1 700, 73 000): Cars with high warranty cost, but average odometer reading (green cluster). This is one of the more interesting cluster, as it shows that relatively high risk cars are sold at a price which is higher than expected (considering high warranty cost as a sign of risk).
6. **centroid** : (5 300, 4 700, 77 000): Exceptionally high warranty cost, high odometer reading (yellow cluster). This cluster is the least populated one, and the most sparse one. It homes those very risky buys, and in addition to that, cars in this cluster are also pretty dated. They are sold, as expected, at a very low price.

Given these outcomes, we tried to understand if bad buys were located mainly in one of those clusters. By plotting this information in figure 10, we noticed that cluster 2 (new cars with low odometer reading), has the least amount of bad buys.

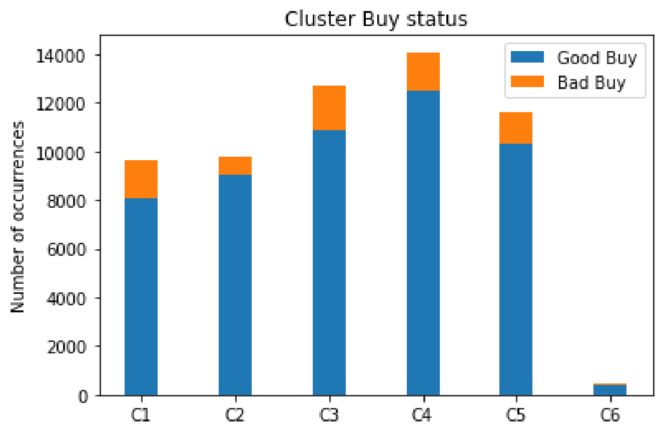


Figure 10: Distribution of IsBadBuy with respect to the 6 clusters found

2.4 DB Scan

In this section, we explain the approach used to generate clusters with DBscan algorithm.

min points	ϵ
32	0.75
64	0.95
128	1.22
256	1.36
512	1.64

Table 2: K-th nearest neighbours parameters

min points	ϵ
32	0.17
64	0.22
128	0.29
256	0.38
512	0.48

Table 3: Manually found parameters

2.4.1 Attributes and distance function

We decided, following the same reasoning we used for *KMeans*, to attempt clustering over the same set of attributes. We also decided to use Euclidean distance, and Z-Score scaling for the data frame.
The results shown in the following sections are only relative to the data frame with columns 'VehOdo', 'VehBCost' and 'WarrantyCost' (the same data frame used for KMeans). Other possible attributes choices did not change much the final result, so we decided that using the same attributes allows us to more easily see the difference between the two algorithms.

2.4.2 Study of the clustering parameters

In order to choose the right ϵ and **minpoints**, we adopted the knee method by plotting the distance to the *k-th* nearest neighbour, where k was taken from [32, 64, 128, 256, 512]. The resulting curves, shown in figure 11, were used to select the right epsilon for attempting the clustering with DB-Scan.

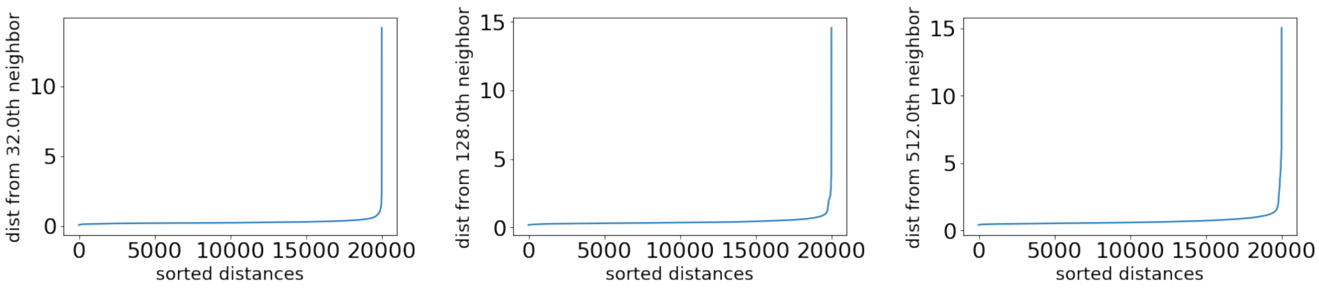


Figure 11: k-th neighbour distance, with k = 32, k = 128, k = 512

Given those plots, we chose epsilon as shown in table 2. It is important to know, however, that this approach failed for reasons described in section 2.4.3, so another set of attributes with more interesting results is shown in table 3. Those values were found by brute force, by attempting, for all k shown in the list before, $\epsilon = 0.1, 0.11, 0.12 \ldots 0.8$, and visually inspecting the results.

2.4.3 Characterization and interpretation of the obtained clusters

First, we are going to analyze the results with parameters shown in table 2. The result was that of a single cluster, containing all the points in the data set, with the exception of ~ 100 elements, which were labeled as noise points. This is because the data forms one big cloud of points, with different density distribution inside. This kind of behaviour represents the conditions under which DB scan performs worst, and this is the reason why the *k-th* neighbour distance approach failed.
We then decided to try and find the most dense areas in the data set, by manually checking a lot of configurations. This approach, however, does not find cluster, but it only finds highly populated areas in the data set. The best results were found when the number of noise points was close to half the total amount in the data set. Those results correspond to the ones found with the parameters shown in table 3 and some example of such clustering is shown in figure 12.

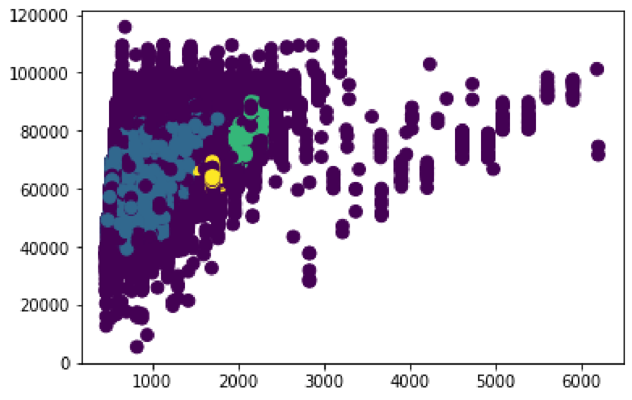
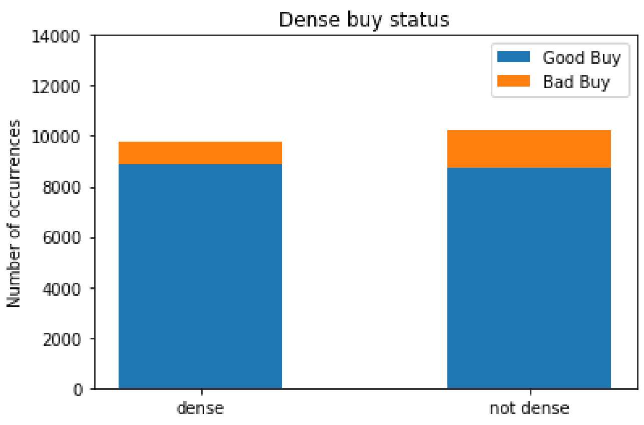



Figure 12: DBScan clustering results with **minpoints** $= 256$ and $\epsilon = 0.38$. Purple colors are noise points

We realised that most of the cars sold have $50\,000 \sim 70\,000$ odometer reading when sold, and denser areas have slightly less bad buys overall.
Having said that, DBscan is the algorithm that performs the worst on this data set.

2.5 Hierarchical Clustering

In this section, we explain the approach used to generate clusters with Hierarchical algorithm.

2.5.1 Attribute Choices

We decided to perform clustering on the following attributes set:

1. 'VehOdo', 'VehBCost', 'AAAP'
2. 'WarrantyCost', 'VehBCost', 'VehOdo'

2.5.2 Algorithms and Dendrograms

We decided to perform clustering with Euclidean and Manhattan distance as metrics, and to perform, for each of those metrics ward, single, complete and average linkages (with the exception of manhattan distance with ward linkage, since it is not allowed).
For each one of those results, we attempted clustering with **numberOfCluster** $\in [2, 10]$, and computed the silhouettes for all the results. Figure 13 shows the silhouettes for the results found with all the algorithms on data frame 2 (the same used for KMeans and DBScan).

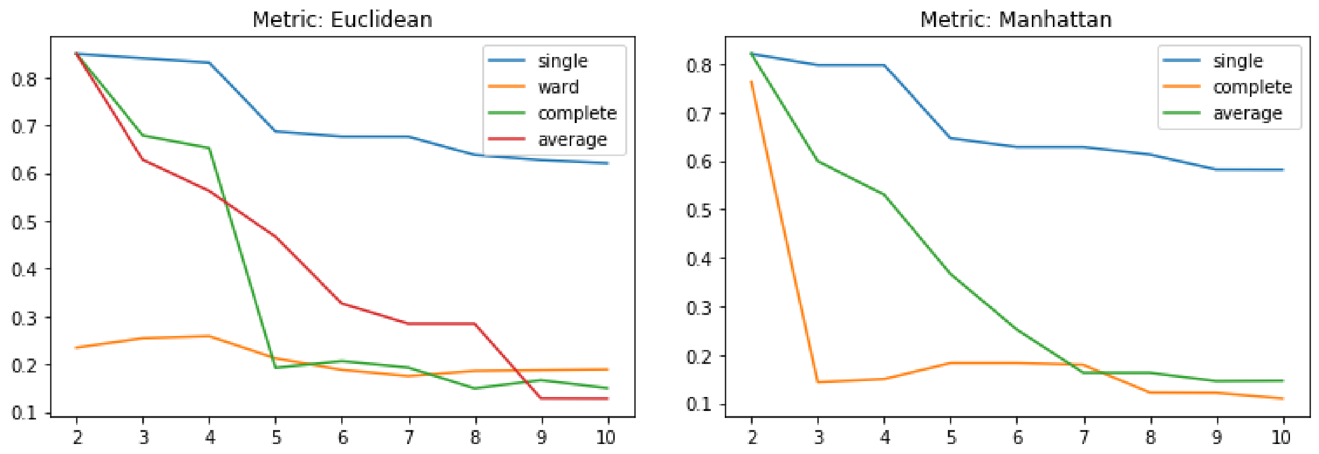


Figure 13: Silhouettes for all algorithms and all metrics on data frame 2

From those plot, we notice a tendency for the silhouette to drop when the number of cluster passes from 4 to 5. We then decide to perform clustering with 4 clusters. Given that, we visually inspected the results and found that the only ones with interesting clusters are:

- Euclidean metric and ward linkage
- Manhattan metric and complete linkage

The visual result of those clustering is shown in figure 14, while their respective dendrograms are shown in figure 15. All the other clustering attempts produced highly imbalanced cluster (one main cluster and some single digit size clusters).

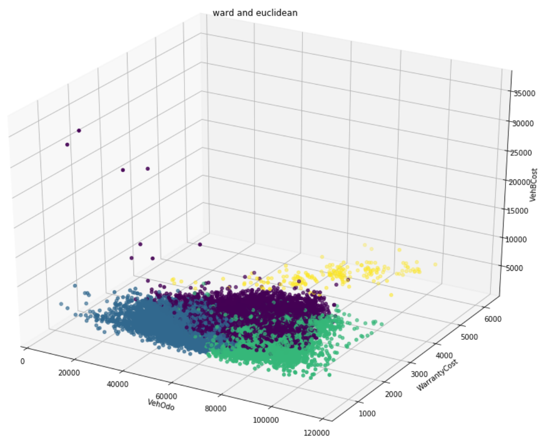


Figure 14: Hierarchical clustering results, number of clusters is 4

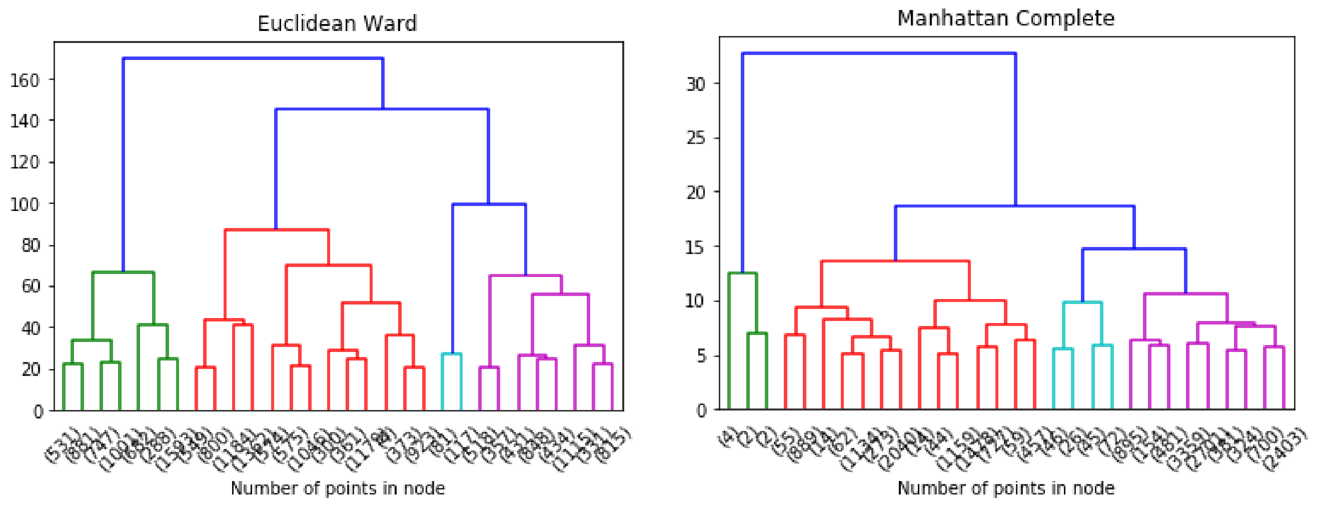


Figure 15: Dendrograms plotted with *lastp* truncate mode

2.5.3 Best clustering approach and comparison of the clustering obtained

In conclusion, the best clustering results were found in the circumstances shown in section 2.5.2. The results, semantically speaking, highly resemble the ones found using KMeans. Both results find a cluster in the high warranty cost cars (displayed in both figure 14 and in figure 9 for KMeans, where the highlighted cluster is displayed in yellow).
The main difference is in the way that points in the "big cloud" are assigned a cluster. The reasoning, anyway, is really similar to the one made in section 2.3.2 regarding KMeans, so we refer to that one.

3 Classification

In the following section, we describe the methodologies and the algorithms used during the classification. The main goal of this task was to predict the variable called *"IsBadBuy"*, that indicates whether a car has been a good business or not.

3.1 Hyper-parameters Optimization

To discover the best way to predict the required variable, we tested a lot of models by optimizing their hyper-parameters. Those act as knobs to fine-tune the model, so to provide the best result, we need to find out the optimal value of these parameters, or in other words, a trade-off between true/false positives and true/false negatives. Since each algorithm has its peculiarity, for each classificator, we created different groups of parameters to experiment. Table 4 shows the setup of our analyses.

Algorithm	**Hyper-parameters**
Random Forest	*n_estimators*: 25, 50, 100, 200, 500, 1000
Decision Tree	*criterion* : 'gini', 'entropy'
	max_depth : 2, 5, 10, 15, None
	min_samples_split : 2, 5, 10, 20
AdaBoost	*n_estimators* : 5, 10, 25, 50, 100
	learning_rate : 0.1, 0.25, 0.5, 0.75, 1
KNN	*n_neighbors* : 1, 4, 7, 10, 13, 16, 19, 22, 25, 28
	weights : 'uniform', 'distance'

Table 4: Setup environment of the tested hyper-parameters

3.1.1 Methodology

The first step was to isolate the test set ("test.csv") because this will be used as ground truth to verify the performance of the created models. Then, we cross-validate each model by using as input data, the given training set ("training.csv"). To reflect the percentage of the initial datasets, we decided to split the data in 60-40 (respectively training and validation). For each algorithm of classification, and for each tuple of parameters, we performed 5 Cross-Validation and we averaged the results to circumvent overfitting/underfitting. Do note that to improve the classes imbalance, undersampling and oversampling techniques have been applied.

3.2 Results

The following pictures show the performance obtained by each algorithm during the optimization. Our strategy was to focus more on optimizing Recall and F1 because our goal was to discover "bad buys". The yellow dots/squares represent our best choice of hyper-parameters. Overall the most suitable classifier for this task were decision trees and random forest.

3.2.1 Random Forest

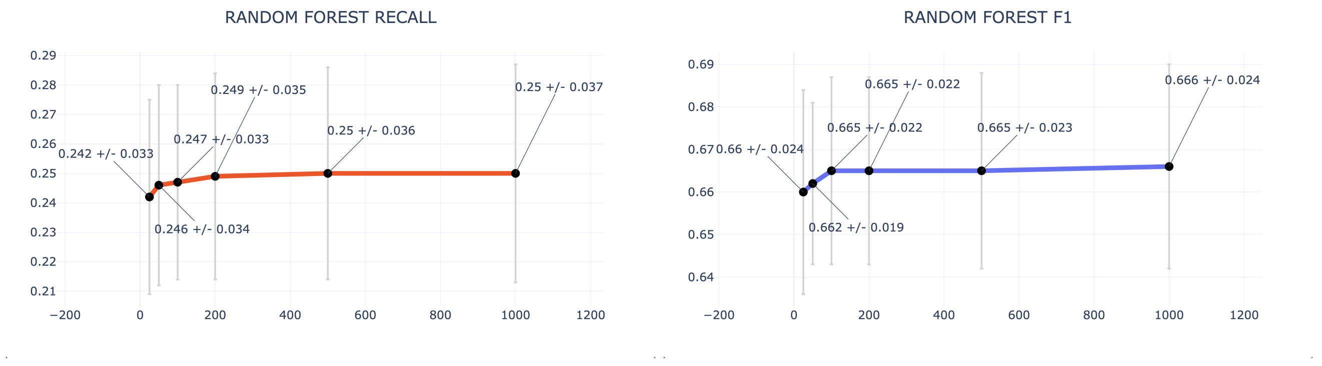


Figure 16: Random Forest tuning

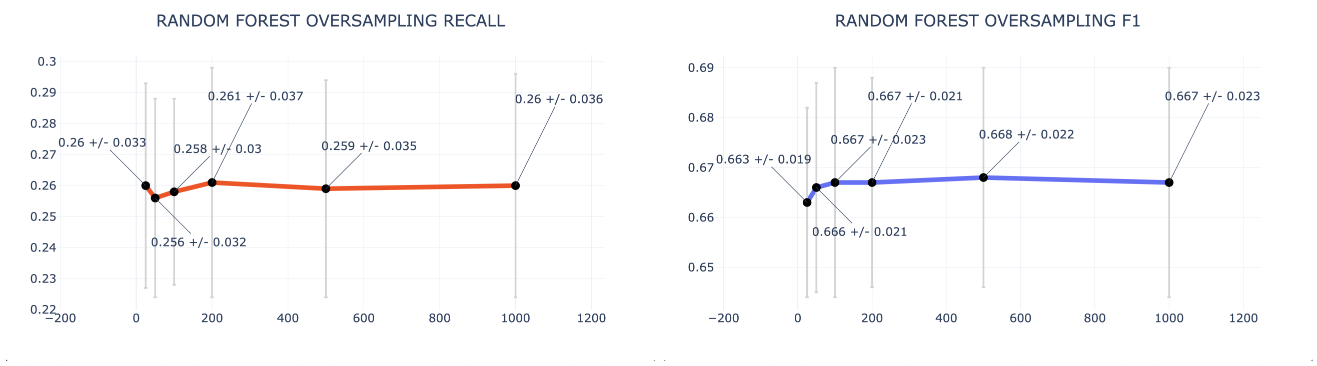


Figure 17: Random Forest + Oversampling tuning

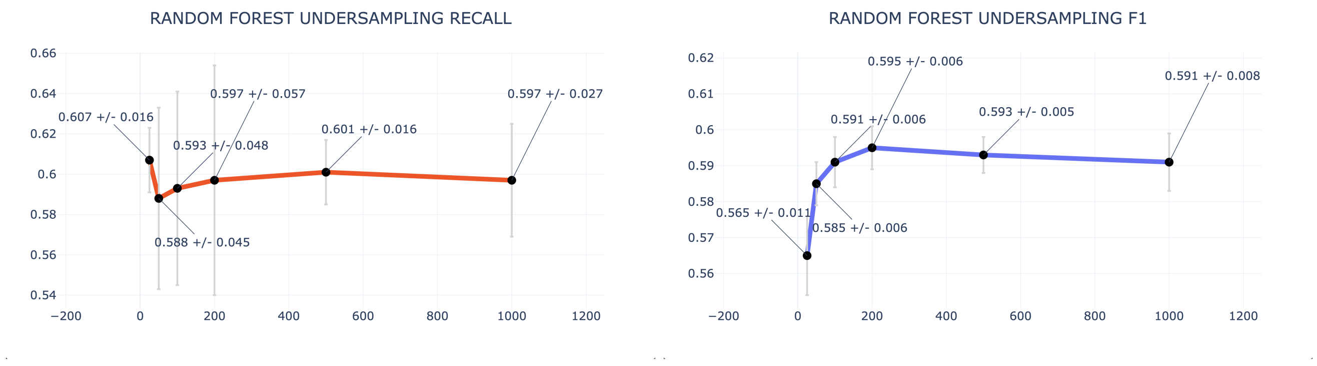


Figure 18: Random Forest + Undersampling tuning

3.2.2 Decision Tree

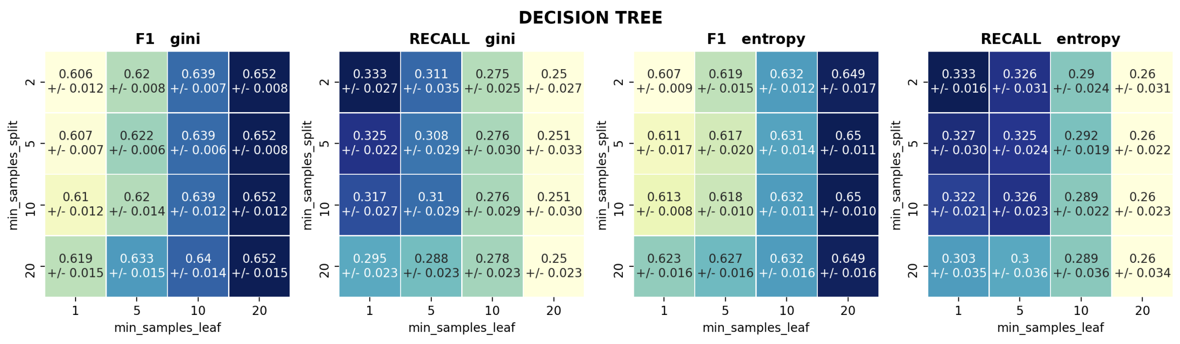


Figure 19: Decision Tree tuning

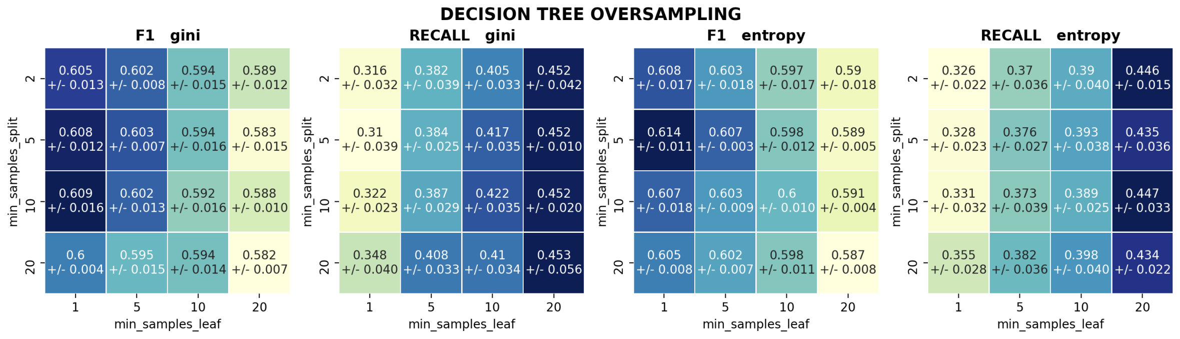


Figure 20: Decision Tree + Oversampling tuning

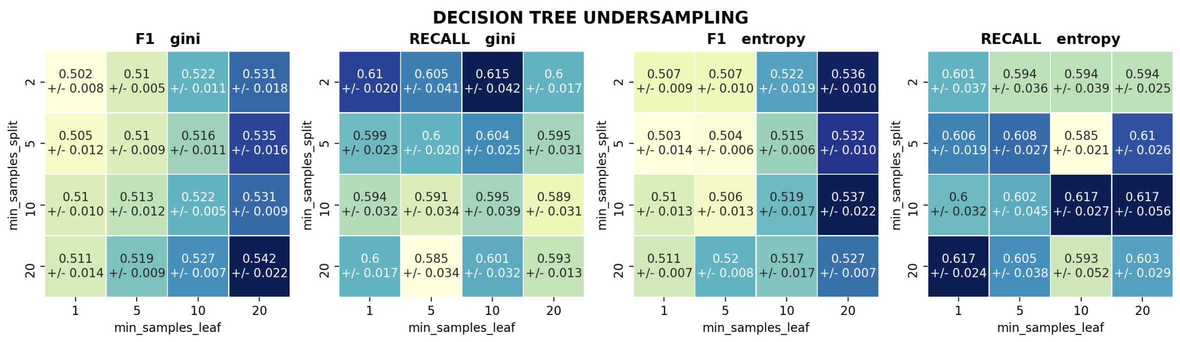


Figure 21: Decision Tree + Undersampling tuning

3.2.3 AdaBoost

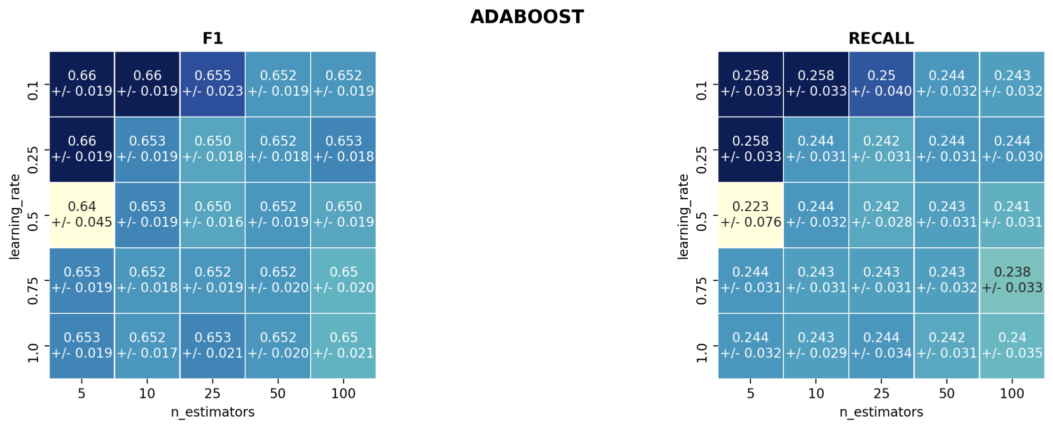


Figure 22: AdaBoost tuning

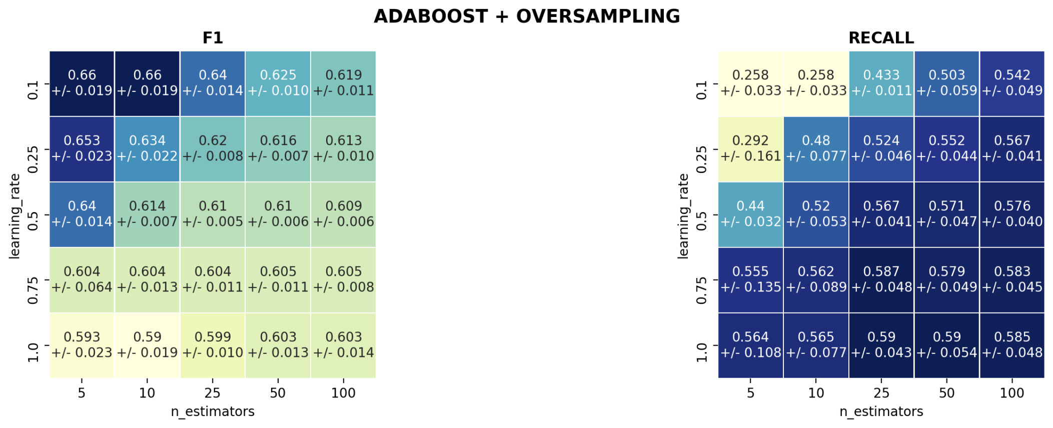


Figure 23: AdaBoost + Oversampling tuning

	Accuracy	Precision			Recall			F1-Score		
		0	1	*AVG*	0	1	*AVG*	0	1	*AVG*
Random Forest	**0.90**	0.90	0.81	**0.85**	0.99	0.22	**0.61**	0.95	0.35	**0.65**
Random Forest + OverSampling	**0.90**	0.90	0.78	**0.84**	0.99	0.22	**0.61**	0.94	0.35	**0.65**
Random Forest + UnderSampling	**0.74**	0.92	0.24	**0.58**	0.76	0.54	**0.65**	0.84	0.33	**0.58**
Decision Tree	**0.89**	0.90	0.68	**0.79**	0.99	0.23	**0.61**	0.94	0.35	**0.64**
Decision Tree + OverSampling	**0.72**	0.92	0.23	**0.58**	0.75	0.54	**0.64**	0.83	0.32	**0.57**
Decision Tree + UnderSampling	**0.66**	0.92	0.20	**0.56**	0.66	0.60	**0.63**	0.77	0.30	**0.54**
AdaBoost	**0.89**	0.90	0.68	**0.79**	0.99	0.23	**0.61**	0.94	0.35	**0.64**
AdaBoost + OverSampling	**0.89**	0.90	0.68	**0.79**	0.99	0.23	**0.61**	0.94	0.35	**0.64**
AdaBoost + OverSampling	**0.89**	0.90	0.68	**0.79**	0.99	0.23	**0.61**	0.94	0.35	**0.64**

Table 5: Results of the optimized algorithm over the test set

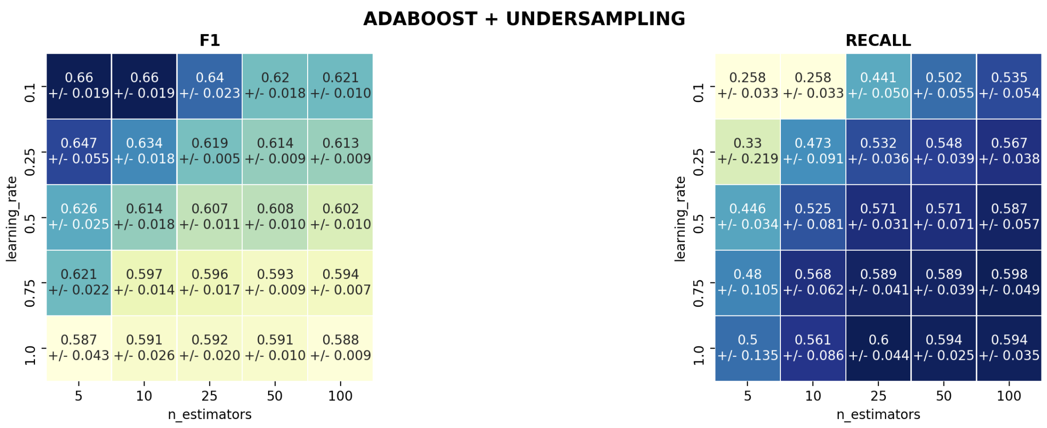


Figure 24: AdaBoost + Undersampling tuning

3.3 Optimized Algorithms : Results

Once discovered the best settings for each algorithm, we trained those algorithms using their best parameters. As far as the input data concerns, we adopted the whole training set. Afterwards, we verified the performance using the given test set. Table 5 highlights that the best approach for this task is Random Forest without any class balancing.

3.4 Decision Tree interpretation

Although the decision tree is not the best method to predict the required variable, its scores are still valid. Note that the following pictures refer to the optimized decision tree that has been trained using a max_depth equal to 2. The most evident point is that the "WheelType" column owns importance that is more than 80%. Hence, a row having the "WheelType" attribute equal to "NULL" (greater than 3.5) is a potentially risky affair.

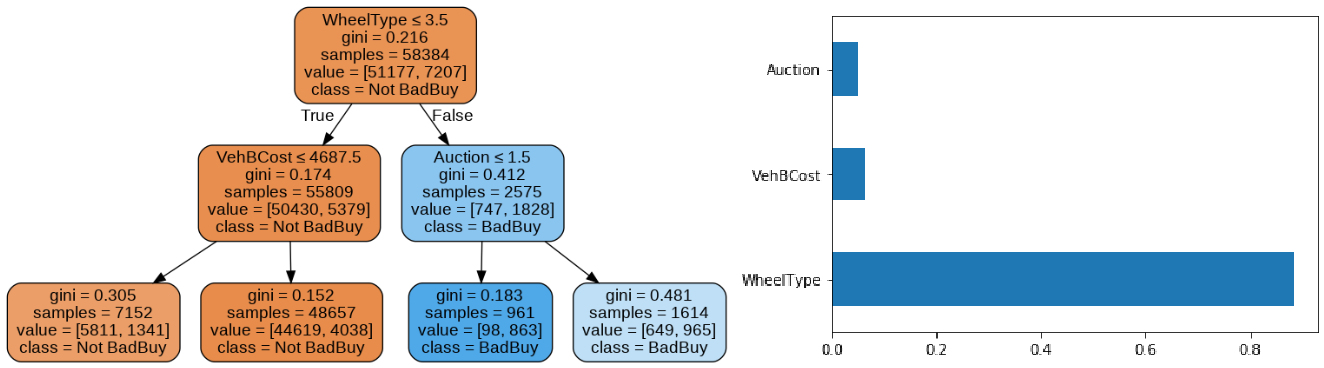


Figure 25: Optimized Decision Tree

4 Pattern Mining

In this section we try to find the best pattern and association rules in order to better understand the information hidden in the dataset.

4.1 Attribute selection and binning

At first, we selected the most informative attributes which are: VehicleAge, Make, Model, Trim, Color, WheelType, VehOdo, AAAP, BYRNO, VNST, VehBCost, WarrantyCost.
We chose to use these attributes because most of them are coherent with the data used in the previous analysis.
In order to perform a better pattern mining, we discretized the numerical attributes 'VehOdo', 'VehBCost', 'WarrantyCost', 'AAAP', 'VehicleAge and then, we transformed numerical attributes into string.
Formerly we plotted their distribution, and in accordance with the distribution we chose to split the attributes into five range.

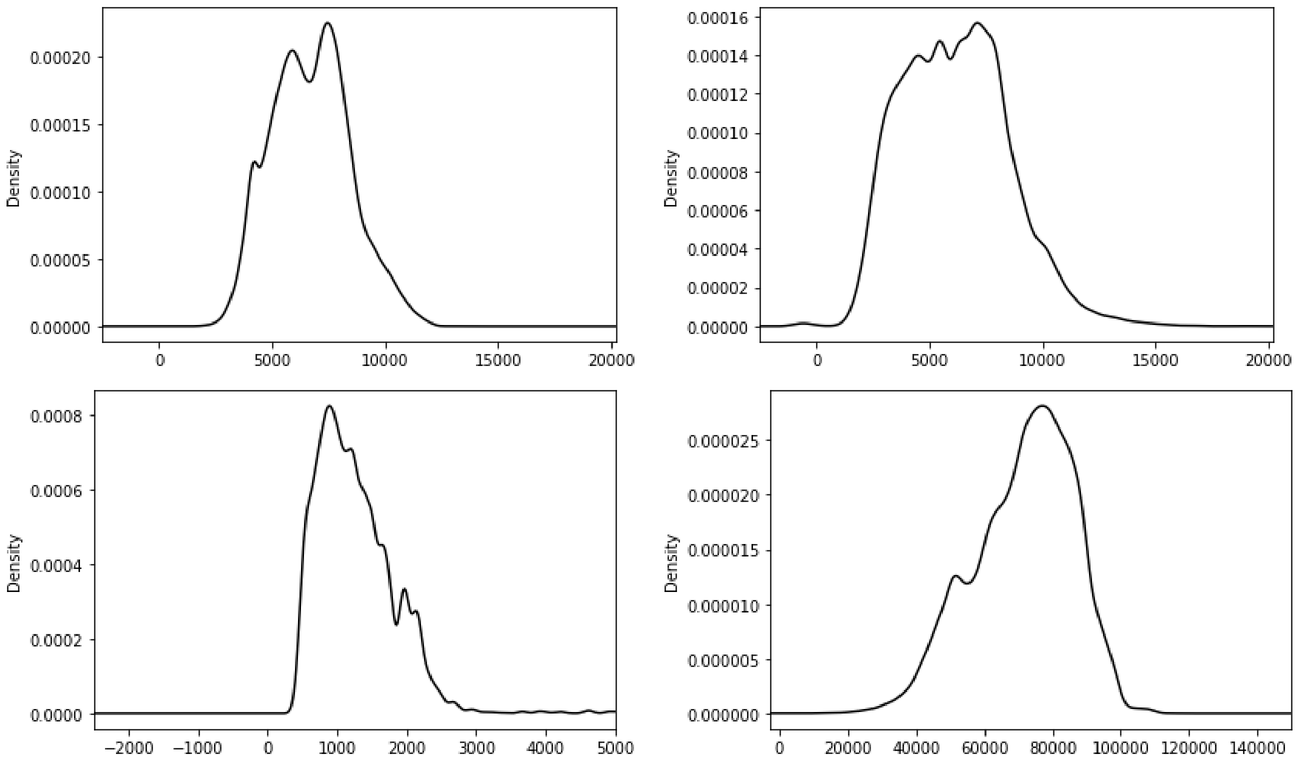


Figure 26: Distribution of numerical attributes

On the top left we bins = [df.VehBCost.min() - 1,5000, 7000, 8000, 10000, df.VehBCost.max()] down left bins = [df.WarrantyCost.min() - 1,700, 1200, 1800, 2600, df.WarrantyCost.max()] top right bins = [df.AAAP.min() - 1, 3000, 6000, 8000, 10000, df.AAAP.max()] right down bins = [df.VehOdo.min() - 1,40000, 60000, 80000, 100000, df.VehOdo.max()]

4.2 Frequent itemsets extraction

We ran the Apriori algorithm for frequent itemsets extraction based on the attributes selected.

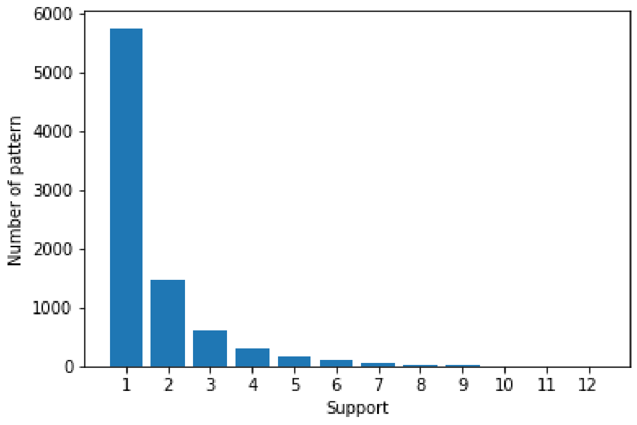


Figure 27: Frequency plof of itemsets with different value of support

4.2.1 Frequent maximal itemset

For frequent maximal itemsets we used a minsupport equals to eight , selecting only sets that have 3 elements.

Pattern	Support
'VehBCost:(5000.0, 7000.0]', 'AAAP:(3000.0, 6000.0]', 'Age:[4, 6)')	6798
'VehBCost:(5000.0, 7000.0]', 'AAAP:(3000.0, 6000.0]', 'Covers')	6653
'AAAP:(3000.0, 6000.0]', 'Age:[4, 6)', 'Covers')	6458
'Age:[4, 6)', 'Odo:(60000, 80000]', 'Alloy')	6392
'AAAP:(3000.0, 6000.0]', 'Age:[4, 6)', 'Odo:(60000, 80000]')	6222
'VehBCost:(5000.0, 7000.0]', 'AAAP:(3000.0, 6000.0]', 'Odo:(60000, 80000]')	5936
'AAAP:(3000.0, 6000.0]', 'Age:[4, 6)', 'Alloy')	5794
'WarrCost:(1200, 1800]', 'Odo:(60000, 80000]', 'Alloy')	5641
'VehBCost:(5000.0, 7000.0]', 'AAAP:(3000.0, 6000.0]', 'Alloy')	5410
'VehBCost:(5000.0, 7000.0]', 'Covers', 'Odo:(60000, 80000]')	5373

Table 6: Number of maximal itemsets with support equals to 8

4.2.2 Frequent closed itemsets

For frequent itemsets we used a minsupport equal to 11%, selecting only sets that have 3 elements

Pattern	Support
'VehBCost:(5000.0, 7000.0]', 'AAAP:(3000.0, 6000.0]', 'Age:[4, 6)'	6798
(('VehBCost:(5000.0, 7000.0]', 'AAAP:(3000.0, 6000.0]', 'Covers'	6653
(('AAAP:(3000.0, 6000.0]', 'Age:[4, 6)', 'Covers'	6458
(('Age:[4, 6)', 'Odo:(60000, 80000]', 'Alloy'	6392
(('AAAP:(3000.0, 6000.0]', 'Age:[4, 6)', 'Odo:(60000, 80000]'	6222
(('VehBCost:(5000.0, 7000.0]', 'AAAP:(3000.0, 6000.0]', 'Odo:(60000, 80000]'	5936

Table 7: Number of closed itemsets with support equals to 10

4.2.3 Frequent itemsets

For frequent itemsets we used a minsupport equal to 30% ,selecting only sets that have 3 elements. The list of the frequent itemsets is largely the same with the closed ones, because the it is a superset of the latter one. (DA RISCRIVERE).

Pattern	Support
'VehBCost:(5000.0, 7000.0]', 'AAAP:(3000.0, 6000.0]', 'Age:[4, 6)')	6798
'VehBCost:(5000.0, 7000.0]', 'AAAP:(3000.0, 6000.0]' , 'Covers'	6653
'AAAP:(3000.0, 6000.0]', 'Age:[4, 6)', 'Covers'	6458)

Table 8: Number of maximal itemsets with support equals to 11

4.3 Association Rules

In this section we discuss the most interesting association rules obtained with the Apriori algorithm. We decided to extract these rules considering only the frequent itemsets with a length major or equal to 3. The association rules are shown in the following tables.

4.3.1 Association Rules for missing value replacement

Given the frequent itemetes founds in Section, we decided to use the rules that best represent car's auction.
For this reason, we use the rules characterized by greater support and best lift value.

RHS	LHS	Confidence	Lift
Alloy	FORD	0.767	1.523
Covers	Age:[2, 4)', 'VehBCost:(5000.0, 7000.0]	0.687	1.501
Alloy	('Odo:(80000, 100000]', 'WarrCost:(1200, 1800]')	0.678	1.346

Table 9: Most significant association rules for missing value replacement

4.3.2 Association Rules for predicting target variable

To predict the target variable we decided to generate a new data set in order to balance the number of 'badbuy' and 'goodbuy' with the aim to produce not only association rules for 'goodbuy', which are the majority of attributes, but also for 'badbuy'.
We selected the most significant rules by using support equals to 10 and confidence of 0.67 and then, we selected rules with a higher confidence and lift as we can see in the Table 8.
By using these rules we try to predict the target variable in the data set obtaining the 83% of accuracy.

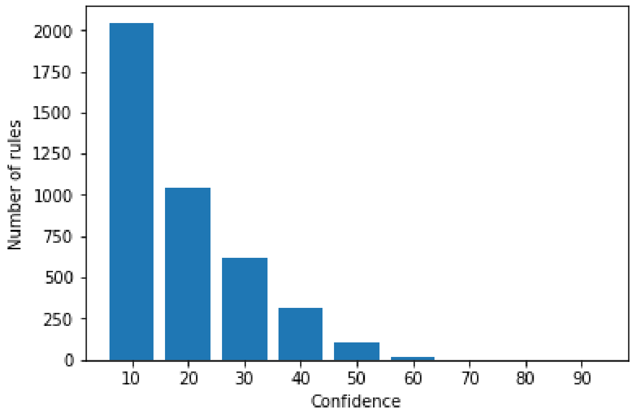


Figure 28: Frequency plof of rules with different value of confidence

RHS	LHS	Confidence	Lift
GoodBuy	AAAP:(6000.0, 8000.0], Covers'	0.701	1.402
GoodBuy	Age:[2, 4), Covers'	0.686	1.373
GoodBuy	CHRYSLER, Covers	0.647	1.295
BadBuy	VehBCost:(224.0, 5000.0], AAAP:(3000.0, 6000.0]	0.613	1.226

Table 10: Most significant association rules for prediction the target variable

		Predicted Value	
		GoodBuy	BadBuy
Actual Value	GoodBuy	47548	3629
	BadBuy	6295	912

Table 11: Confusion matrix obtained from the application of the association rulss for the target variable prediction

5 Conclusion

Chi ha tolto il caffééééééééé???