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 dataset contains information about car auctions occurred in the US between 2009 and 2010 by the company Carvana. In this section, we'll explain how we analysed the data and how we assessed the quality of them. Furthermore, we'll show how we fixed the problem of the missing values (both invalid and misleading).

1.1 Data Semantics

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

- *RefId*: a unique number ID assigned to each vehicle;
- *Auction*: a categorical attribute which defines three different auction providers (Manheim, Adesa, other);
- *Make*: a categorical attribute specifying the name of the company producing that vehicle;
- *VehOdo*: a numerical discrete attribute that points out the mileage;
- *VehBCost*: a continuous numerical attribute indicating the actual final auction price;
- *MMRA*: four numerical attributes that define the acquisition price of each vehicle:
  1. price for this vehicle in average condition at the time of purchase;
  2. price for this vehicle in the above Average condition at the time of purchase;
  3. price for this vehicle in the retail market in average condition at the time of purchase;
  4. price for this vehicle in the retail market in above average condition at the time of purchase;
- *MMRC*: four numerical attributes that define the current price of each vehicle:
  1. price for this vehicle in average condition as of the current day;
  2. price for this vehicle in the above condition as of the current day;
  3. price for this vehicle in the retail market in average condition as of the current day;
  4. price for this vehicle in the retail market in above average condition as of the current day;
- *IsBadBuy*: the target variable. It states if the car was a good deal or not;

Finally, by analyzing all the attributes, we noticed that *Acquisition Type* and *Kickdate* are not present in the dataset, although they are declared in the dictionary.

1.2 Variables transformation and elimination of redundant variables

We opted to specialize the attributes *Model* and *SubModel* because they contained too much information. As an example, we found a row having the *Model* attribute set as follows: EQUINOX FWD V6 3.4L.

It's quite clear that the dataset needs a reorganization because, as it is, it does not even respect the first normal form. Hence, we reorganised this information into five different attributes as explained hereafter:

- *EngineLiters*: this information was both in *Model* and *SubModel* column, usually in the form of a float number and an L character (i.e. 2.7L);
- *NumCylinders*: this information was taken from both variables (i.e. V4, V-8, I4...);
- *WheelDrive*: wheel drive configuration (i.e. 2WD, 4WD...);
- *4X4*: four-wheel-drive derived from *WheelDrive* (4X4 means 4WD);
- *NumDoors*: the number of the doors in a car (i.e. 5D), extracted by *SubModel*;

The information contained in these new attributes is not very interesting in itself, but thanks to this approach, we were able to "clean" both columns from their meta-information. As an example, the Model column passed from about 1000 unique models to just around 200 models without any loss of information.

Afterwards, we also created four different attributes to separate the information contained in the variable *PurchDate*. In detail, we created: *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.

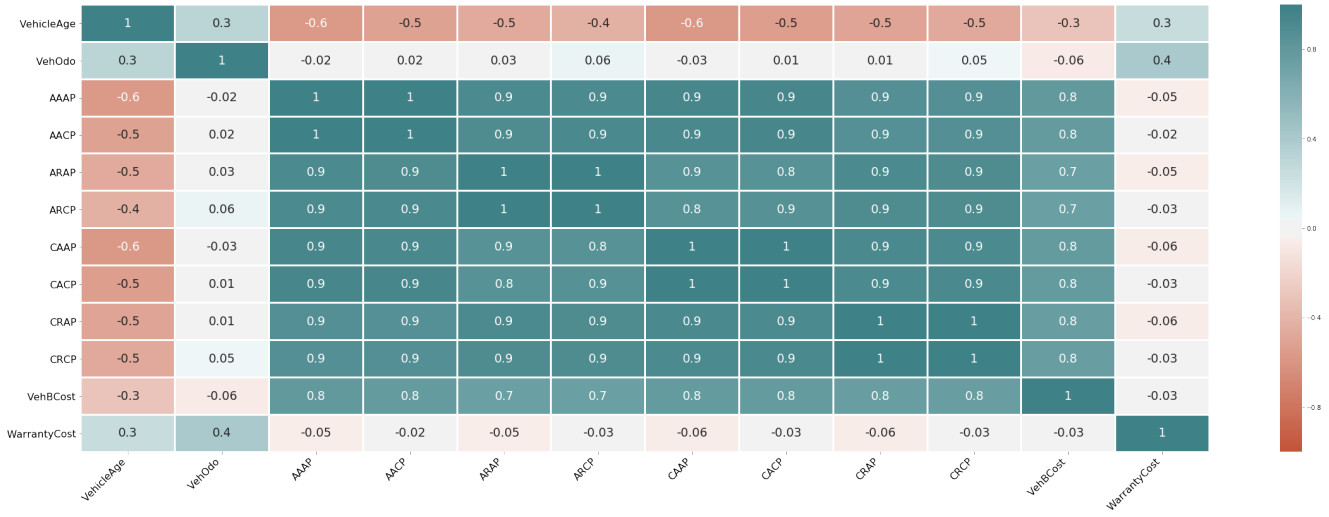


Figure 1: Correlation matrix

The correlation matrix in Figure 1 reveals that all the *MMRs* are strongly correlated (close to 0.9). Consequently, we chose to squeeze them using the PCA technique into two attributes called *PCA1* and *PCA2*. The first one became the most correlated to the original prices, whereas the second one was principally expressing the variance between the observations. For the sake of clarity, we used *PCA1* and *PCA2* only in Section 3, while dealing with classification.

Do note that during the pattern mining and the clustering phases, we reintroduced *MMRAcquisitionAuctionAveragePrice* (or *AAAP*), because we needed to have clearer semantics regarding car prices.
The discussion regarding the distribution of *MMR* attributes and *PCA* is delayed to Section 1.4.

1.3 Assessing data quality

To assess the quality of the data, we performed some variable wise consistency checks (for example on *VehYear, VehicleAge, VehOdo, WarrantyCost*), but we did not find any notable error. We noticed in the dataset some values with inconsistent naming (for example a pair of Japanese cars labelled as American or one model named in two different ways), and we manually fixed those errors. Nevertheless, we found a lot of 0 values for the 8 *MMR* prices (in addition to their 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.

As far as the outliers are concerned, 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 expensive cars, but we could not label these cars as outliers, as we thought that this behaviour is to be expected.

By looking at Figure 2 it is possible to recognise the missing values of our dataset (identified by the white lines). The attributes *PRIMEUNIT* and *AUCGUART* have too many missing values, thus we decide to discard them.

With the other variables, we chose to handle the replacement in different ways depending on the meaning of each one.
Some variables had very few missing values (*SubModel, Color, Transmission, Nationality, Size, TopThreeAmericanName* all had $\sim$ 10 missing values), therefore we replaced them using the mode. For instance, we decided to replace missing colours with the mode over groups of cars with the same model, or the transmission with the mode of the whole dataset (as almost all the cars have an automatic transmission). However, we were able to inspect manually the attributes *Nationality* and *TopThreeAmericanNames* and fix manually those values by just checking the car *Make*.

The main columns with missing values were the 8 *MMR* prices, *WheelType, WheelTypeID* and *Trim*. We handled those cases with the following approaches:

- *Trim* ($\sim$ 2500 missing): we treated *Trim* like the attributes discussed before. Particularly, we adopted the most frequent *Trim* for a given model (or Make when given information was unavailable). In case both pieces of information were missing, we used *bas* as it is the mode over all the dataset.

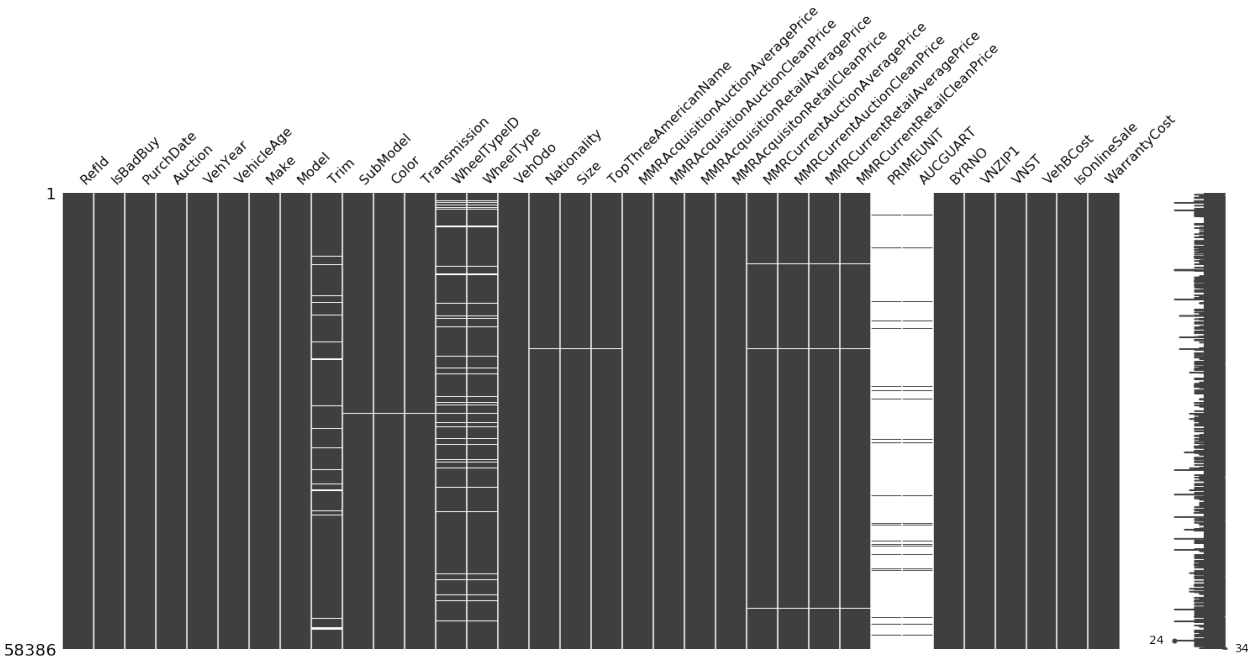


Figure 2: Missing values

- *WheelType* ($\sim$ 3500 missing): We noticed that if the value of *WheelTypeID* is missing, then the chance to be a bad buy highly increases (discussed in Section 1.4, and shown in Figure 7). Assuming that this behaviour is meaningful, we chose to fill missing values with *Unknown* and so adding a category to the attribute. Conclusively, the possible values for the *WheelType* attribute became *Alloy*, *Covers*, *Special* and *Unknown*. We instead decided to drop *WheelTypeID* column as it is redundant.
- *MMR prices* ($\sim$ 500 per attribute missing): we decided to handle entries with **price** = 0 as if they were missing values. After that, we imputed those values using MICE (Multiple Imputation by Chained Equations) [1]. The new distribution is almost identical to the previous one, but of course, there is no more a peak of zeros.

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 (before imputation) which point out the different prices of Vehicle as we explained in the first part of Data Semantics (Section 1.1).

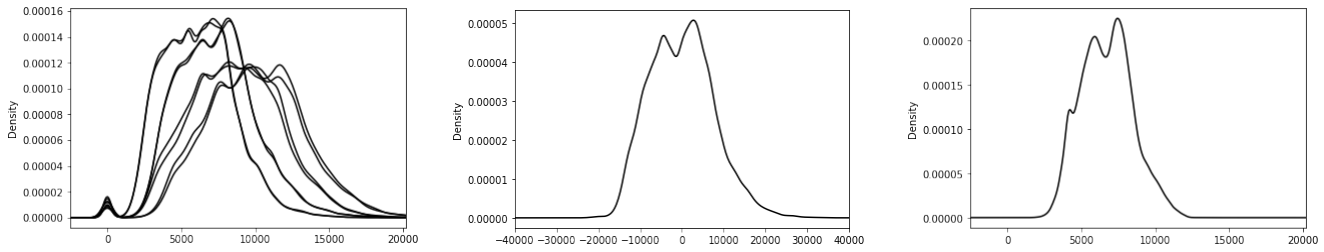


Figure 3: Distribution (from left to right) of 8 MMR prices, PCA1 and VehBCost

We found that MMR attributes are very correlated, but the correlation is even higher when considering those prices pairwise (when considering the clean price and average price together).

Furthermore, we noticed before the cleaning, the presence of a very high peak corresponding to the 0. This peak is no longer present in the variable *PCA1* because, during the phase of imputation, we treated these zeros as missing values. Investigating 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.

[1] Azur MJ, Stuart EA, Frangakis C, Leaf PJ. Multiple imputation by chained equations: what is it and how does it work?

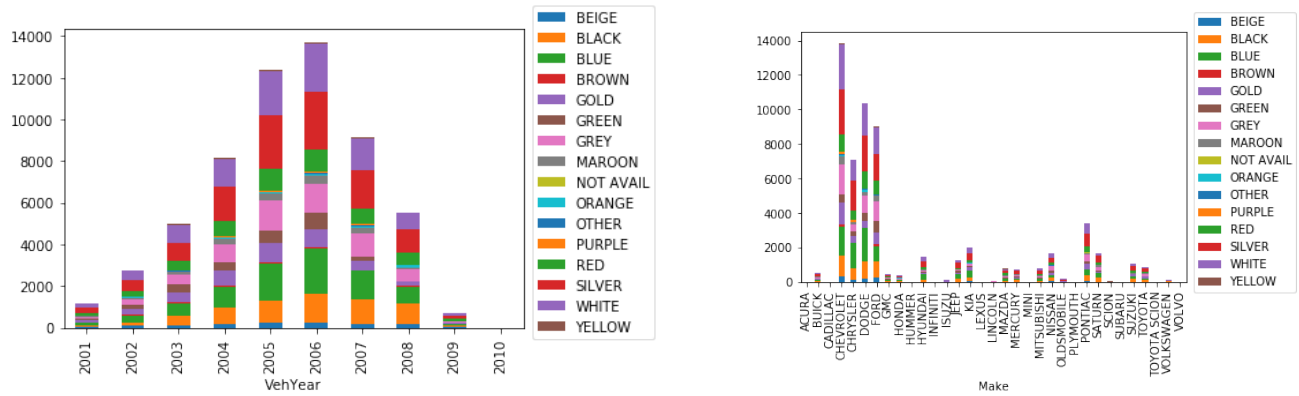


Figure 6: Color attribute plotted w.r.t vehicle year and make

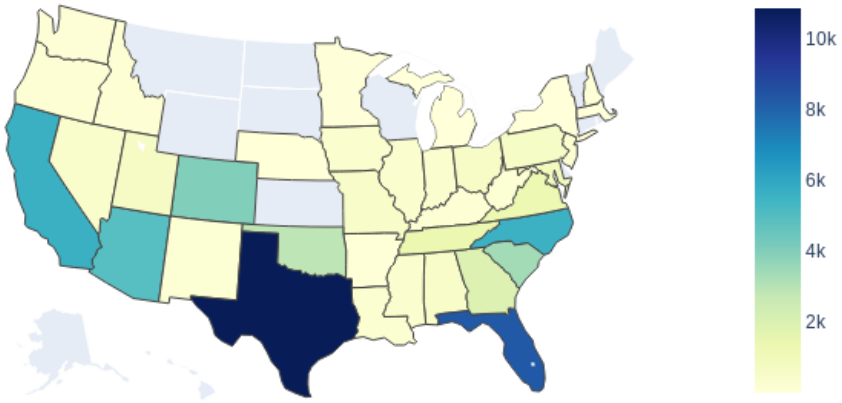


Figure 4: Distribution of the attribute VSNT

To improve the understanding of the attribute *VSNT*, we plotted this variable on the map of the United States. The figure is suggesting that Texas is the state with most of the auctions (18800), followed by Florida (8317) and California (5673). On the other hand, the state with the lower number of auctions is New York (4). Interestingly enough, there are no auctions in Montana, Wyoming, North Dakota, South Dakota, Kansas, Wisconsin, Maine, Vermont, Rhode Island and Connecticut.

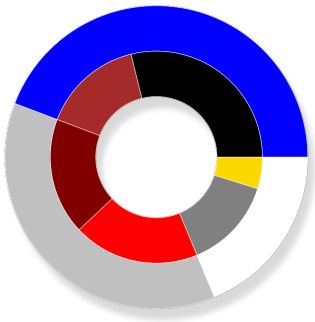


Figure 5: Distribution of the attribute Color

As far as the attribute *Color* concerns (Figure 5), it is not surprising that the most common colours are blue, silver and white. We also tried to plot colours along with other categorical attributes like *VehicleYear* or *Make*, but unfortunately, as Figure 6 shows, we discovered that this attribute is independent of the others, in fact, its distribution does not change almost at all.

Afterwards, we inspected the variable *IsBadBuy*. In Figure 7 (on the left), we first put *WarrantyCost* and

VehOdo in relationship. The most interesting point was that bad purchases are more frequent in less dense areas. Whereas when we dealt with *WheelType* (on the right), we noticed that the number of bad buys is huge whenever the information is lacking (corresponding to the column labelled as *Unknown*).

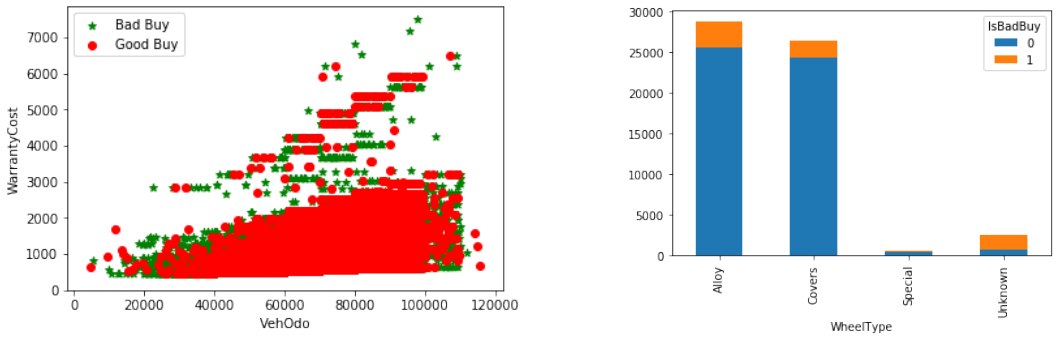


Figure 7: IsBadBuy distribution with respect to VehOdo and VehBCost

Next, in Figure 8, we emphasise that each car manufacturer belongs to a precise price range and, more importantly, the warranty costs change a lot among them. Note that the size of the circles describes the number of cars belonging to that manufacturer. We understood that the *Top Three American Names* (namely Ford, General Motors and Chrysler) hold the great majority of sales. Their cost is usually close to the average, but the warranty cost is pretty high compared to Japanese cars. Furthermore, Hummer is by far the most costly car, while Suzuki has the best quality/price ratio.

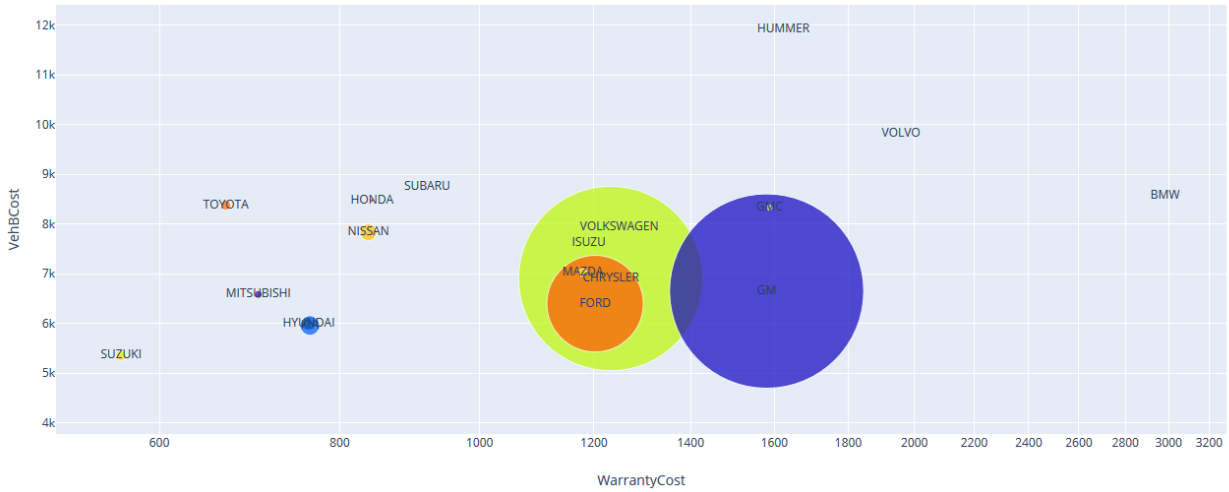


Figure 8: Distribution of *Make* w.r.t the cost and the warranty cost. The diameter of each circle represents the frequency of that make in the dataset.

2 Clustering

In this section, we describe the three Clustering algorithms applied to the data set (KMeans, DBScan and Hierarchical), and their 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 chose to use the MinMax one. In DBScan and Hierarchical, instead, we adopted the Standard one because it gave us slightly better results

2.2 KMeans

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

2.2.1 Attributes' selection

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

We built 5 Dataframes with these attributes 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

2.2.2 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 ∈ [2,16]. The best SSE was obtained in the Data Frame 3, which was originally chosen for its high 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.

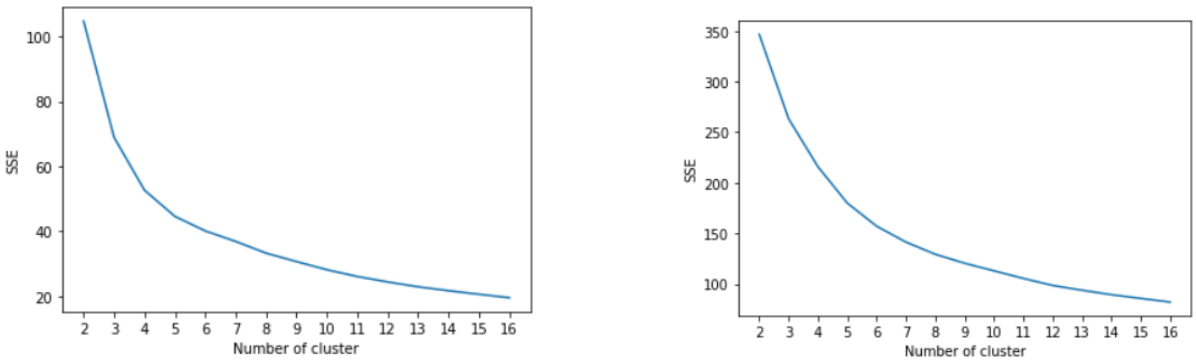


Figure 9: Plot SSE of Attribute set 3 (left) and 4 (right).

We noticed that the SSE curves for all the data frames share a strong similarity (same curvature, but different SSE). Indeed their behaviours are very close to the ones shown in Figure 9. Then we discovered ,using the *knee method*, that the best value for all the data frames was $k = 6$. A comprehensive analysis of all the data frames is 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 essentially groups of cars in different price ranges. We decided that the best clustering, both semantically and parameter wise, was DF4 (the 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

2.2.3 Description of the best clustering

The following descriptions refer to the results of the clustering that were proposed as the best in the 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:

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

In Figure 10 the clustering results are shown plotted in 2 dimensions (*WarrantyCost* and *VehOdo*). The plot does not show the third dimension (*VehBCost*) because this latter is the least important feature (as it will be evident from the centroids shown when describing the clusters in Section 2.2.3). The figure also shows the cluster centroids with a star on the plot.

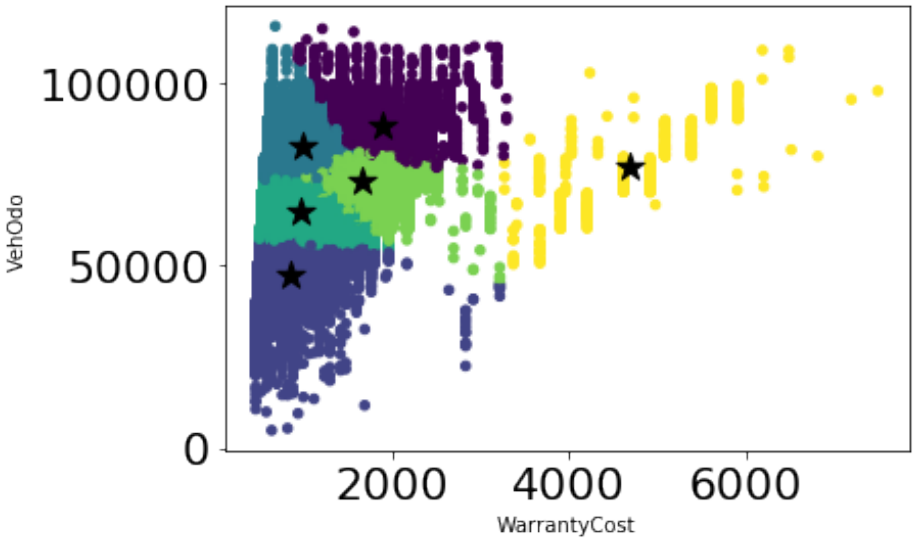


Figure 10: Clusters plotted with WarrantyCost and VehOdo

The clusters descriptions for Figure 10 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 most interesting clusters, 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 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.

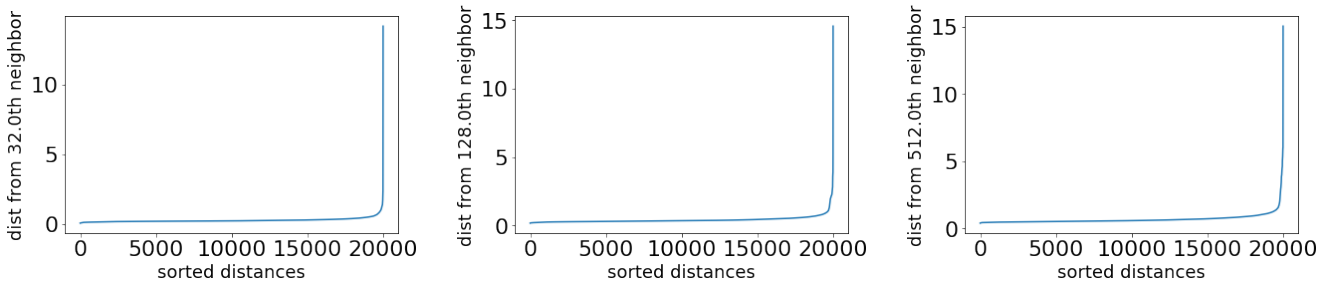


Figure 12: k-th neighbour distance, with k = 32, k = 128, k = 512

Given these outcomes, we tried to understand if bad buys were located mainly in one of those clusters. By plotting this information in Figure 11, we noticed that cluster 2 (new cars with low odometer reading), has the least amount of bad buys.

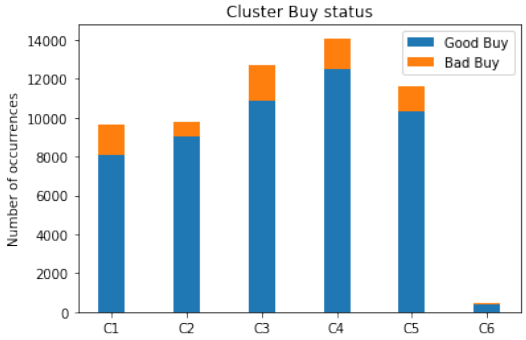


Figure 11: Distribution of IsBadBuy with respect to the 6 clusters found

2.3 DB Scan

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

2.3.1 Attributes and distance function

We decided to follow the same reasoning used for *KMeans*, thus we attempted to do the clustering over the same set of attributes. We also chose to use Euclidean distance, and Z-Score scaling.

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.

2.3.2 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 12, were used to select the right epsilon for attempting the clustering with DB-Scan.

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 2.3.3, 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.

2.3.3 Characterization and interpretation of the obtained clusters

First, we are going to analyze the results with parameters shown in Table 2. The algorithm produced as a result a single cluster containing all the points in the data set, with the exception of ~ 100 elements, which were labelled as noise points. The main reason was that the data forms one big cloud of points, with different density distributions 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.

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

We then decided to try and find the densest areas in the data set, by manually checking a lot of parameters configurations. This approach, however, does not find any 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 13.

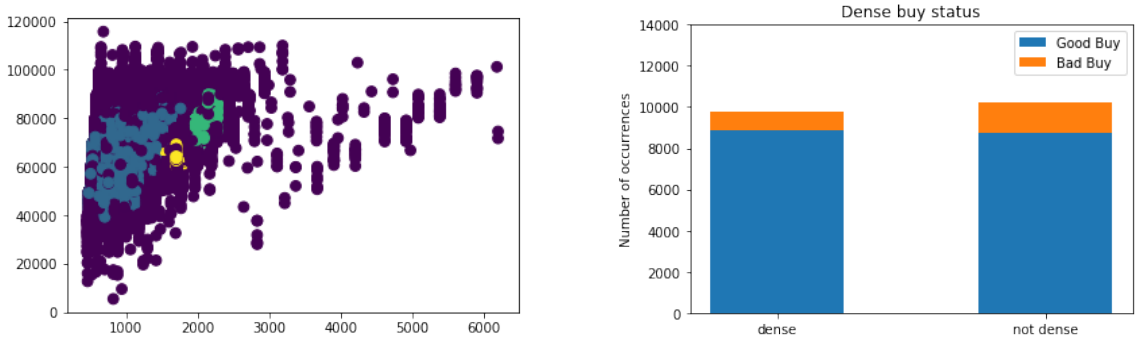


Figure 13: DBScan clustering results with **minpoints** = 256 and $\epsilon = 0.38$. Purple colors are noise points

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

2.4 Hierarchical Clustering

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

2.4.1 Attribute Choices

We decided to perform clustering on the following attributes set:

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

2.4.2 Algorithms and Dendrograms

We decided to perform hierarchical 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, being it 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 14 shows the silhouettes for the results found with all the algorithms on data frame 2 (the same used for KMeans and DBScan). 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 15, while their respective dendrograms are shown in Figure 16. All the other clustering attempts produced highly imbalanced cluster (one main cluster and some single digit size clusters).

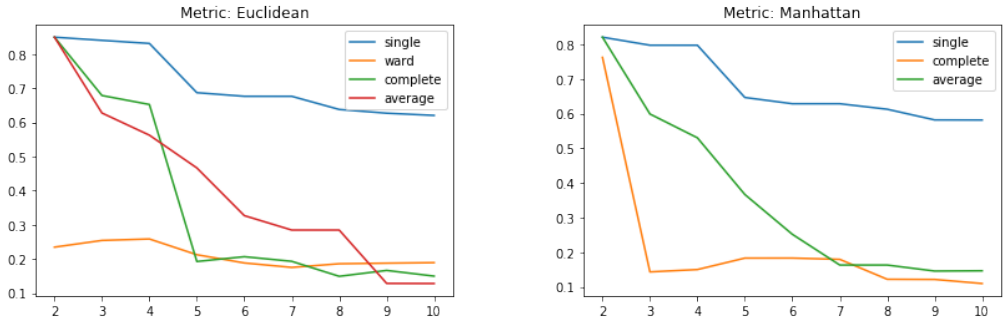


Figure 14: Silhouettes for all algorithms and all metrics on data frame 2

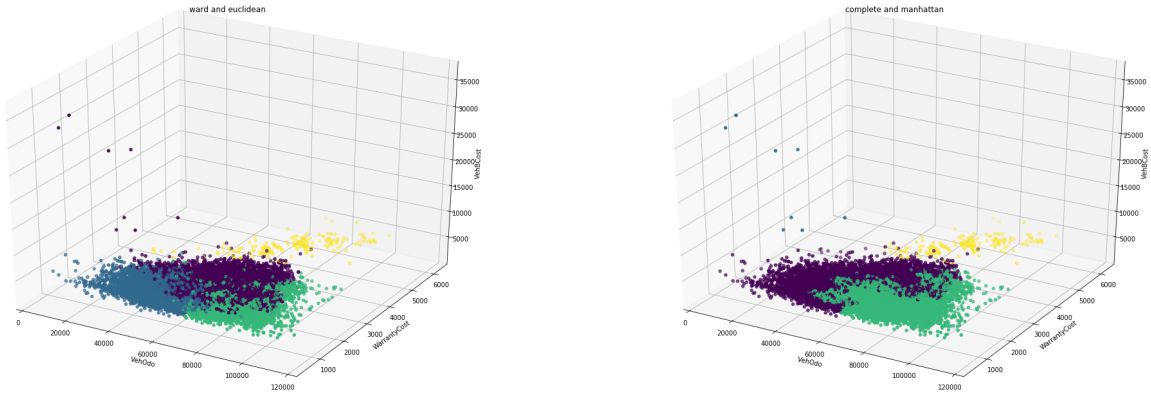


Figure 15: Hierarchical clustering results, number of clusters is 4

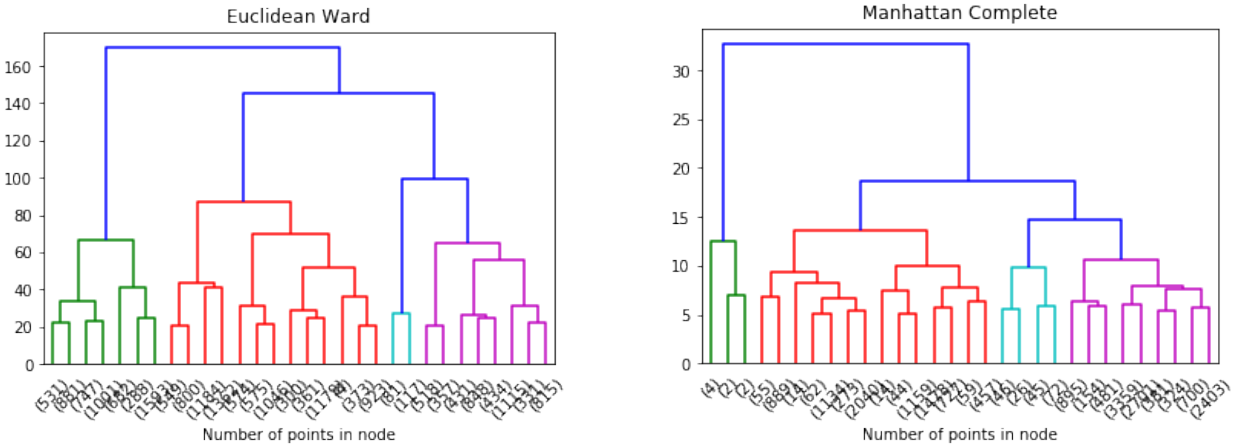


Figure 16: Dendrograms plotted with *lastp* truncate mode

2.4.3 Best clustering approach and comparison of the clustering obtained

In conclusion, the best clustering results were found in the circumstances shown in Section 2.4.2. 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 15 and in Figure 10 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 2.2.3 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

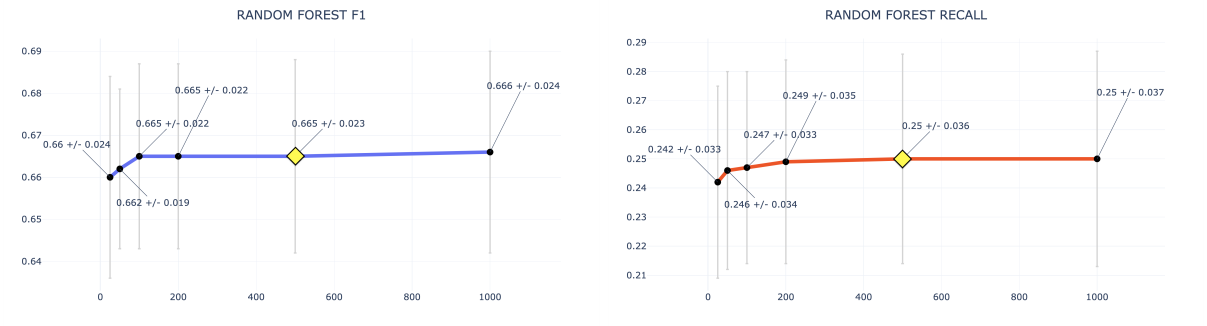
3.1.1 Methodology

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 final 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



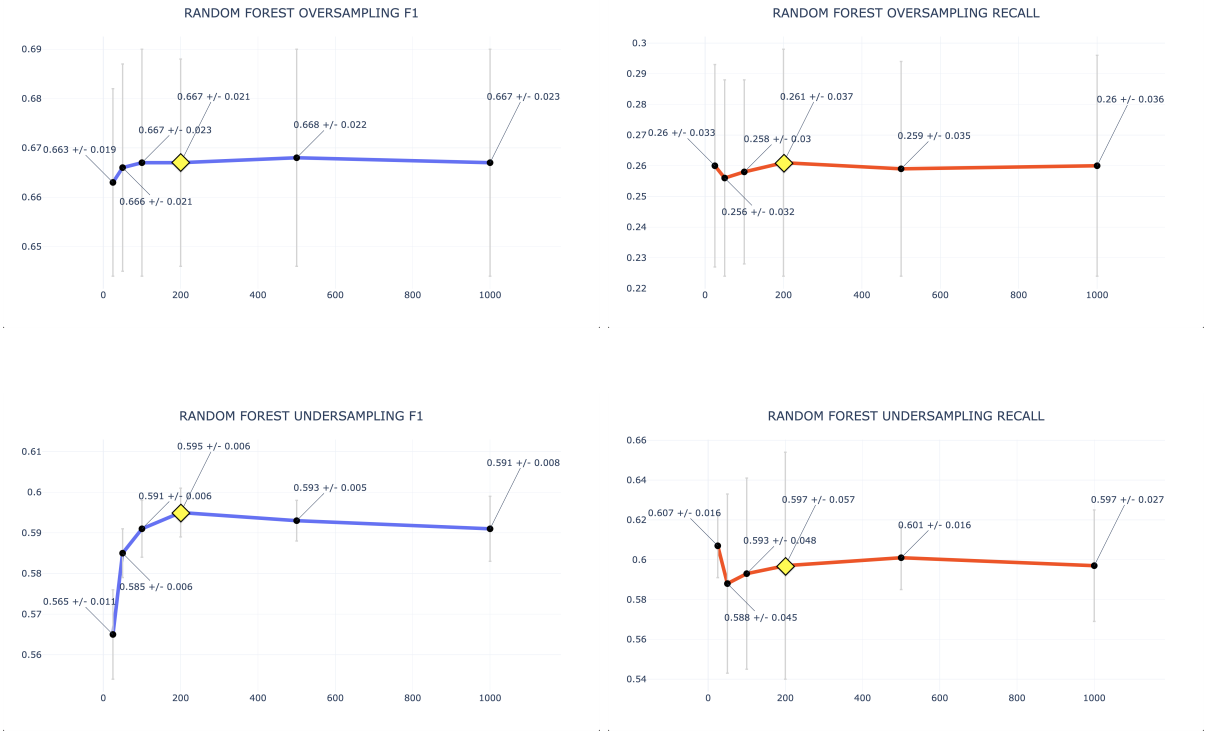
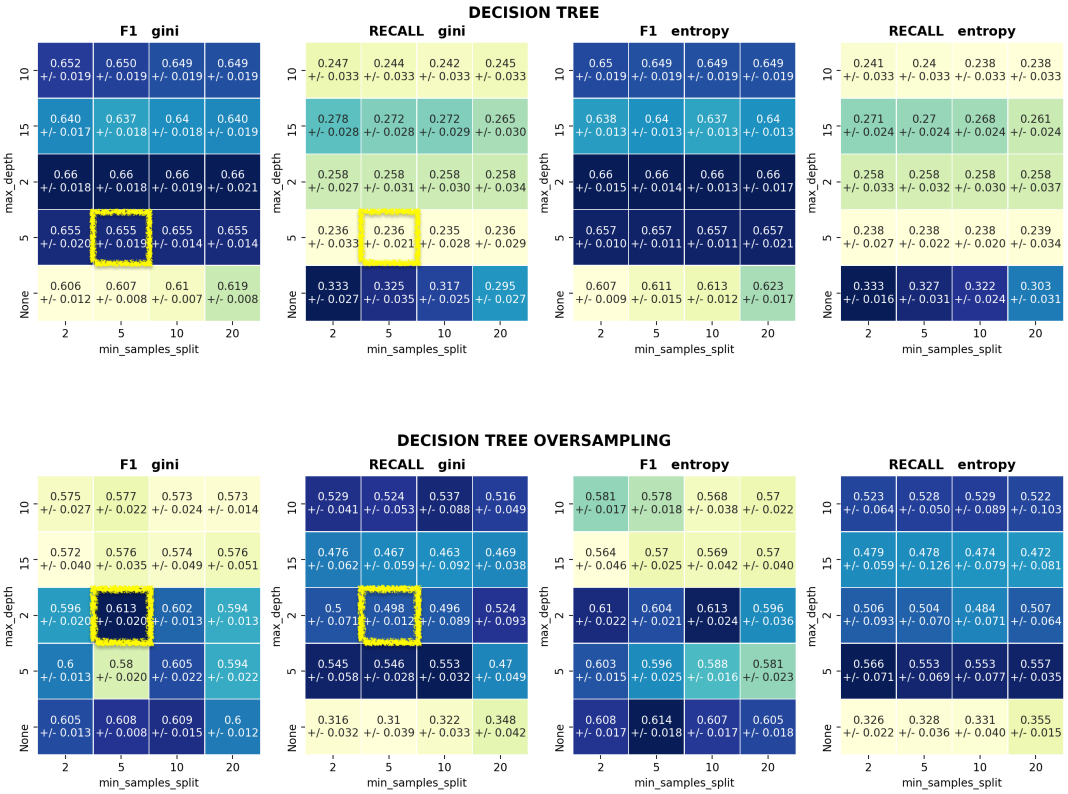


Figure 17: Random Forest tuning

3.2.2 Decision Tree



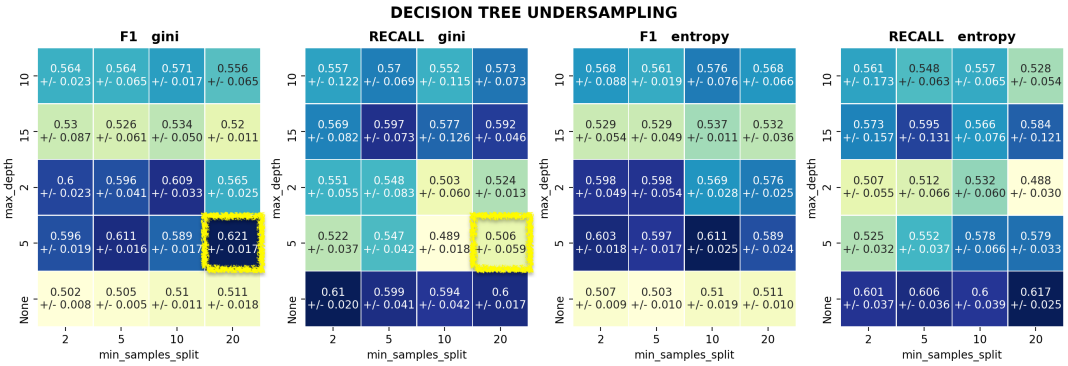


Figure 18: Decision Tree tuning

3.2.3 AdaBoost

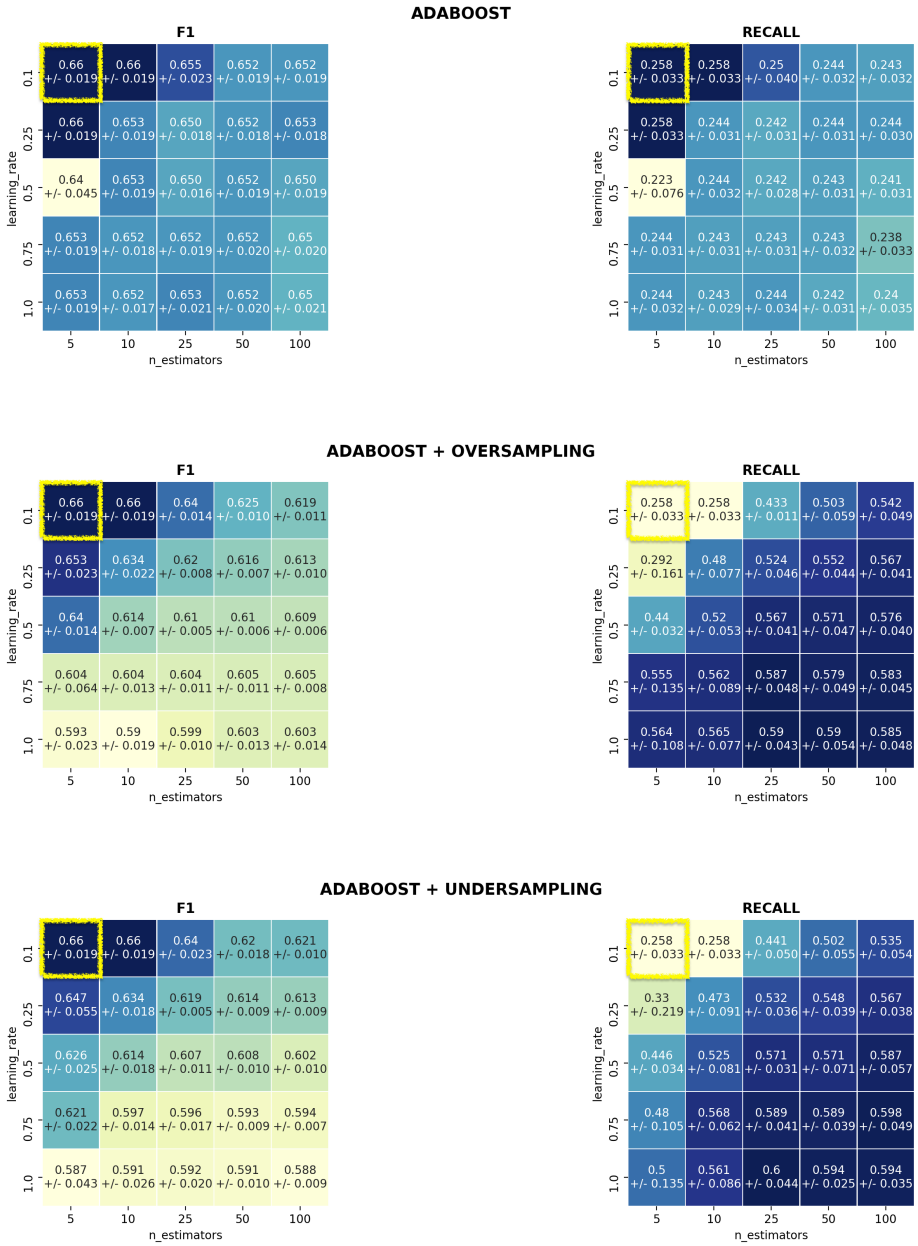


Figure 19: AdaBoost tuning

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.

	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	**0.90**	0.90	0.85	**0.87**	0.99	0.21	**0.60**	0.95	0.34	**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

3.4 Optimized 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 five (here we're showing only four levels due to lack of space).

Surprisingly enough, the *WheelType* column owns importance that is more than 70%. Hence, a row having the *WheelType* attribute equal to "NULL" (greater than 3.5) is a potentially risky affair. The second leading variable is the type of auction. In particular, cars sold through "ADESA" (smaller than 1.5) auctions point entirely towards "Bad Buy" leaves. Furthermore, the Gini coefficient (0.412) suggests that this split is almost useless because the preponderant feature is still the previous one. As far as the red branch concerned, there is just one blue leaf. This latter represents the category of cars with a final price above 12260$ and age less than 4.5.

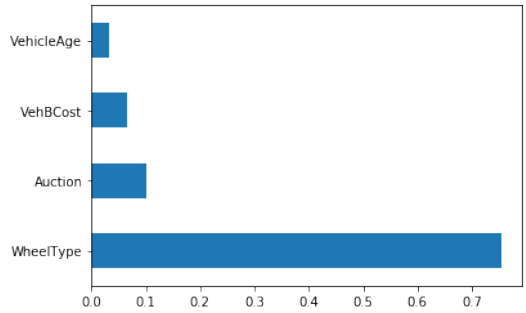


Figure 20: Optimized Decision Tree Features Importance

3.5 Optimized Random Forest: Interpretation

AAAAA

AAAAA

AAAAA

AAAAA

AAAAA

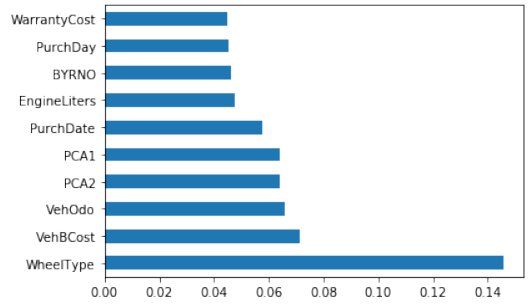


Figure 22: Optimized Random Forest Features Importance

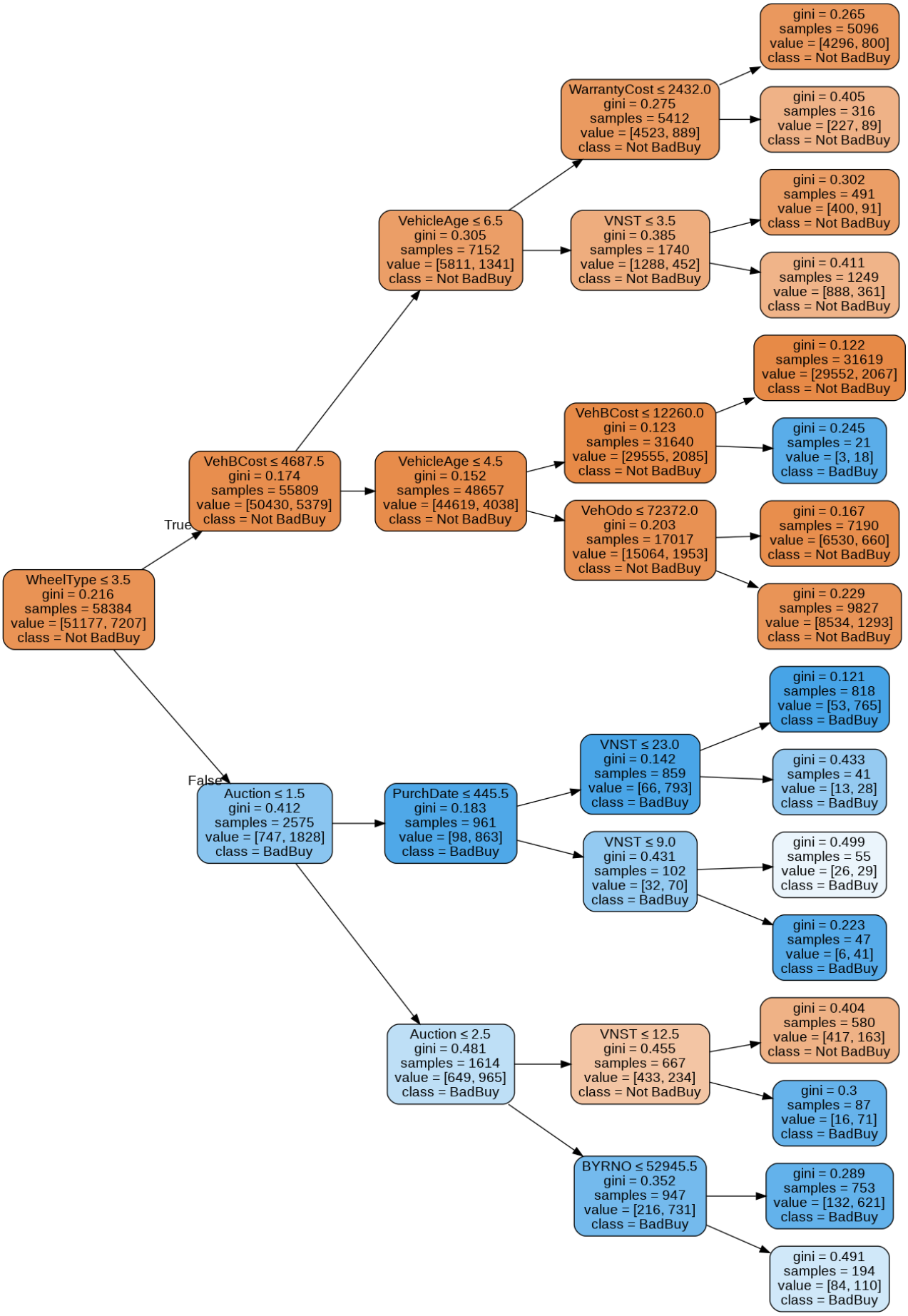


Figure 21: Optimized Decision Tree Schema

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.
To perform this task, we discretized the numerical attributes *VehOdo, VehBCost, WarrantyCost, AAAP, VehicleAge*.
Formerly, we plotted their distribution, and, following the distribution, we chose to split the attributes into five ranges.

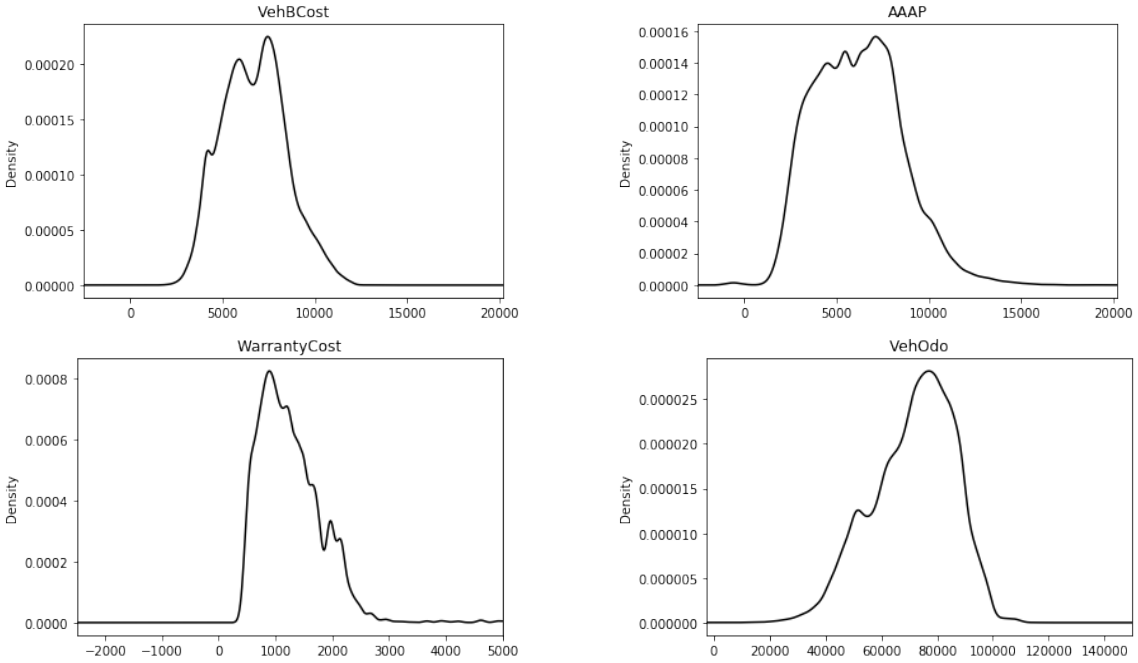


Figure 23: Distribution of numerical attributes

In the Figure 23, starting from the left side, we have the distribution of the attribute *VehBCost* on the top (1,5000, 7000, 8000, 10000), and the distribution of *WarrantyCost* on the bottom (1,700, 1200, 1800, 2600). To the right side, the Figure shows the distribution of *AAAP* on the top (1, 3000, 6000, 8000, 10000) and lastly, the attribute *VehOdo* on the bottom (1,40000, 60000, 80000, 100000).

4.2 Frequent itemsets extraction

We ran the Apriori algorithm for frequent itemsets extraction based on the attributes that produced the best results in the previous analysis. Furthermore, we decided to drop some attributes like *Transmission, Nationality* and *SubModel*, because they produced patterns which were too generic and, as a consequence, the results were not meaningful. This behaviour depends on the strong imbalance between the two classes.
Subsequently, we try to extract frequent patterns with different values of support by using frequent, maximal and closed types.
After several attempts, we agreed that the best min_support was between 9% and 11%, selecting only sets that have three elements. Nevertheless, probably for the low amount of attributes that we used for this task, even though we used different types (*frequent*, *maximal*, and *closest*) of support, the results produced were the same, which are shown in Table **??**.
We noticed that the number of patterns changes substantially by increasing support. Indeed, by using support of

2% we obtained 1467 patterns, a number which is three times lower than the previous one (5759 with support 1%). The distribution of patterns is represented in a *logarithmic scale* in Figure 22.

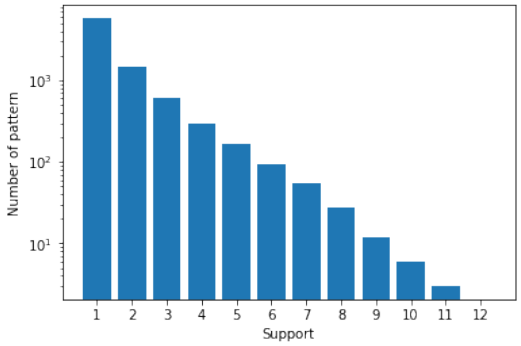


Figure 24: Frequency plot of itemsets with different support values represented on a **logarithmic scale**.

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
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
AAAP:(3000.0, 6000.0], Odo:(60000, 80000], Alloy	5306
WarrCost:(700, 1200], Odo:(60000, 80000], Alloy	5266

Table 6: Pattern of different types with support between 9% and 11%

From the generated itemsets we can see that cars which are not new but neither old, are usually sold at a slightly more expensive price compared to the average price at the auction (Pattern 1 in the Table, which is the one with highest frequency). The mileage on the vehicle is usually between 60000 and 80000 (Pattern 5) for cars that have an average price between 3000.0 and 6000.0 and age between 4 and 6.
The attributes in these patterns are very meaningful since they are the same that give us the best results in clustering analysis section.

4.3 Association Rules

In this section, we discuss the most interesting association rules obtained with the Apriori algorithm. We decided to attempt extracting these rules considering support up to 100%, length of maximum three elements and confidence lower or equal than 70%. We show in Figure 23 how the number of rules varies by varying the confidence value (confidence is between 65% and 67% in the plot). We also plotted, on the right, a scatter plot showing confidence, support and lift. We can notice that the number of rules with **support** >= 10 is very low with respect to the whole amount.

We found optimal results with **support** = 11 and **confidence** $\approx$ 67%. The association rules found with those optimal parameters are shown in Table 7.

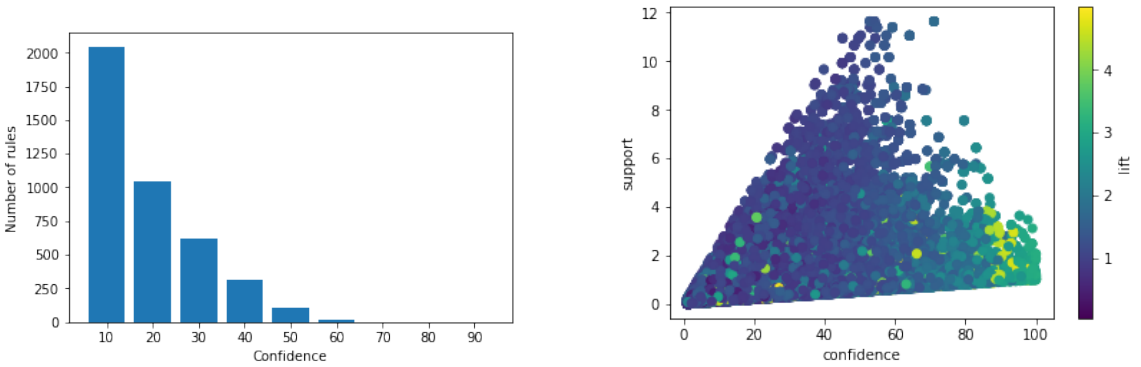


Figure 25: On the lefthand side, frequency plot of rules by varying the confidence value. On the righthand side, a scatter plot of support, lift and confidence

RHS	LHS	Confidence	Lift
GM	LS	0.768	2.273
GM	WarrCost:(1800, 2600]	0.736	2.177
CHRYSLER	Odo:(40000, 60000], WarrCost:(700, 1200]	0.689	2.149
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 7: Most significant association rules with optimal parameters

As we can see from Table 7, General Motors vehicles (GM) are usually LS series, with the highest confidence and lift. Among the American car makes, Chrysler is the one that has the lowest average warranty cost as we can also see in Figure 8.

4.3.1 Association Rules for missing value replacement

After performing data Preparation for pattern mining, we decided to attempt replacing the missing value of the attribute *WheelType*, the only one which contains missing values.
For a maximum of three elements, we used **support** = 11 and we selected as confidence between 0.67% and 0.76% to find the best rules. Then, we chose the most interesting rules for the most frequent elements within *WheelType*(*Alloy* and *Covers*) and we replaced only the value 'Unknown' by following the rules extracted.
The rules we chose replaced exactly 935 elements out of a total of 2575 missing values, obtaining coverage of 36,31%.

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 8: 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 perform *undersampling* in order to balance the number of 'BadBuy' and 'GoodBuy' for producing 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** = 10 and **confidence** = 0.67 and then, we selected rules with a higher confidence and lift.
We used those rules to predict the target variable, and whenever they did not match any element in the data frame, we decided to predict them as 'GoodBuy', since it represents the majority class.
The result of this process is described in Table 9, whereas the confusion matrix is shown in Table 10. From this table, we compute the accuracy which is 83%.

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 9: Most significant association rules for prediction the target variable

		Predicted Value	
		Yes	**No**
Actual Value	**Yes**	47548	3629
	No	6295	912

Table 10: Confusion matrix obtained from the application of the association rules for the target variable prediction.

If we compare these results with the ones obtained in the Classification (Section 3), we can note that:

- the number of True Positive is similar;
- the number of True Negative is lower;
- the number of False Positive is almost the same;
- the number of False Negative is very high.

Since we use only four rules for predicting the target variable, we can evaluate it as a good result.

5 Conclusion

For the analysis of this car auctions data set, we used different data mining techniques, trying to understand whether a deal is a good transaction or not.
In the *Data Understanding* section, we addressed issues related to data semantics and quality, like fixing missing values and deleting redundant variables. Our main goals were to clean up the *Model* variable (which we believed to be one of the deciding factors for the target variable), to reduce the dimensionality of highly correlated variables and have a basic understanding of variable distributions.
In the *Clustering* section, we concentrated our focus on numeric attributes (prices, odometer and warranty cost) to find groups of similar points in the data set. We found, as an obstacle, the main cluster of points of various densities. We found KMeans and Hierarchical clustering to be the best in performance, and attempted to describe the meaning of given results.
In the *Classification* phase, we tried to predict the value of the target variable, namely *IsBadBuy*. We performed 5-fold cross-validation with various classifiers and configurations, attempting also both *undersampling* and *oversampling*. We optimized the parameters for *F1 score* and obtained $\mathbf{F1}_{avg} = 0.65$ on the test set with the best classifier. The accuracy is very high, but the classifier does not obtain better results because of the recall on the negative class (the minority one).
Lastly, in the *Pattern Mining* phase, we obtained some interesting rules by discarding highly imbalanced categorical attributes. We were able to obtain decent results both when attempting to predict the target variable and when replacing the missing values.