DATA SCIENCE PROJECT

Guilherme Pires | 102132
Miguel Ferreira | 90899
João Cruz | 90731

Dataset1

Data profiling

Data dimensionality

Due to space constraints, all data profiling images are available in the appendix in the end of the document.

We have almost 6568 records with missing values for the **VEHICLE_ID** and 421 for the **PERSON_AGE**. We have **21** variables of which **15** are **Symbolic**, **2** are **Binary** and **4** are **Numeric**, and **45650** records. This is already presenting us with some problems: how to treat the high number of records with missing values (can they be simply dropped, or should we do something else? Regarding the symbolic variables, how are we going to treat them?). Lastly, our target variable has two values: Injured and Killed. On the dataset there are around **45 thousand** records for Injured and around **250** for Killed, making this a very imbalanced dataset. This means that we have roughly 180 times more information about Injuries than deaths.

Moreover, and before continuing, we need to be aware of some incorrect values (not noise nor outliers). For PERSON_AGE, we have 29 records with age > 140 or < 0 (with some > 1000). We dropped all of these values (also they are all from the majority class). Moreover, there's little to no interest in working with features that represent an ID [UNIQUE_ID, VEHICLE_ID, COLLISION_ID, PERSON_ID], this way for the remaining of the work, we won't be working with any of these variables.

Data granularity

Regarding time hierarchy, CRASH_DATE records are grouped into 'Weekday', 'Weekend' and 'Holiday' and CRASH_TIME is grouped into 'Dawn', 'Morning', 'Lunch time', 'Afternoon', 'Dinner time' and 'Night'. Both choices were made after analysing the data distribution for both variables. These changes will give us less granular detail, but we gain in more symbolic information, since we don't believe that having so much detail regarding the specific dates and hours is better or valuable.

Regarding granularity, we need to deal with our symbolic features. For them, we present a simple methodology: For each feature, we create 'major' groups, where, for each group, we attribute a number (depending on, for example, the severity of an injury, or the degree of protection of a safety equipment) and then, for each 'major' group, if we need more detail (with we usually do), we create one more (finer) level, where we also attribute a number depending on some criteria. This way, for each major group, if there are many, we fix a number (10, 20, 30, ...) and then for each finer group, we fix another number (11,12,13... 21,22,23...). This way, we can deal with our symbolic features, treating them as ordinal ones. Obviously this has some problems, however, it presents a good trade-off between information, detail, and usability in our models since most can't handle symbolic data, and, to use the one-hot encoding, would also be extremely challenge since we would have a great number of features. A curious challenge that we found, was, what value to attribute to 'Unknown' or 'Not Applicable'. For each feature, we tried to create, with a given criteria and logic, a 'distance' from these values to the other ones, that sounded good and intuitive. We present a detailed description of the changes made in the appendix.

With these granularity changes, we mostly wanted to enter 'common-sense' knowledge and also give to all symbolic features an order. This order, obviously, is limited, however, and regarding encoding, it may prove to be a better choice than using one-hot encoding.

Feature	POSITION IN VEHICLE	EMOTIONAL STATUS	EJECTION	PED LOCATION	PED ROLE	PERSON TYPE	BODILY INJURY	PED ACTION	SAFETY EQUIPMENT	COMPLAINT	CONTRIBUTING FACTOR 1	CONTRIBUTING FACTOR2
Old unique values	10	8	4	4	5	4	14	16	16	19	46	40
New unique values	5	7	3	4	5	4	10	12	9	7	23	23

Table 1: Symbolic features (excluding CRASH_TIME and CRASH_DATE) granularity changes

Data distribution

Regarding data distribution, for our only numeric variable, PERSON_AGE, we have a normal distribution with an expected range.

Also, for the symbolic features, there is some information that we can retrieve. For one, almost every feature has one value that has a much bigger count than all the others. For instance, for the Ejection feature, almost 35 thousand records are just for one value, 'Not Ejected'. There's not much that we can retrieve from the distribution other than the fact that the majority of the accidents are probably quite identical, supported by the fact that so many features have one value with a much higher weight than the others.

Obviously, this is not to say that we can't retrieve more information. For example, almost half of the records (26 thousand) have, as PED_ROLE, the value DRIVER. This is information that is quite trivial and expected. Roughly 13 thousand have PED_ROLE as Passenger. We also observed that the large majority of the records have as 'Ejection', the value 'Not Ejected'. Even though it is quite good and reassuring to know that the majority of the accidents, the majority of the victims are not 'Ejected' or 'Partially Ejected' (ejection we can assume is something bad!), the reality is that it doesn't give us much information in terms of modeling this particular problem of predicting when do we have an accident with death as an outcome. We can only retrieve high level information, as, for example, what is the most common POSITION_IN_VEHICLE (in this particular context). This information may prove to be helpful when analyzing our classification models and other results in this report.

Data sparsity

Regarding data sparsity, we assess that the data is sparse in the sense that we have 'holes' in the space of the data. Moreover, from the sparsity charts, due to the symbolic to numerical transition we did for the features, there's not much information for us to retrieve. To gather more helpful information, we've

used seaborn stripplot with jitter to better understand the features we have, for both Injured and Killed records. Reinforcing that correlation doesn't mean causation, for Killed, the bodily Injured are mainly on the head, entire body and chest, with contrast to the Injured records where there's no body part that stands out (this way we hope to see this feature with high importance in the next steps). In terms of safety equipment, the Injured records mostly have Belt (probably because they are on a car), but for the Killed records, the most recurrent safety equipment is none, and then a helmet (one can hypothesise that this is because of motorcycles and bicycles). Other relevant information is that for the Killed records, there's two values that stand out on the person type feature: occupant and pedestrian.

Regarding the ejection feature, we can also assess that there is a majority of the Killed records that ejected, contrasting with the Injured ones, with a majority of non-ejection. For the COMPLAINT (🔗) feature there's also a majority on GREEN for the Injured records, while for the Killed records there's a majority for 'Yellow', and GREEN only holds 7% of records. This trivial information is important for us to better assess how the data is distributed between both records - Injured and Killed. This way we can better understand with what we are dealing and also better understand our future results.

Another area of big difference is the EMOTIONAL_STATUS, where, for the Killed records, 65% of the records have the value 'apparent death' and 26% are 'Unconscious'. This finding is of the **upmost importance** because what it is saying is that a majority of the Killed records are defined as having an emotional status of 'apparent death'. In a rough generalisation, this is almost what we are trying to predict! As such **we won't be working with this feature** (that may be classified as a false predictor). For the Injured records the majority of the records (92%) have the value 'Conscious', while for the Killed ones the conscious value only holds for 6% of the records. Actually, for the Injured records only 241 in almost 40 thousand records have EMOTIONAL_STATUS has 'Unconscious' or 'Apparent Death'.

Similar to the EMOTIONAL_STATUS, the COMPLAINT feature takes the value 'Internal' (before we've applied our granularity changes, but even after the distribution is similar) for 56% of the Killed record, while, for the Injured ones, only 1% take it, this means this feature will probably conceive a good decision boundary for algorithms like Decision Trees. Furthermore, for the Injured records almost 62% of the records take the value "Complaint of Pain or Nausea", where for Killed records only 3% take it! Ultimately, this feature could also be classified as a false predictor, however, our reasoning for the EMOTIONAL_STATUS is that it is almost a feature 'after the fact', while the complaint is 'during the fact' (here 'the fact' is an accident), thus **we will work with this feature!** Actually, regarding the correlation values, the feature with highest correlation with the target feature is COMPLAINT with 0.13, versus 0.096 from EMOTIONAL_STATUS, almost 35% more. For all the features that we didn't mention, either there weren't any findings or they weren't relevant.

For the symbolic features there are some correlations that stand out: [PED_ACTION, PED_LOCATION], [EJECTION, PED_LOCATION], [EJECTION, SAFETY_EQUIPMENT], [PED_ROLE, POSITION_IN_VEHICLE]. Most of these correlations, even though they are high, don't give us much information, however one can expect that when doing Feature Selection, some of these features may be dropped for being redundant. More important is probably the correlation between the feature EJECTION and SAFETY_EQUIPMENT.

Regarding correlations with respect to PERSON_INJURY, if we divide the dataset between Injured and Killed, we find a lot of very strong correlations between many of the features for the Killed records, this is particular important because it may point that when doing pattern mining and clustering, we may expect finding good results.

Data preparation

Missing value imputation, dummification and other transformations

From now on, regarding models performance, we just present what we think is necessary, as in a pragmatic choice. In the Figure 2, below, we present the values for NB and KNN just for the test set. However, if and whenever the train results provide any insight, we will also present them on this report.

For the **missing value imputation**, we test two approaches: a custom missing value imputation and the replacement of the missing values with the most common value. Previously we were also doing it with replacement by a constant but it is a 'blind' replacement that disregards the information from the dataset, as such we won't proceed with it.
For the custom missing imputation, we use a simple set of rules:

- Records without person age are removed;
- For each row of the dataset, the rules in figure 1 are used.
- Some additional transformations (🔗) were also made. Regardless of the value, if it empty, or, not, if the person type is pedestrian, then the position in vehicle is set as 'Does Not Apply'. If the person type is not pedestrian and is different than 'Occupant', then the ped_location is set as 'Does Not Apply'.
  - With these changes, we have no more missing values.

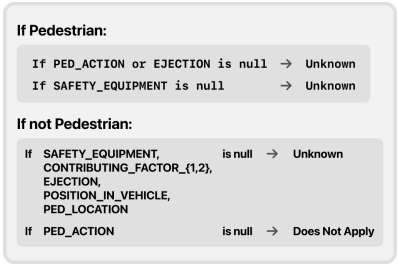


Figure 1: MV Custom Imputation

Regarding the **encoding**, even though on the Data Granularity section we presented a way to encode our features, we still want to test one-hot encoding, as such we will be using it, and we will also be doing encoding with the rules from the granularity section. Note that for CRASH_TIME and CRASH_DATE we always perform one-hot encoding.

For this encoding, what we do is that we've previously created an excel file, with multiple pages (one for each feature) where we have three columns for each page: "Old Unique Value", "New Value/Group", "Value". This way, and with the values that we've described in the granularity section, each feature is assigned the value of the new respective group. The main idea, just to recap, was to give an implicit order to the values, and, with the concrete numbers that we've attributed, we've tried to create distances between the different groups that were consistent with the criteria used.

Regarding One-Hot encoding, we quickly dismiss it because for KNN the Recall is very low. With One-Hot for KNN the predictions were almost all 'Injured' for the Killed records. This leaves us with Ordinal Encoding for both Custom Imputation and Replacement with the most common value. What we want to maximize is Recall, since we want to have the most Killed records being identified as Killed. This way, **we choose the Custom Imputation with the Ordinal encoding** since they hold the best results for recall.

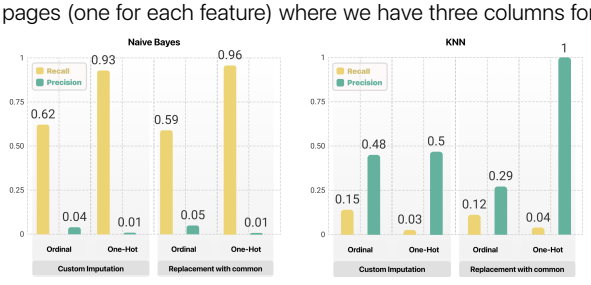


Figure 2: Naive Bayes and KNN recall and precision for each one of the configuration combinations.

For the Custom Ordinal Encoding with Naive Bayes as it is a probabilistic classifier, and because of what we've seen on the sparsity section, it is not strange that it behaved so well on all metrics (sure, not in Precision, but if we deep dive, this is in fact not that bad since the data is so unbalanced!). For the

KNN and because of the sparsity of the data and its imbalance, the results were also not strange. Ultimately, what this showed us is that, for KNN - a distance based algorithm -, we aren't able (with our configurations) to achieve an high recall (the metric that we focus the most). We can hypothesise that this might be due to our custom encoding, however here we had to find a trade-off since we needed to transform our symbolic features to numeric ones.

Outliers Imputation

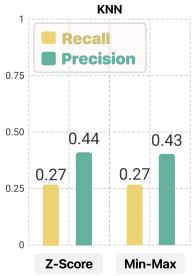
Regarding outliers for our only original remaining numeric variable: PERSON_AGE, with the iqr criteria, 291 records are identified as outliers, and, with the stdev criteria, 2174. Regarding the stdev criteria, 42 of the 2174 found records belong to Killed records. Because of our already deeply unbalanced dataset, these records won't be identified by us as outliers, as we don't find the forecast of this trade-off to be of value, this is a **pragmatic** decision!

Regarding the remaining values for the stdev criteria, we are mostly talking about the very young and very old people. If we look at the variable boxplot, it doesn't make sense for us to be losing information for young individuals, and, regarding old ones, for the stdev criteria starting from 70 years, every record is classified as an outlier. Even if they are 'theoretically' outliers, they present valuable information, and some of these values shouldn't be classified as such.

With this, a good trade-off is to use the iqr criteria, however, 14 of the 291 records are Killed (we still think this is a too high percentage of records from that class to drop), thus we just remove the remaining ones, they are so few that they won't impact any of our models. This way, and for this dataset, **our only outlier imputation is for the values that are classified as such with the iqr criteria, and that are Injured and not Killed**. For the remaining features, they were Categorical and we then proceeded to encode them, as such, the traditional methods shouldn't be used here. Moreover, the results remain exactly the same (for Accuracy, Precision and Recall for both KNN and Naive Bayes).

Scaling

For Scaling, our results didn't improve for Naive Bayes (expected due to its probability and frequency base). For KNN, the results improved. This way we only present the results for KNN, using the previous selected missing value imputation, dummification and other transformations methods. If we remember, only PERSON_AGE is numeric, and all other features were symbolic but then transformed into numerical ones. What this means is that, since our features distributions aren't Gaussian, and they are bounded, min-max scaling should work better. In this particular case, they didn't. However, we should also notice that the difference between z-score and min-max is quite small, in practice is the difference between having two Killed records being misclassified! This way, we are going to **proceed with the z-score scaling**.

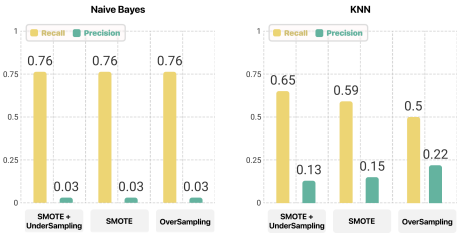


Balancing

Regarding Data Balancing, we are going to test SMOTE, SMOTE + Undersampling and OverSampling. just using UnderSampling held weak results, as such, here we do undersampling of the majority class to 15 thousand entries, and SMOTE for the minority one, to match the majority class entries.

We need to choose what balancing technique to use. In theory we could even not use any, however, since our data is so unbalanced, it wouldn't be correct as we need to try to properly define our each one of the classes properties, and, mainly for Naive Bayes, it is important to have similar frequencies for each class. One could argue that if without balancing the results were similar, then, maybe, we were already generalising the data, however we can't be sure of that.

Regarding Naive Bayes, the precision slightly decreased, however the recall improved from 0.62 to 0.76 (at the expense of more False Positives). Thus, regarding Naive Bayes, any of these Balancing Methods could be the one we proceed with. With this said, there's a motto that we want to follow: prevent every death we can! What this means is that for KNN, recall improved but the precision fell dramatically. Our True Positives more than doubled however the false Positives grew even more. Following our motto, we want to preserve every life the best we can (depending on the context one could argue that the monetary cost or the panic that might be generated of trying to save so many more lives - the false positives - could maybe not be bearable and put in risk the resources for the true positives). This way, we want to maximize recall, **we proceed with SMOTE + Undersampling**.



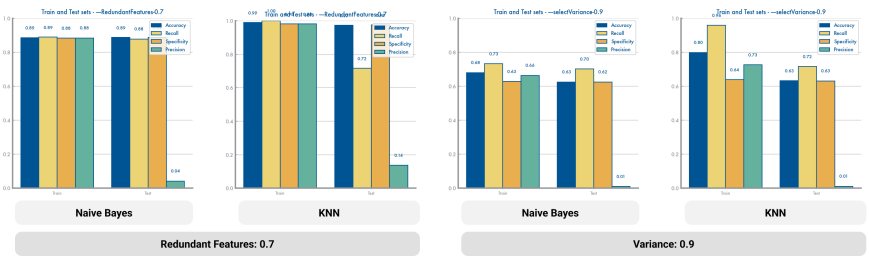
A note must be done that when balancing the data, the False Positives increased significantly for both the Naive Bayes (1244 to 2095) and KNN. One can probably assume that this is due to points on the boundaries of the decisions between the two classes.

Classification

Feature selection

With a threshold of 0.7, for the redundant features, the features to be dropped are: ['POSITION_IN_VEHICLE', 'CRASH_DATE_WEEKEND']. With 0.7 are: ['CONTRIBUTING_FACTOR_2', 'CONTRIBUTING_FACTOR_1', 'POSITION_IN_VEHICLE', 'PED_ACTION', 'CRASH_DATE_WEEKEND']. We've also tested with other thresholds for both redundant variables and variance, but ultimately the ones we present held the best results! Furthermore, it is expected that dropping redundant features bears good results since we have another feature that 'behaves' - in a way - similarly.

Moreover, for variance, with 0.9 the 10 out of 22 features are removed (namely SAFETY_EQUIPMENT, PERSON_TYPE and EJECTION are chosen to be dropped). Due to the way our symbolic features were encoded into numeric ones, COMPLAINT has a variance of 32.711, CONTRIBUTING_FACTOR_1 of 44.71 and SAFETY_EQUIPMENT of 70, and so forth. Thus, for this particular dataset, and given our encoding, variance doesn't, indeed, work.

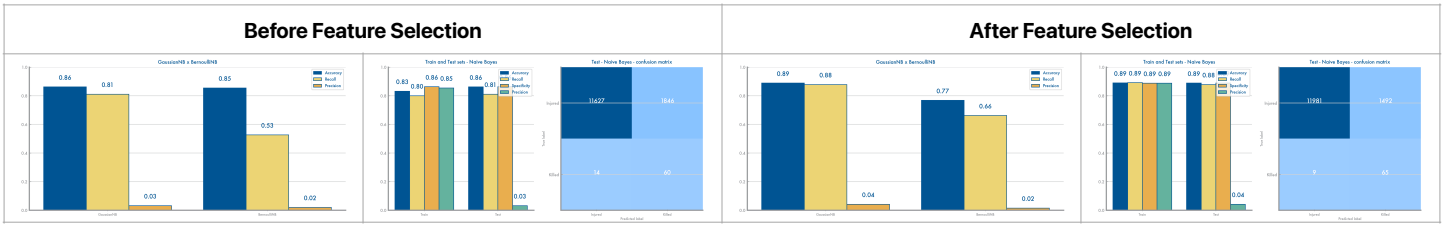


Dropping the redundant variables with a threshold of 0.7 bears the best results. For Naive Bayes, both recall and precision improved. Regarding KNN, if we want to, again, maximise the number of True Positives then we should go with the threshold of 0.7, where the False Positives also decrease. Regarding Variance, the accuracy dropped to 0.11, being discarded.

This way, we are going to proceed with the feature selection method using the redundant analysis with a threshold of 0.7. Thus, for the classification section for dataset1, **redundant features with a threshold of 0.7** is the method we use to do feature selection!

Now, for classification, we need to test it before and after performing feature selection. **As such, every time that we don't present any image explicitly for 'before' and 'after' feature selection, it means that the results are equal or very identical.** Furthermore, if there is any significant difference, we will describe it! Also, regarding the metrics we study, we've primarily chosen recall, since it is the metric we want to maximise. However, if for any given recall plot, there is little to no difference regarding the different model parameters **we also test the accuracy and precision plots to see if they provide more information**. In case they do, we show them, otherwise we don't. This same logic is used for showing or not the training set performance. Furthermore, for each different classification algorithm, the best model will be identified if it is regarding before of after feature selection.

Naive Bayes

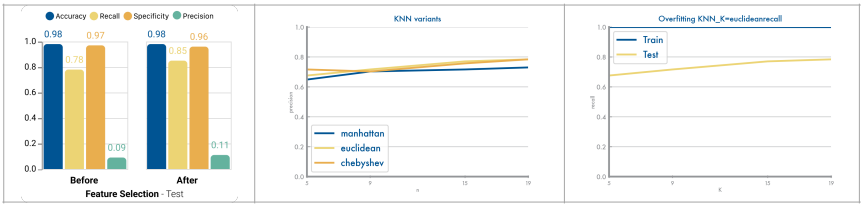


Regarding Naive Bayes, what we observe is that with feature selection Recall Improved. This makes sense since what we are trying to achieve with feature selection is the removal of 'noise' (less important) features, allowing the model to focus on what is more important (in a sense giving more weight to more relevant features). Regarding Gaussian and Bernoulli models, for recall the gaussian one was always better, something that was expected given that even though our features aren't gaussian distributed, they are more like it than a Bernoulli distribution (as they are mostly non binary)! In the end the best model before and after feature selection is the Gaussian Naive Bayes.

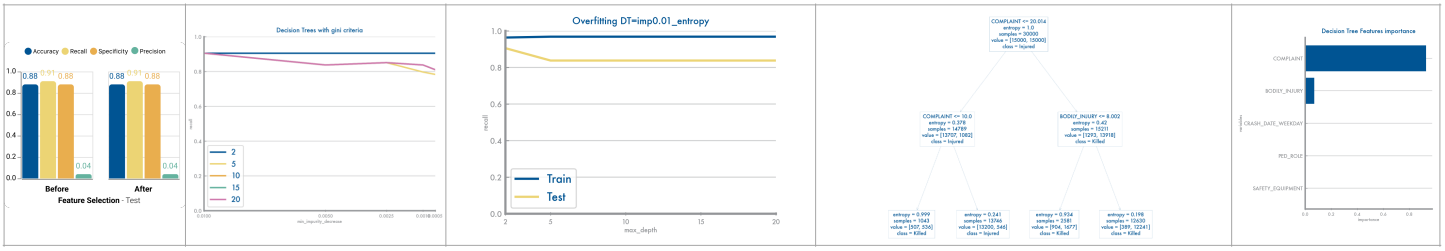
KNN

With feature selection our results improved for both recall and precision! We may argue that, due to the way the algorithm works, having less features, and keeping only the more relevant ones, may allow the algorithm to more easily find 'decision-boundaries' between the records. Furthermore our results mainly with the increase of K. As we've described, even though there are high correlations between some features, there's no easy way to describe how is the typical 'Killed' record, as there are multiple 'profiles' for Killed records.

This way, it is not strange that with the increase of K the results improve. Regarding the metrics, and because of the way our symbolic data was transformed (similar to a 'grid'), it is no surprise that chebyshev holds better results as it allows distances to be measured 'diagonally' between cells, whereas the Manhattan distance does not. Ultimately, the best model was with euclidean distance and 19 neighbours after feature selection. Furthermore, as the recall keeps improving regardless of the number of neighbors, we aren't in a overfitting territory, even though for the training set, the recall is one. Furthermore, we don't present the KNN variants plots before feature selection as it is identical and we want to prioritize our explanations in favour of images that would provide no new information.



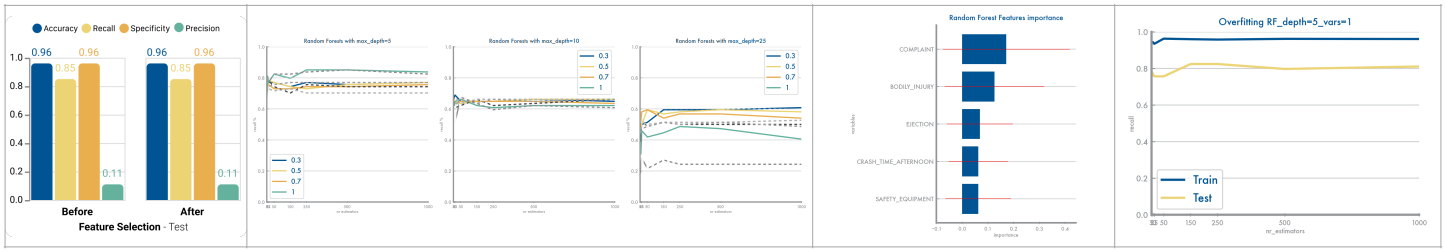
Decision Trees



Regarding decision trees, we observe identical results for 'after' and 'before' feature selection and also for both entropy and gini criteria. Furthermore, as expected, COMPLAINT is the feature with more importance, accounting for more than 0.8 of the importance. Since only two features are used, (and COMPLAINT is used multiple times since on C4.5 numerical features can be used multiple times), and since these features aren't discarded with feature importance, the best tree is also equal before and after doing feature selection. Also with a bigger max depth, recall drops for the test set. With bigger depths, precision does improve since there are more decision boundaries, however and also as expected, we lose in generalisation of the Killed records.

Ultimately, we also obtained expected decreases of recall with a lower min_impurity_decrease, as we are allowing greater decreases in impurity. Furthermore, for both before and after feature selection the best model is with entropy criteria, depth=2 and min_impurity_decrease=0.0100, resulting in a recall of 0.91. Regarding overfitting, the recall remains constant for depths 5, 10, 15 and 20 for both before and after. However, and since the recall deteriorates from max_depth 2 forward, we are overfitting in our training data, be aware, however, that we shall not forget how we performed our balancing: with smote, resulting in many artificially generated new records for the minority class, which will, in a sense, create very similar points (almost as duplicating, albeit not as naive), which can be the explanation of these results. Furthermore, if may be inglorious to classify this as overfitting, as it is in the nature of this model to deteriorate recall at the cost of precision with bigger depths...

Random Forests

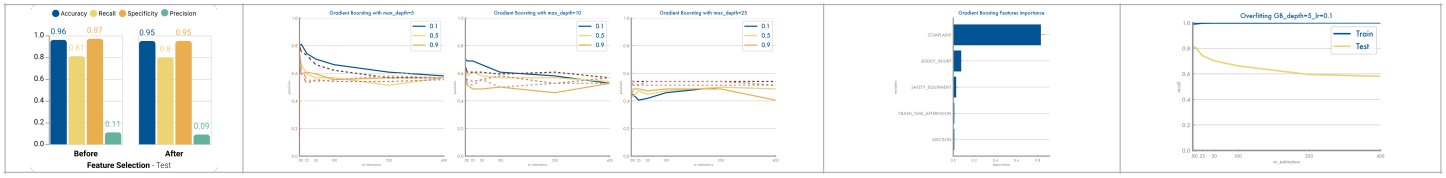


For the random forests study image we plot the before and after on top of each other, where the gray dotted lines are for the '**before**' feature selection. As expected with more depth, the worse the model, as we will have more 'tight' boundaries that will favour the more 'sparse' Injured records. Furthermore, and for the same reason, the more estimators we have, also the worse recall, even though we obtain much higher precision values! Also, regarding max_features, we obtain mixed-bag results! Since we've used 1, and not 1.0, sklearn understands max_feature as the number of features to consider at each split. This way, it is understandable that with only 1 feature, the lower the depth, the better the results.

Furthermore, and for the float values (0.3, 0.5 and 0.7), sklearn uses this as a fraction (max_features * n_features) to be considered. Here, we also understand that we don't want to use too much features (since our dataset is sparse and many features, as we've understand by now, aren't that important nor correlated), this way the lower the value, the higher the results (0.3). Similar to decision tree (lower depths are better) with a depth of 5 and considering one feature at each split is when we obtain the best results. Also as decision trees, the most important features are complaint and bodily_injury, even though, due to the random forests way of working, other features (ejection for example) also have some weight in the features importances. It is curious that, since every estimator has equal weight, and with such extreme features importances, the recall isn't that lower than for the Decision Trees. What this tell us is that with just COMPLAINT and BODILY_INJURY (from the decision trees) we can obtain a valuable generalisation of the data (generalising EJECTION and SAFETY_EQUIPMENT, the features from random forests). The best model was found with depth 25, *1* feature, and 10 estimators for before feature selection and 50 for after feature selection.

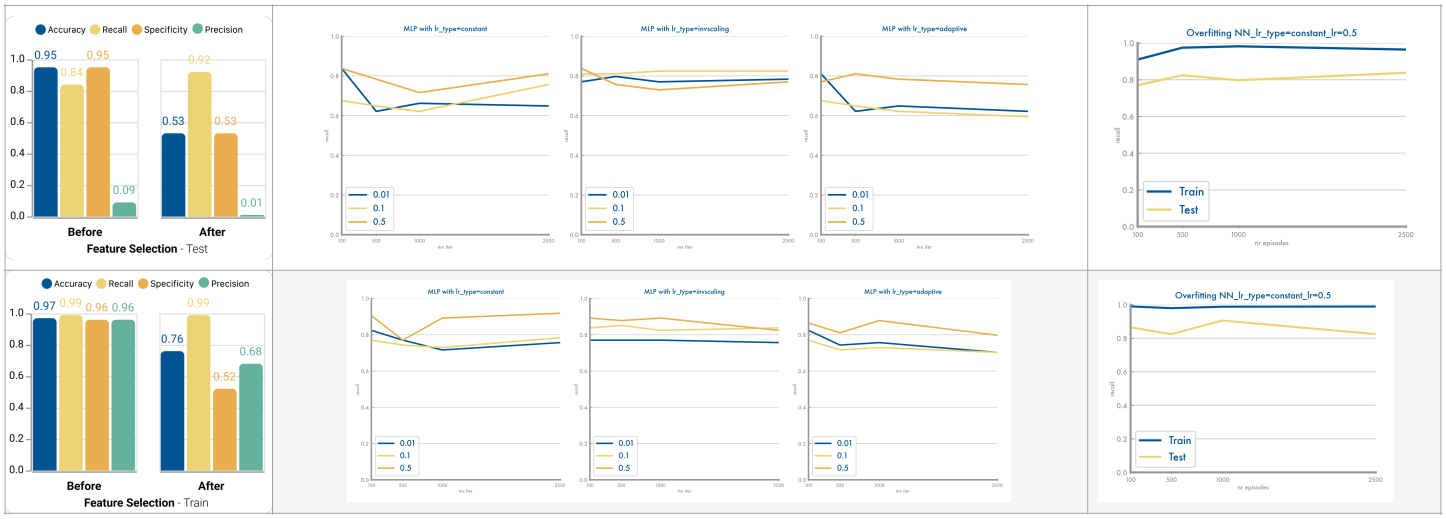
We don't plot because there is not much interest, but, when doing the plots for before and after feature selection but with regard with precision and not recall, regardless of before or after, the bigger the depth, the higher the precision which is the opposite that happens with recall. This is a very expected result as we have more information to further perform the splits at the cost of loosing generality.

Gradient Boosting



With Gradient Boosting, and as we've seen previously, COMPLAINT and BODILY_INJURY are as expected the most important features. Also, the lower the depth, the better the results since our data doesn't have well defined boundaries and since our balancing technique, SMOTE + Under-sampling, can't totally overcome it. Furthermore, and for lower max depths, the lower the number of estimators, the better, at the cost of precision. This is also seen on the overfitting plot, as with more estimators, the model isn't generalising the data, and is, instead, overfitting to the training data. Furthermore t**he best model results are with depth=5, learning_rate=0.1 and 5 estimators for both before and after feature selection**.

Multi Layer Perceptron



Lastly, and as deep learning is a world on its own, it is not easy to properly analyse these results. Neural networks have many shapes, number of different layers, ways of training, activation functions, and the list go on on things we could do differently. For example, the number of hidden layers on sklearn are 100, and the activation is relu. Even though these can be good default numbers, they probably aren't for this dataset. Ultimately, regarding the lr_type, not much difference is observed between the three different types, thus we can't properly analyse them. Theoretically the adaptive learning rate should bear better results, however we can't clearly say it. Moreover, the MLP wasn't able to converge independently of the number of iterations we did it sometimes even got worse with the adaptive lr_type, and also didn't bear better and more stable results with a lower learning rate, as we would expect. Due to the high number of parameters that this network had, and the way our dataset is distributed, it is not that strange that it didn't handle it well. What is

noticeable, however, is the difference in accuracy and precision after feature selection. MLP's are a type of model that usually handles very high-dimensionality data with some ease. Thus, we can hypothesise that with less features, and due to the way neural networks work, the model wasn't able to create decision boundaries good enough to properly distinguish between Injured and Killed records as most of the neurons on the hidden layers probably had very low gradients.

Ultimately, and with more episodes, after feature selection the recall deteriorates, therefore we are overfitting to the training data. Lastly, the best model both before and after are with lr_type=constant. For the before best model, it was found with lr_rate=0.01, max_iters=100, and for the after feature selection best model, lr_rate=0.5 and max_iter=2500.

Note: The first line of the table presents the mlp study before feature selection, and the second line presents it after.

General analysis

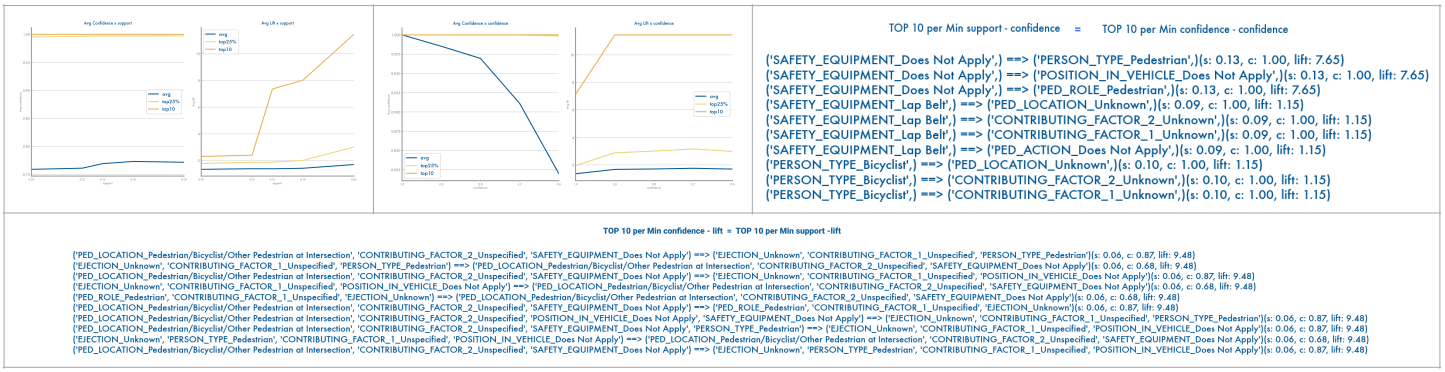
Comparing the results between all the models, after feature selection there is a tie between MLP and decision trees regarding recall. Before feature selection, the best model was also found with decision trees. For this dataset, and as we've stated before, there was a false predictor (Emotional Status), however, we haven't used it and we still obtained better results. Actually, for decision trees, when using Emotional Status, it became the feature with more importance however, the results slightly deteriorate! Furthermore, as expected - given the correlation with the target feature - , COMPLAINT was always the feature with more importance.

Even if it is not the point for us to choose what model would we deploy in a real-life situation, it is important to understand the trade-off between performance, complexity and interpretability between the models. With MLP we obtain a recall of 0.92 against 0.91 of the Decision Tree, however with MLP the computational performance is substantial worse (more demanding), it is much harder to understand what the model is doing and when making a specific prediction it is also harder to understand why a given prediction was made. At the end of the day, even if the Decision tree was, let's say, 5% worse, it would probably still be a better choice. Moreover, with the decision tree, as we have more insight into the model, we could also try to predict if there is some type of bias, regarding sex or age.

In the end of the day, even the *simplest* models, KNN and Naive Bayes performed very well, however, and given that the leap to decision trees is a easy one, we would go with decision trees.

Pattern Mining

Regarding pattern mining, we followed the suggestion that was given to us. For the first dataset we've dropped the numerical features, all ID's and the target feature, and performed one-hot encoding. This way we won't be testing for equal-width nor equal-frequency discretisation. Also, we used a minimum and maximum support of 0.05 and 0.2, after trying with different values.



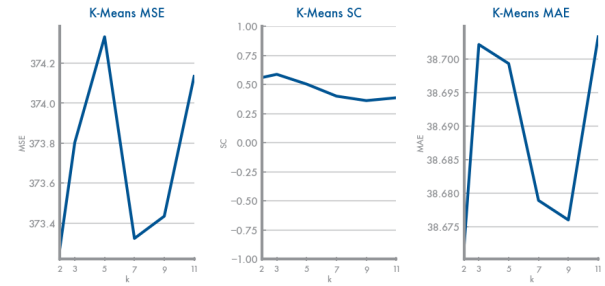
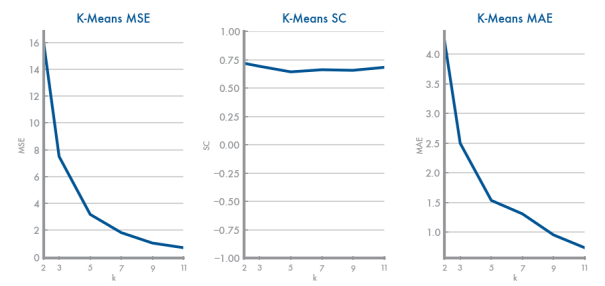
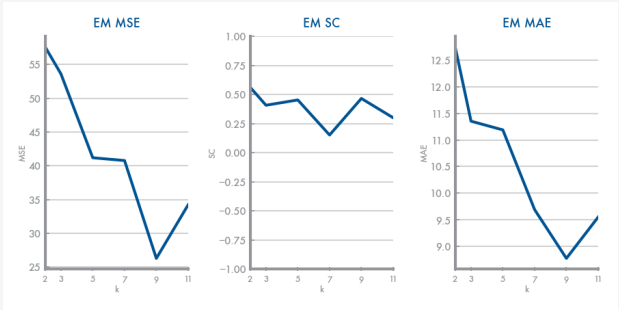
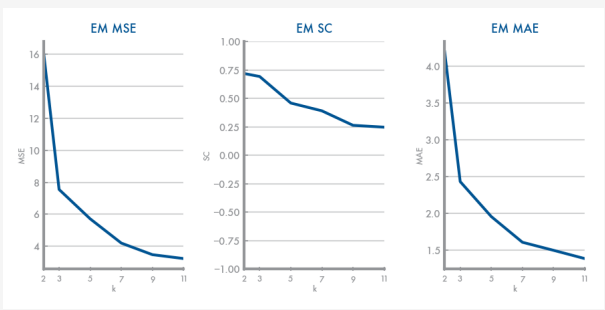
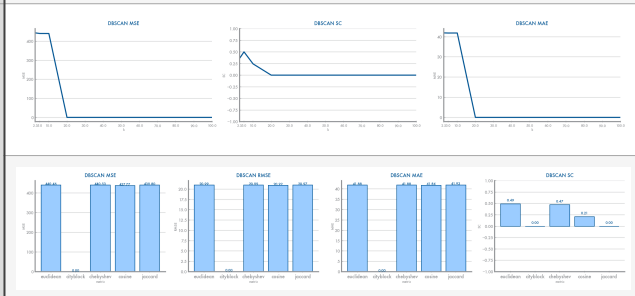
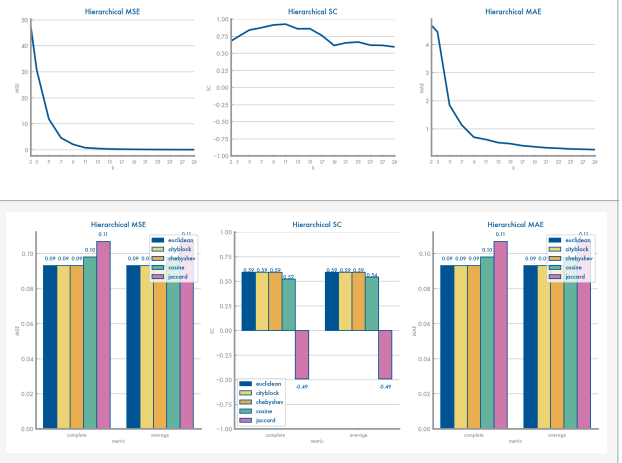
As expected, given that our data is quite sparse, we need to use very low support and confidence values to obtain substantial amounts of rules. However, even for not that small support values (0.2), almost 1500 patterns are found and around 50k rules. Furthermore, as observable on the plots, it is still possible to obtain rules with very high levels of confidence. We also only present two images regarding the rules because they were equal for both 'TOP 10 per Min support - confidence' and 'TOP 10 per Min confidence - confidence' and ' TOP 10 per Min confidence -lift' and 'TOP 10 per Min support -lift'. Regarding the results *per se* for the 'TOP 10 per Min support', the results are quite weak. None of the results are of interest and mostly they are for 'Unknown' values. Here, what is of interest if the fact that it was not possible for the apriori algorithm to retrieve better rules, something that we weren't expecting since, as we've seen', there are good correlation values, however, and probably due to the sparsity of the data, there aren't enough values with enough support and confidence to retrieve good results. For the 'TOP 10 per Min Confidence', the results are also mostly regarding 'Unspecified' or 'Does_Not_Apply' value, and are also of no interest.

Lastly, there's not much explanation that we can give to these results. We were expecting some rules to be obtained like, 'SAFETY_EQUIPMENT_NONE" ==> EJECTION_EJECTED. However we didn't. This is probably mostly due to the fact that the biggest correlations were for the Killed records, however, since this data isn't balanced, these records are statistically irrelevant...

Clustering

Regarding Clustering, the first challenge we encounter is to select which variables to use before PCA. Since most of our variables were Symbolic ones, but that are now transformed, this brings further difficulties. This way, the pair of features that we want to use are, as weak rule of thumb, the ones that cover the most of their domain. With this reasoning, one pair arises from the sparsity plot: (Bodily Injury, Contributing factor 1 or 2). However, and after testing it, we got quite bad results, thus, and after testing other pairs, we are going to use the pair (Bodily_Injury, Safety_Equipment).

As a remark, note that our data is already drastically "visually" separated due to the granularity changes we've applied - as it is possible to observe on the sparsity plot in the appendix -, thus we are not expecting information of great value to be retrieved by the clustering analysis. Also, note that number of clusters tested were between 2 and 11, as having more clusters would rapidly coincide with the number of unique values for any of the features.

	BEFORE PCA	AFTER PCA
K-Means		
Gaussian Mixture (EM)		
DBSCAN		
Hierarchical		

(Remark: we've chosen MSE and SC since they provide us metrics for cohesion and separability. We also present MAE but with the note that our dataset has little to no outliers.

What we found is that after PCA, most of the results did indeed improve for both metrics, showing that our data is, to a certain point, linearly separable. Moreover, before applying PCA, the best methods were DBSCAN and Hierarchical, something that was expected given that they are Density-Based and Hierarchical methods and that can, therefore, 'capture' more diverse structures in the data, like round and spiral shapes.

Regarding the values per se, the silhouette score gives us a value between 1, and -1, being 1 the best. What we observe is that the for the models with 'k', the bigger the k, the lowest the silhouette score. Furthermore, this is not without unexpected results. For example regarding K-Means, the results actually got worse with bigger values of k before PCA for MSE and MAE. As it is said in chapter 15 of the Data Mining Book, representative-based clustering methods as K-means are usually used for finding more oval and circular shapes, which could provide an explanation for the poor results with bigger K as our data is not circular as seen in the data profiling plots. Regarding DBSCAN we are once again plagued by the way we performed our encoding as most of the features are in well defined small ranges (mostly ranging from -10 to 10). Moreover, the highest score for SC for DBSCAN is precisely at eps=10, which was the distance used to separate most of the groups we made when encoding our symbolic features. Also, it is with the city-block metric that better results are obtained, something that is a result of most of the features being encoded in a city-block like structure.

Ultimately, if we wanted, we could now group different records based on their safety equipment and their respective body injury. With these results, if we wanted, we could even try to create new features, based on the different groups before applying PCA (while we can assess the results that we have).

Feature Generation:

Regarding feature generation, and as an extra, there is the possibility that new features, made with information available on the current dataset, may help our models. One possibility would be to, at the beginning while we had the VEHICLE_ID feature, create a feature that was the number of times that a car was involved in an accident. We would need to group the VEHICLE_ID with the COLLISION_ID (we need to group these records because otherwise, in an accident where there was a driver and an occupant, the vehicle would be counted as being involved in two accidents when it was just one). After grouping these records we would just need to verify the number of times this 'pair' is repeated and we could add a new numeric feature that was the number of times a vehicle was in an accident (or a boolean feature with yes/no regarding it it was the first accident).

We would *hope* that with this, there might be some correlation between the number of accidents a vehicle was involved in (the number of vehicles that were in more than one accident would presumably a minority) and its lethality. Intuitively speaking, one could assume that someone who was involved in more than one accident might have a more aggressive driving style (or is just unlucky!). Even if this doesn't hold, it is still a good result to try!

Dataset2

Data profiling

Data dimensionality

Regarding supervised classification methods, we are going to use **ALARM** as the target variable. We want to be especially alert to this feature when analysing our data.

We have around 17000 records with missing values for the **Field_1** and around 7000 records with missing values for each numeric air variable(Every numeric variable except Field_1, GbProv and FID). We have **32** variables of which **27** are **Numeric**, **4** are **Symbolic** and **1** is **Binary**, and **169273** records. This is already presenting us with some problems: how to treat the number of records with missing values (Can they be simply dropped, or should we do something else? Regarding the symbolic variables, how are we going to treat them?). Lastly, our target variable has two values: Danger and Safe. **90%** of this dataset consists of records for Safe and **10%** on records for Danger, making this an imbalanced dataset.

Data granularity

Before analysing the data granularity, we need to understand how our data is distributed. We analysed every column to see if we could find any incorrect values (not noise nor outliers!), like negative numeric air values/emissions, and found out that some entries for GbCity had 's' instead of the real code, it was always for Fuyang so we searched for that code. We were able to find it: 3412, we replaced it in the code and after that we changed the column dtype to int64 since we only had codes and the variable is numeric. Moreover, there's little to no interest in working with all the features that represent an ID. In this case there's only one - **FID** so **we won't be working with that variable.** We also have a variable that won't affect the dataset in any way and that variable is **Field_1**, so **we won't work with that variable** as well. Finally, we have two pairs of variables that represent the same thing and those are City_EN/GbCity and Prov_EN/GbProv so **we decided to keep only one of each pair** and those are **GbCity** and **GbProv** because they are numeric and make the dataset easier to run without having to dummify the dataset to more than 300 columns and if we need the city/province names we have a 1:1 mapping.

Regarding time hierarchy, date records are grouped into 'Weekday', 'Weekend' and 'Holiday'. (We discovered a python library that could do exactly this for Chinese holidays as mentioned in the Dataset1 part). We also thought about grouping each City by Province or Region(that would mean dropping GbCity and working only with GbProv) but we could lose out valuable information and since they are two numeric columns it won't disturb the dataset.

Finally, for our remaining numeric variables that contain the air records we could group the Mean and Standard (X_Mean and X_Std) values into different groups like "Low", "Average" and "High" or "Very Low", "Low", "Average", "High" and "Very High"... but we found out that by using this grouping the performance was actually lowering (we've tested it with Naive Bayes and KNN) so we opted not to group any numeric variables.

Data distribution

Regarding data distribution, one important characteristic of this dataset to mention, and something that will be with us during the entire dataset exploration, is that it is unbalanced, with 16968 Danger records and 152305 Safe ones. This means that we have roughly 9 times more information about Safe records than Danger ones.

For our only symbolic variable, date, we could find a pretty expectable distribution with the majority of the records being weekdays(more than 100000), followed by the weekends(around 40000 records) and then by holidays(less than 20000). For our numerical air quality variables we found pretty normal and expectable distributions for every single one of them with most of them being represented with a Log Normal distribution.

For our other two numeric variables, GbCity and GbProv, there's nothing much we can retrieve by looking at the data, we have 31 different provinces with Shandong being the one with the highest count with 13470 records and Shanghai the one with the lowest with 449 records and have 371 different cities with Yichun being the one with the highest count with 898 records and Beijing the one with the lowest with 449 records (By using the 1:1 mapping we got the corresponding cities and provinces' names by their code).

Data sparsity

For most of the (Std,max) pairs, we have something quite tunnelled, which makes sense since the biggest the maximum value, usually the biggest the stdev since usually these particle values are in a well defined range. There are well correlated features that won't probably be necessary. Also, as an important notice, there are many pairs that cover a vast majority of the domain, as the pair No2_Mean x O3_Mean.

By using color in the scatter plots it is possible to easily identify the 'weight' each feature has to predict 'Danger' or 'Safe'. For example, between 03_Mean and PM10_Min it is possible to clearly identify a separation between Safe and Danger and this is well generalizable for other cases. Moreover, some features like So2 and CO have high correlation values which is probably explainable due to physical phenomena.

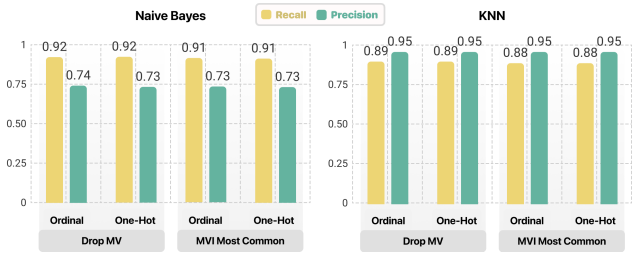
Some of the high correlations are also present with regard to the target feature. For example PM2.5 and PM10 which makes sense since PM10 and 2.5 are a measure of the amount of pollutant particles of size 10 and 2.5 micrometers and lower in the air.

Data preparation

Missing value imputation, dummification and other transformations

To test what works the best we created a pipeline that first imputes the missing values and then encodes them. For the imputation we test three different approaches: dropping all rows with missing values; replace the missing values with a constant value; replace the missing values with the most common value. Regarding the encoding we test two approaches: a custom ordinal encoding and one hot encoding. Custom encoding: Replace each entry for date with a numeric value(Ordinal encoder). We also thought about encode cities/provinces by distance but that was going to be too complicated and over the top so we opted not to do it.

For the other missing value imputations methods, they are quite straightforward. One replaces the missing values with the most common value for each feature, and the other replaces with a constant value (for numeric values replaced with 0, for symbolic replaces with 'Unknown', and for binary replaces with *False*).



By comparing the results obtained by the Custom Ordinal Encoding and the One-Hot Encoding we can only find one little difference and that is in the Naive Bayes for Drop Missing Values precision where they differ for 0.01 so we came up to the conclusion that it would be the same using one or another encoding. However, as stated in the dataset 1, we can't proceed with Custom Ordinal Encoding since the values that are being replaced with don't have any type of reasoning.

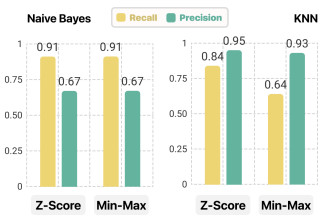
By comparing the results of the three imputation methods we can find similar results as well but the one that has the highest value for the most important metric, the Recall , both in Naive Bayes and KNN, is the **Drop Missing Values** one(it loses 0.01 on specificity but we want to focus on the true positives, not true negatives). By choosing this imputation method we will drop almost 8000 records. This could turn out to be a problem if we lose too many records on the class variable, but only 12 records are lost so we see no risk at all by choosing this imputation method, since we are still left with more than 140000 "Safe" records and virtually keep the same number of "Danger" records.

Outliers Imputation

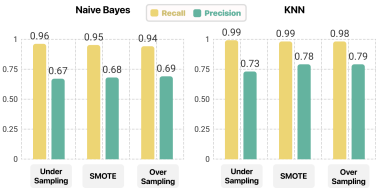
We found around then 200 records considered outliers with the iqr criteria and 0 outliers with the stdev criteria. There are almost no records considered outliers so we **decided to keep** them since the iqr criteria considered the dataset had 0 outliers but since it's such a low sample, keeping or changing their rows doesn't change the results.

Scaling

Since the distributions for the air quality variables are Log Normal we expected the results to improve with the z-score but they were slightly worse. The results for MinMax were expected to be worse than the z-score since we already have mean and std scores in different columns, it would be doing the mean of something that is already a mean. Since both configurations had worse results than the previous one **we opted to leave the scaling out of the equation**.



Balancing

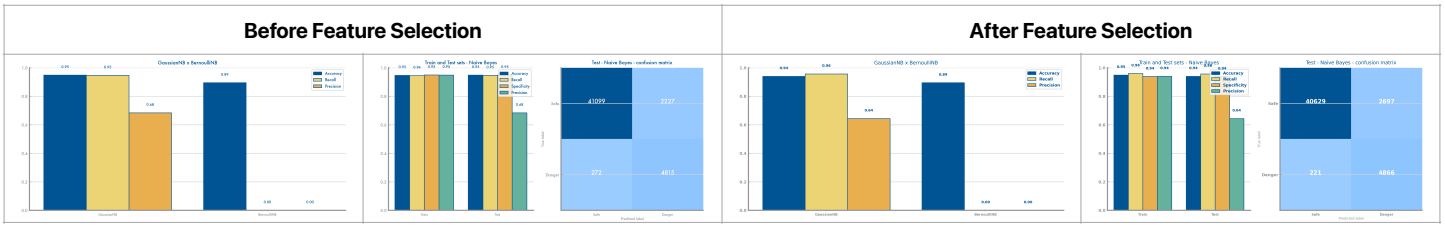


Between SMOTE and Over Sampling we got slightly better results with SMOTE which was expected since our dataset is mainly numeric and since most of the features cover a significant range of their domain. Moreover, regarding Under Sampling, not only are the results worse, but we also wouldn't want to proceed with it since the dataset is highly unbalanced and we want to preserve (or even increase) our data. Since this new configuration is better than the old one **we are sticking with SMOTE**.

Classification

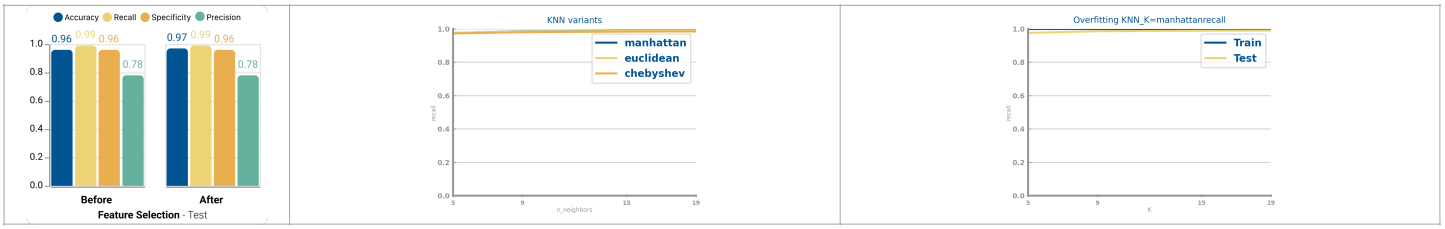
Regarding classification, we need to test it before and after performing feature selection (We present the results of the feature selection analysis on the bottom of this section so we can make a comparison between expectations and actual results).

Naive Bayes



Regarding Naive Bayes, what we observe is that with feature selection the recall improved. This makes sense since what we are trying to achieve with feature selection is the removal of 'noise' (less important) features, allowing the model to focus on what is more important (in a sense giving more weight to more relevant features). Regarding Gaussian and Bernoulli models, for recall the Gaussian one was always better, something that was expected given that, even though our features aren't Gaussian distributed, they are more like it than a Bernoulli distribution (they are non binary)! Before and after feature selection, the recall and precision for the Bernoulli model is 0 which means that the model only predicts "Safe" records and that is the last thing we want, since we want do maximise the number of true positives, the Bernoulli model is a bad option.

KNN

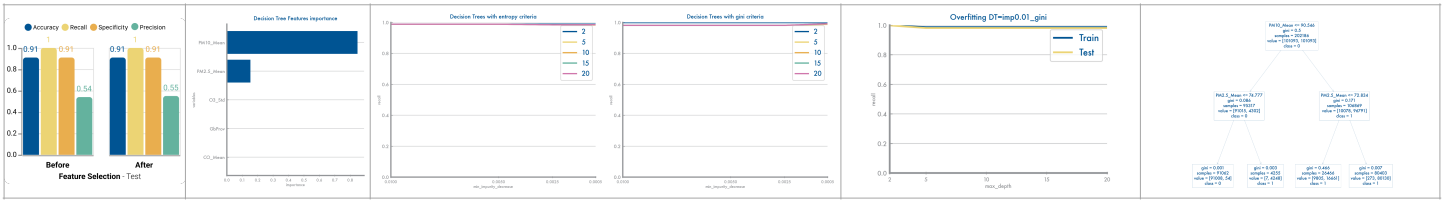


We only have one image for the KNN variants (excluding the graphs for the test results) and one for the Overfitting because the results before and after feature selection were identical.

With feature selection our results remained almost the same with only the accuracy going up by 0.01. This doesn't reflect the impact of feature selection, but the results were already great before it so there's little room for improvements. Furthermore our results mainly with the increase of K and that makes sense, since that with more neighbours to compare the more precise the outcome will be. Note that we only used odd K numbers, to avoid ties since our class variable is binary, and the k number that got us the best results, before and after feature selection, was 19. Regarding the distance metrics, the metric that got us the best results, before and after feature selection, was the Manhattan distance, but every result with every metric is very close to each other. That was expected since our dataset had really high sparsity, so the distance metric applied wouldn't matter that much. We obtained really high recall(0.99) which makes sense since our sparsity analysis showed us exactly that, the majority of our dataset had high sparsity, so it was expected to have a really high recall with the KNN method due to neighbours having similar classification.

We can also conclude that overfit didn't occur since we don't see a growing gap between the train and the test.

Decision Trees



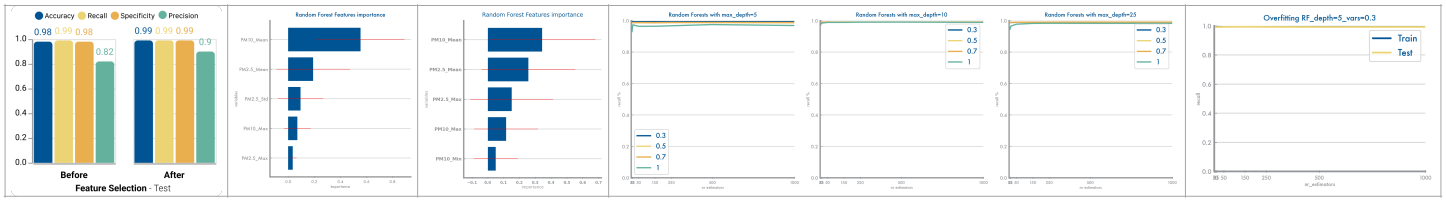
We only have one image for each study (excluding the graphs for the test results) because the results before and after feature selection were identical.

Once again, the results obtained with both criterias and different depths were almost identical obtaining an almost perfect recall. The best results before feature selection were obtained with gini criteria, 2 depth, min impurity decrease of 0.0100 and after feature selection with entropy criteria, 5 depth and min impurity decrease of 0.0100 but the difference between every metric and value is minimal due to having a high sparsity and a big dataset. The feature importance only relies on two features: PM10_Mean and PM2.5_Mean and those are only the two features present on the best tree obtained. This adds

explanation to why the results before and after feature selection are very similar. Our results are already great so it's difficult to make them better, but with the decision trees classification method we essentially only use two features and those features, obviously, aren't removed in feature selection so the results barely change. With an almost perfect recall achieved, we can also confirm our high data sparsity, with only two features we can predict almost perfectly every "Danger" record.

We can also conclude that overfit didn't occur since we don't see a growing gap between the train and the test.

Random Forests

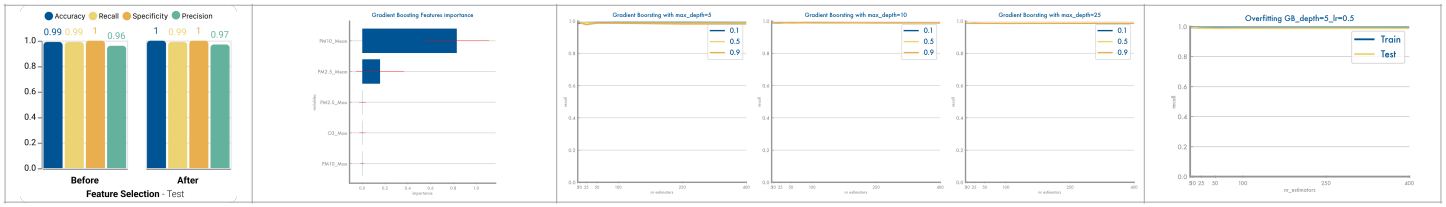


We only have one image for the criteria and overfitting studies because the images/results before and after feature selection were identical.

The results improved slightly after feature selection and that is due to the feature ranked third(PM2.5_Std) in the feature importance being removed with feature selection what causes the ranking to change significantly after feature selection. The best results were obtained, before and after feature selection, with max_depth=5, 0.3 features and 150 estimators. The max_depth=5 and 0.3 features values are the lowest we tested and it makes sense that they are the ones that generate the best results, since we are testing less things the performance of the model is better, for the max_depth we already know by the Decision Trees model that the best tree obtained was with only a depth of 2 so it makes sense the lowest max_depth(higher than 2) value tested generates the best results and the same applies to the features, as we can see by the ranking, only 4 features(of 30 before feature selection and of 26 after) have an importance of 0.1 or higher. The difference between features might not be that significant but it is noticeable in the study with max_depth=5 we can see a descending performance when we use more features.

We can also conclude that overfit didn't occur since we don't see a growing gap between the train and the test.

Gradient Boosting

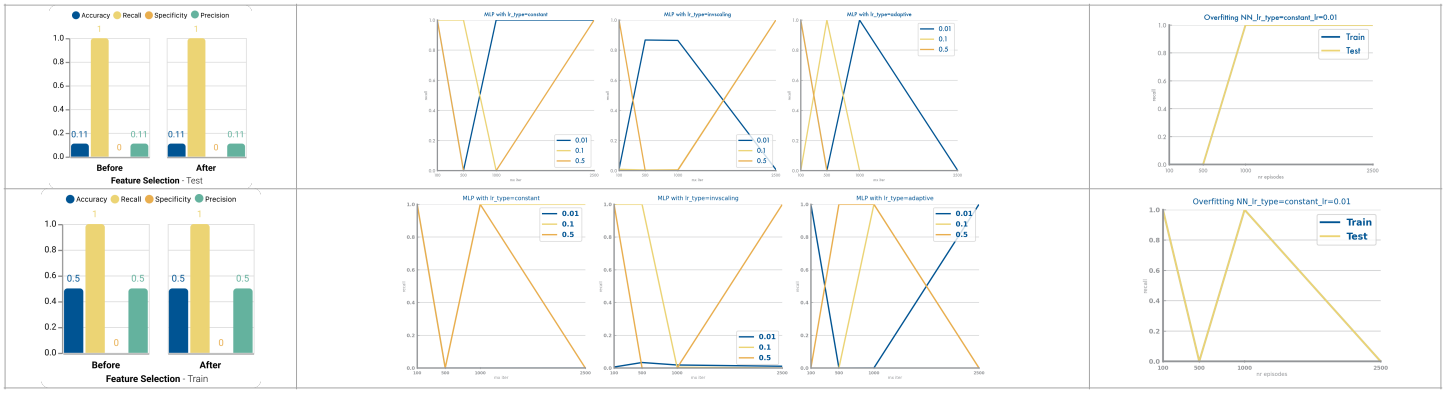


We only have one image for each study (excluding the graphs for the test results) because the results before and after feature selection were identical.

Gradient boosting already has an almost excellent performance before feature selection and it gets a bit better after it. By looking at the ranking we see only two features matter(PM10_Mean and PM2.5_Mean), the same ones from Decision Trees. The best results before the feature selection were obtained with max_depth=5, a learning rate of 0.5 and 100 estimators. This results only in learning rate from the ones after the gradient boosting where it is of 0.1. The max_depth=5 is the lowest we tested and it makes sense that it is the one that generate the best results, since we are testing less things the performance of the model is better, we already know by the Decision Trees model that the best tree obtained was with only a depth of 2 so it makes sense the lowest max_depth(higher than 2) value tested generates the best results. Even though these were the metrics that obtained the best results, every metric combination had really good results where the difference was really minimal, so we can say the learning rate doesn't matter much. Actually, we were even expecting the learning rate to be a higher value after feature selection due to having less variables, but it went from 0.5 to 0.1.

We can also conclude that overfit didn't occur since we don't see a growing gap between the train and the test.

Multi Layer Perceptron



The top graphs are the results before feature selection and the bottom ones after feature selection(excluding the histograms at the top top that is for test and bottom for train).

The best results in this method, obtained with a constant learning rate type, a learning rate of 0.01 and 100 max episodes, might have given us a recall of 1 but overall the results look very poor and very hard to analyse. Due to the high number of parameters that this network had, and the way our dataset is

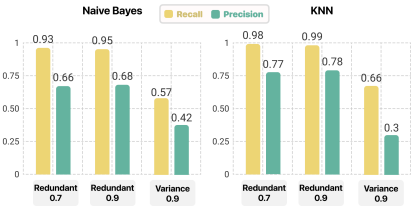
distributed, it is not that strange that the MLP didn't handle it well. The best results improved remained the same before and after feature selection probably due to the fact of only four features of the dataset being removed so we are still left with a high number of parameters, a high number of records and a similar distribution. By looking at the results we can conclude that the MLP isn't a good classification model for this dataset since it can't extract stable results due to the distribution and dimensionality of this dataset, due to this we can't be precise about the overfitting evaluation.

General analysis

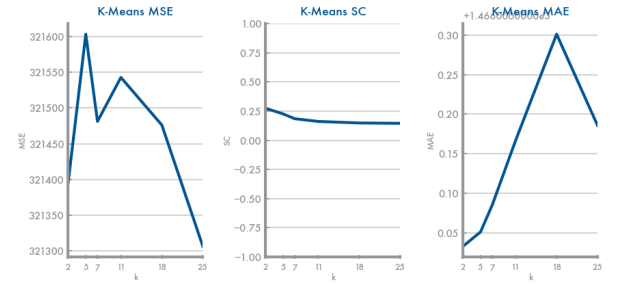
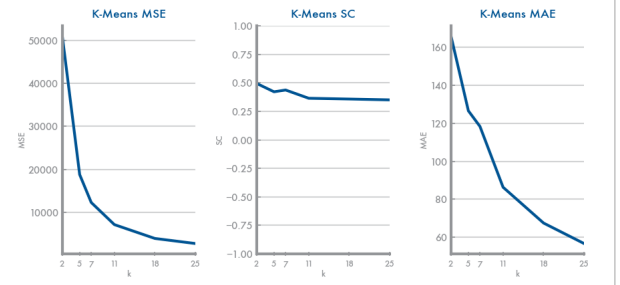
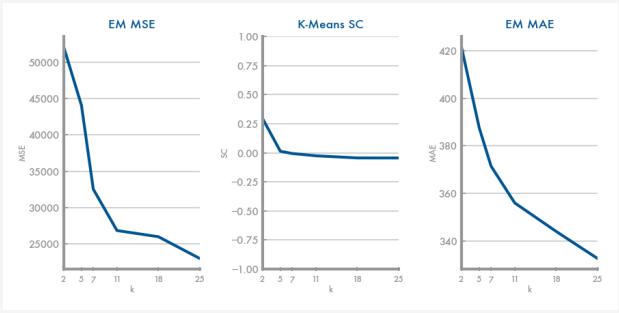
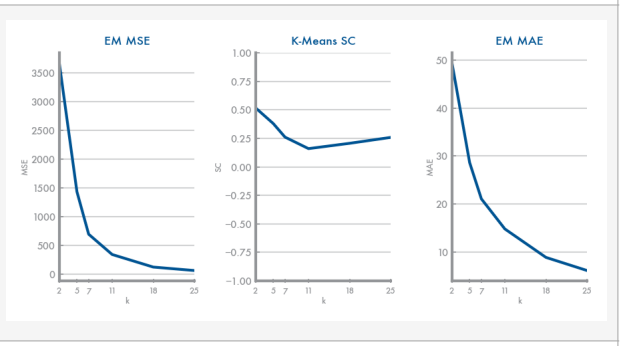
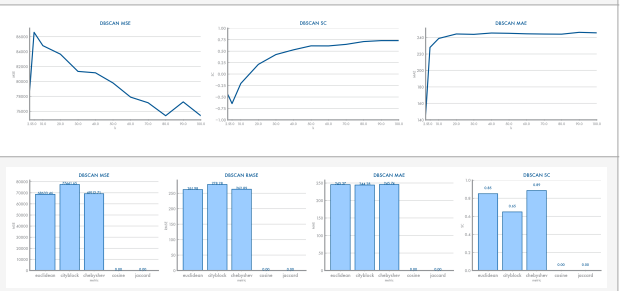
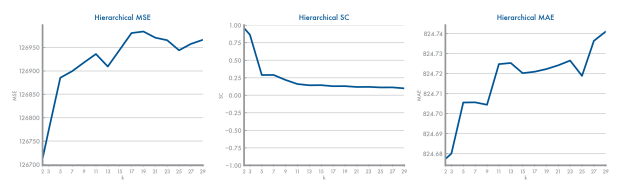
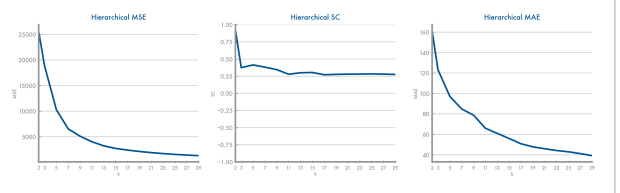
On a general level we can say that all models had a great performance classifying/predicting this dataset except for the MLP that could't produce great and stable results. The model that obtained the best results overall was the Gradient Boosting and the model that obtained the best recall was the Decision Trees. We can't really compare the Naive Bayes, KNN and MLP with other models but we can see some similarities amongst Decision Trees, Random Forests and Gradient Boosting since they brought us some similar conclusions like the PM2.5_Mean and PM10_Mean are the most important features to classify this dataset and that they all got better results with a low depth meaning only a few features really matter.

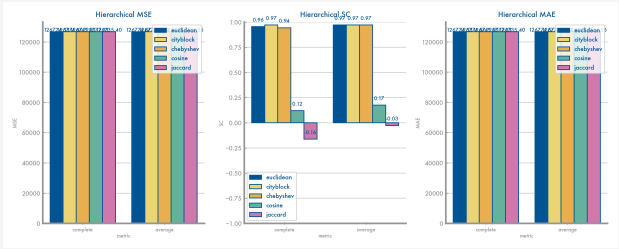
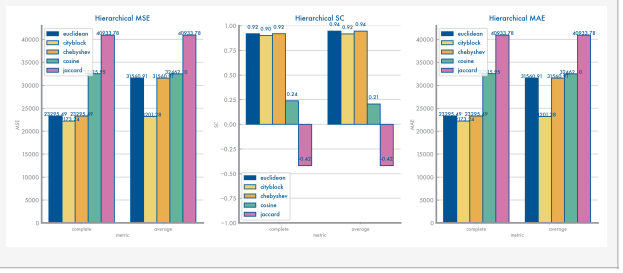
Feature selection

Regarding **feature selection** only 4 variables were removed - ['CO_Std', 'PM2.5_Std', 'PM10_Std', 'SO2_Std'] - the criteria used and the one that obtained the better results was **redundant features with 0.9 features**. This was a surprise to us since on three different models, Decision Trees, Random Forests and Gradient Boosting, there were only less than a handful of features that were ranked as important so we were expecting that the better results were obtained with lesser features which wasn't the case - The results got worse overall with less features for the Naive Bayes and KNN methods.



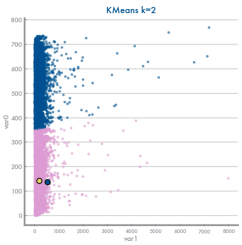
Clustering

	BEFORE PCA	AFTER PCA
K-Means		
Gaussian Mixture (EM)		
DBSCAN		
Hierarchical		

	BEFORE PCA	AFTER PCA
		

Regardless of the method used, with PCA we obtained expected values for SC, MSE and MAE. What this means is that with the increase of the number of neighbours for K-Means and the number of components for EM, MSE drops, while SC doesn't fluctuate much.

Moreover, regarding DBSCAN we also obtain better values with bigger eps distances. These distances go in accordance with our features range, as we've seen in the data distribution section, where the 'min' and 'mean' deviation features (for example for NO2_Mean and O3_Min) have ranges between 0 and around 200. Moreover, for the 'max' features the range is bigger.

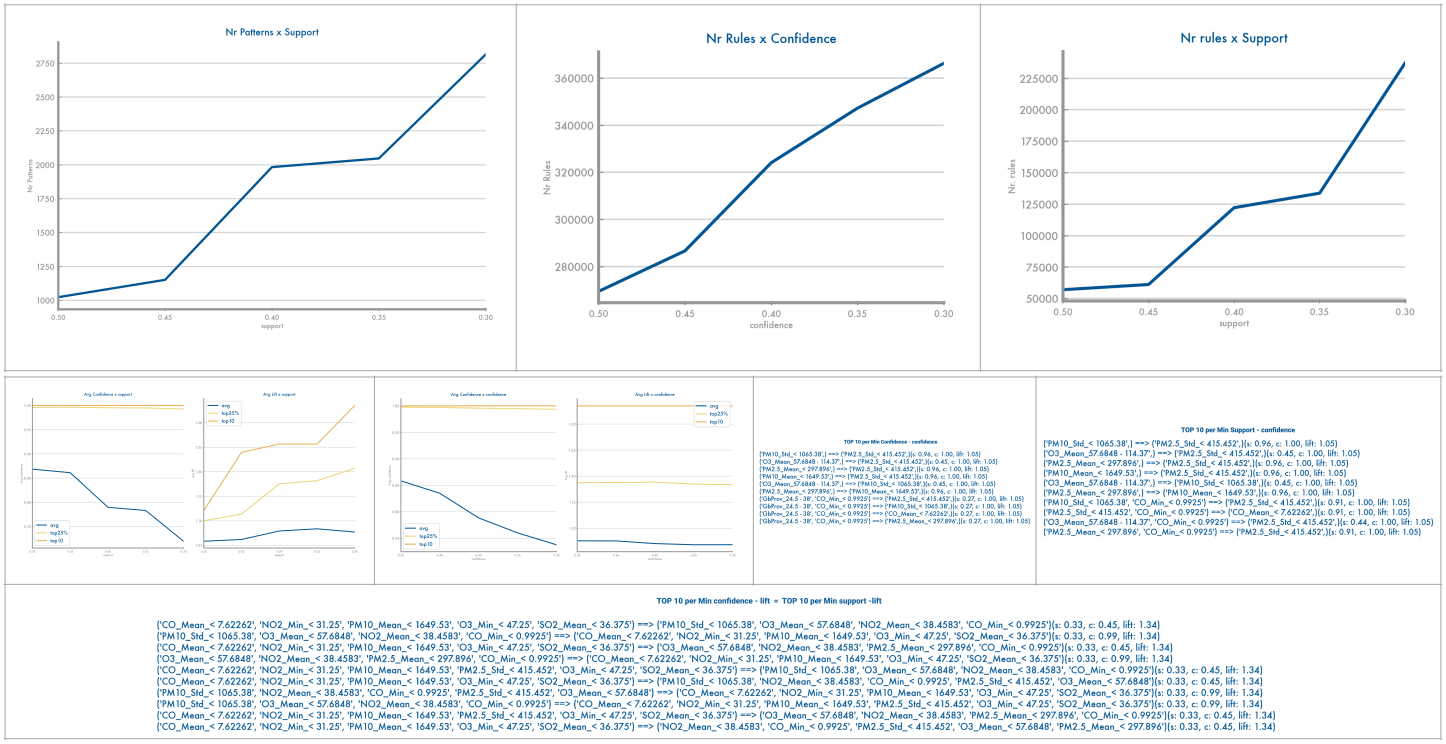


Regarding the metrics for DBSCAN we were surprised with the values obtained for MSE with regard to the cosine metric. Before applying PCA, and since the cosine metric is mainly used when the magnitude of the two vectors it is measuring doesn't matter, we wouldn't expect the results to be good. However, the fact is that the features we choose (the clustering plots are available in the appendix) are mainly *divided* by their angle, making cosine a good metric. After applying PCA, the results with the cosine distance were (once again not as expected) bad. As we can see in the image on the right, the transformation for two-dimensions was made mainly as a 'vertical' zone. Even if we remove the values for var1 bigger than 1000, we would get a 'square', making, once again, cosine similarity not suitable for this case.

Ultimately, PCA improved our results and, as we can observe in the appendix, we obtained well divided clusters for all the clustering methods. This greatly contrasts with the results obtained for the first dataset where the clustering was naively worse before and after PCA probably due to the granularity changes applied. Moreover, and using the elbow-method, we conclude that the best number of clusters is around 2-5.

Pattern Mining

Equal width



As the support and confidence lower their value, the number of rules and patterns found substantially increases. Regardless of the support, we can find different rules that always have a confidence near to 1. For both confidence and lift, regardless of the support and confidence values, their average values, as expected, continuously decrease.

When the support decreases, the average lift increases, as expected, and since the lift is always higher than 1, it seems like we have rules created where the antecedent and consequent are highly related, the support is also considerably high meaning we actually found some good rules using the equal-width binning.

We can see that the first bins regarding minimum values, means and standard deviations are the ones that create the strongest rules, since when the lower the emissions the safer we are, as we can see by the data sparsity analysis. And, generally, when we have lower values of something, the mean/standard deviation of that something is also lower, and the same applies vice-versa, the lower the mean/standard deviation the higher the chance of having lower minimum value (for the particular case of the particles of this dataset).

Note that we've used 7 bins with equal width. This choice was made by assessing that most of the features have a large domain with most of the data concentrated for the lower values. As such, we need more bins to capture the granularity of the lower values.

Equal frequency

With equal frequency, and similar to equal-width, as the support and confidence lower their value, the number of rules and patterns found substantially increases. For both confidence and lift, regardless of the support and confidence values, their average values, as expected, continuously decrease. When the support decreases, the average lift increases, as expected.

Curiously, when we put the plots from the equal-width and the equal-frequency side to side we can't really tell which one belongs to which binning technique, but what we can extract from both is actually pretty different as we can see from each top 10. We believe that grouping the data by equal-frequency isn't the best idea since it's splitting up the correlated data into different bins, so it won't create stronger rules as we can see in the different top 10s, even though they present a really high lift which mean the antecedent and consequent of the rules are highly related, the support is really low, meaning that we can't find any good rule that covers a significant part of the dataset.

Note that we've used 5 bins with equal-frequency. Here the reasoning is that since we have the majority of the records for each feature on the lower end of the domain, since all bins have almost the same number of records, the 'bigger' values won't impact the discretisation as much as in the equal-width discretisation, thus with 5 bins we can *capture* a reasonable amount of detail.

Critical Analysis

While the Data Science process doesn't come to an end, since we need to update and monitor our models, this report does 🥹! Our objective was, ultimately, to assess the basis of the Data Science process, mostly by analysing each step in a critical way. While we are happy with the final result of our report, this doesn't mean that we can't assess the fragility of some of the steps taken. The encoding for the first dataset might have been done in a finer way, or, at least, we could have tried other options. Furthermore, Pattern Mining wasn't as broad as we wanted, nor have we used their results to create new hypotheses. Some of these caveats, and others, however, didn't shoot us further from the fundamental understanding of how any future data science project should be assessed, as such, we need to be happy for our work!

The most important, while trivial conclusion, that we can obtain, is that each dataset is unique and needs to be worked on its own. Both datasets differed on the most fundamental parts, starting on the number of Symbolic and Numerical features each one has, and how to deal with them. Furthermore, the conclusion is that even though *rule-of-thumb* can be a good start, they don't always work as we (theoretically) may expect. For example, regarding the first dataset balancing, we would think that SMOTE would probably bear the best results, however, it was with a hybrid method that we've achieved the best results, with SMOTE and undersampling of the majority class. What this means is something very profound: fewer data led us to better results. This is the ultimate proof that one isn't smart by just having a library full of books (or records in this case) but needs to be able to remove all the 'noise' and focus on the essential, and here, it was by performing undersampling of the majority class that led us to achieve our classification objective: predict the most true positives as possible.

Similar conclusions are also extrapolated for the second dataset, even though it had its own caveats. The second dataset allowed for us to, maybe, be more focused on the raw outputs of the models and algorithms (from classification to pattern mining, passing by clustering) as the dataset didn't need as many changes as the first one.

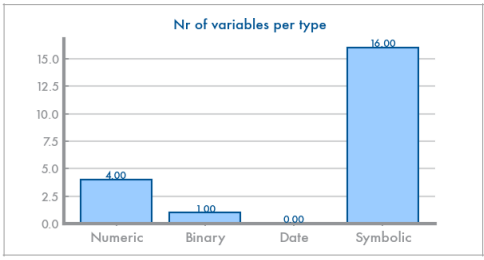
Focusing on the classification problem at hand, we need to clearly observe if are they good, or not, for more serious use, even if we discard explainability and monitoring problems that may be felt at the end of the road. What we have observed is that for both datasets, the most relevant features were roughly the ones that we thought would be. For the first dataset: COMPLAINT and BODILY_INJURY and, for the second dataset, PM2_5_Mean and PM_10_Mean.

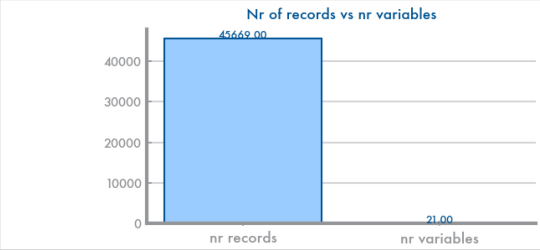
Ultimately, and for each one of the datasets best models, we've achieved good results across the board: high accuracy, recall and precision, while, at the same time, achieving it with a somewhat explainable and simple model like the Decision Trees. Thus, and because the numbers don't lie, we need to be pragmatic: with such results, good and valuable help could be provided in a real-world situation.

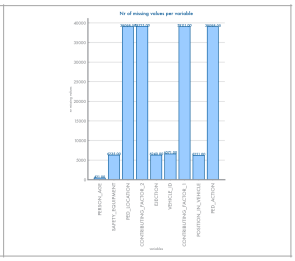
Other figures and details

Dataset 1 - Other images and informations:

Dimensionality

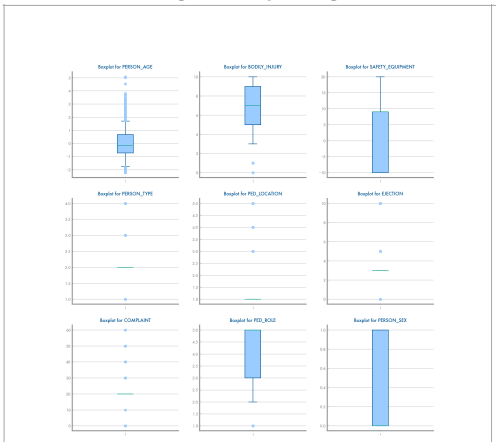


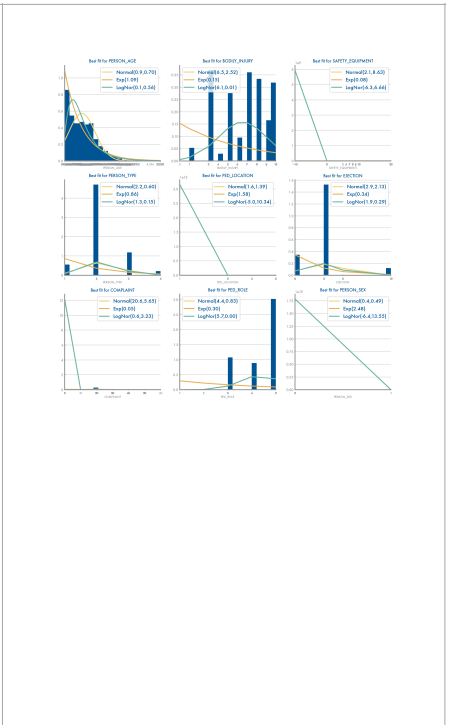


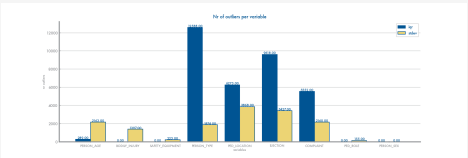


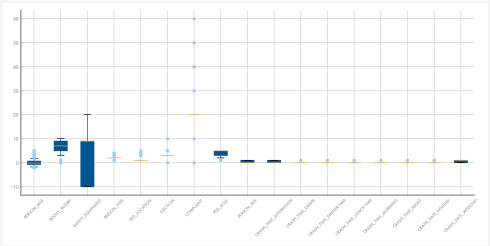
Distribution

Distributions after granularity changes

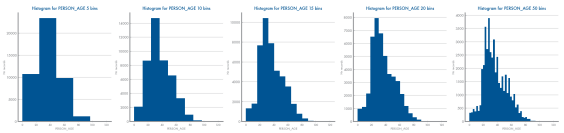




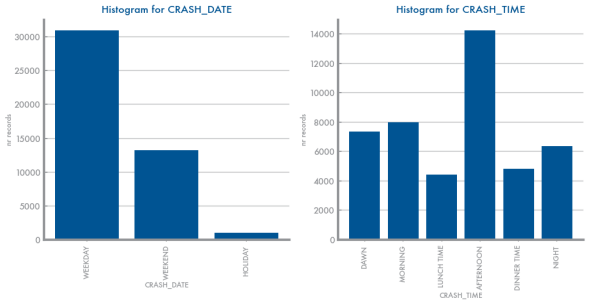


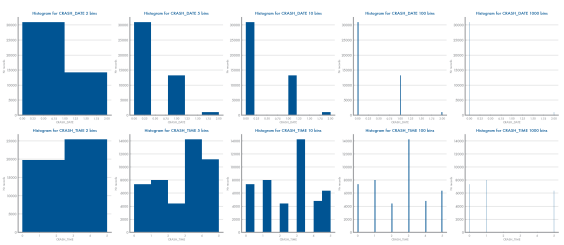


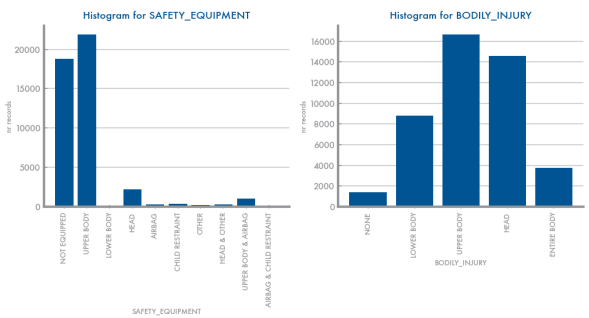
Granularity

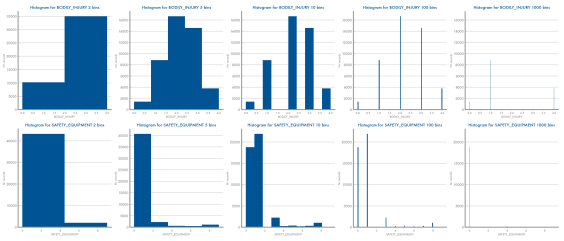


Regarding numeric features, we only present the plots for our only remaining 'original' numerical feature: PERSON_AGE.





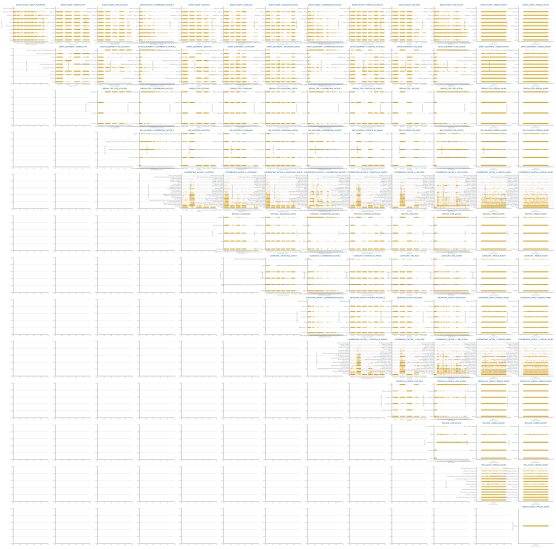




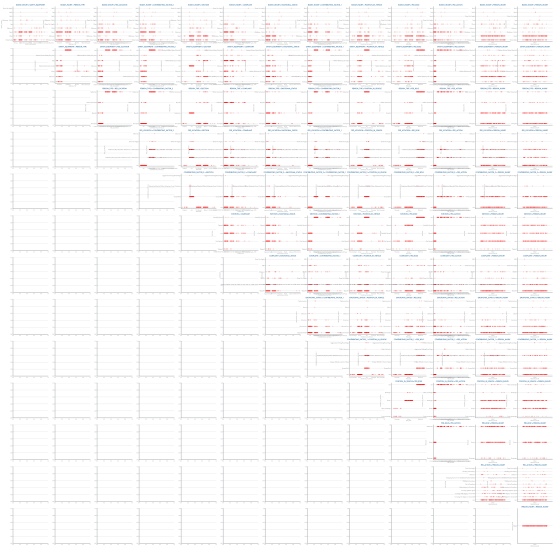
Furthermore, we present some of the granularity plots for the symbolic features that at the time of the granularity lab we thought were necessary. Ultimately we ended up applying granularity changes for all the symbolic features, however here we present the ones that we thought of multiple ways how to 'group'. For the remaining symbolic features that we encode, we didn't try multiple encodings. We also remark that we aren't presenting the charts for the original dataset, but yes for the one after applying some changes, like dropping all the ID's features.

Sparsity and correlation

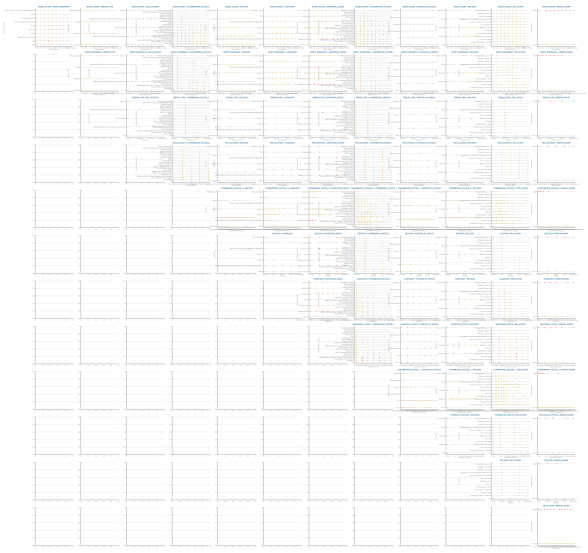
Sparsity study Injured



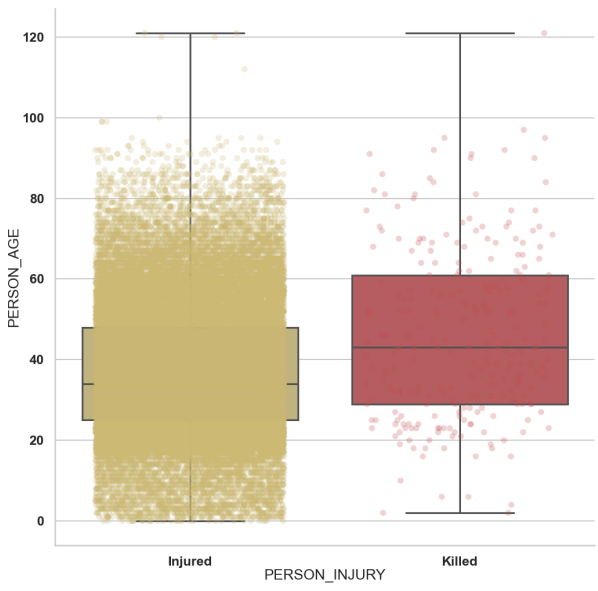
Sparsity study Killed



Sparsity study all:



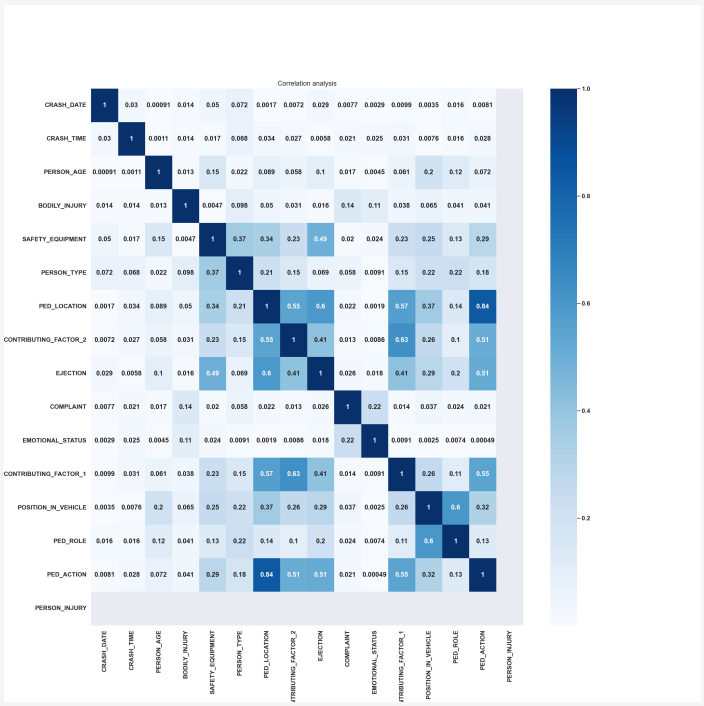
Sparsity study 'original' numeric feature:



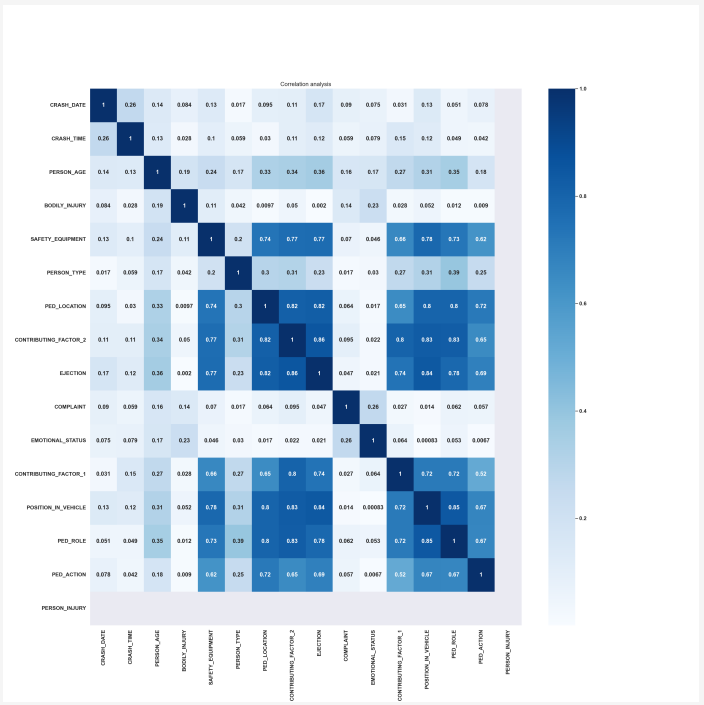
Remark: we are aware that the resolution of the file may not be the necessary one to be able to perform a level of zoom that enables the precise understanding of the sparsity plots, however, and to not overly-increase the size of the report, we think this is a good trade-off since the main idea can be obtained.

Correlation Injured:

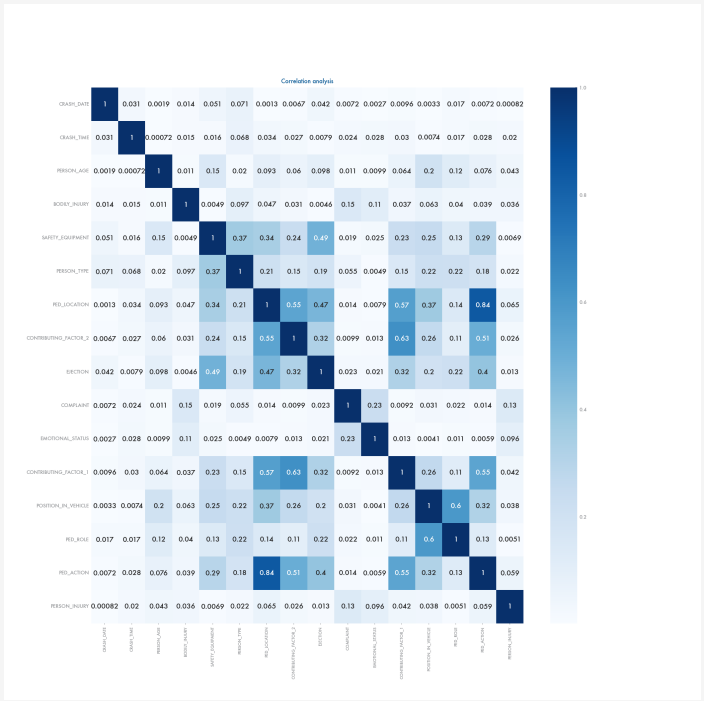
Correlation Killed:

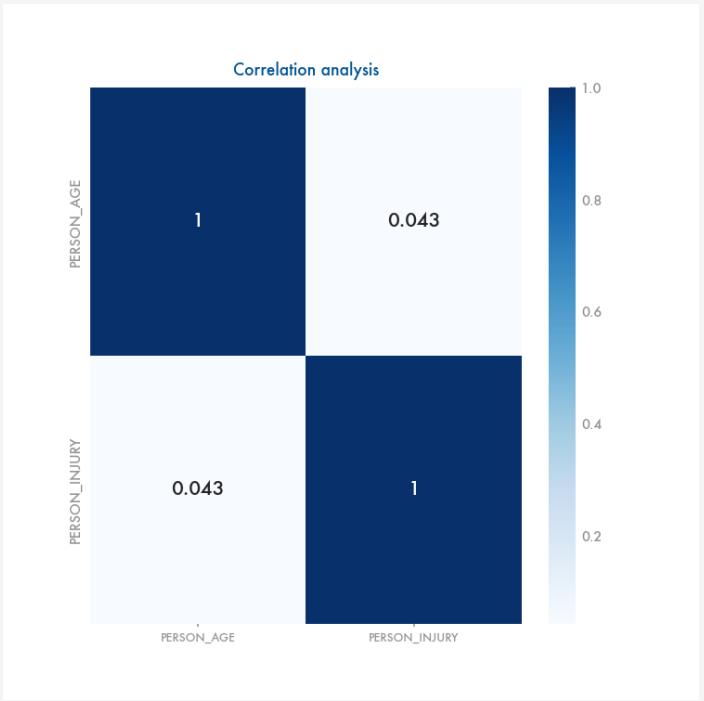


Correlation all:



Correlation 'original' numeric feature:





Granularity changes

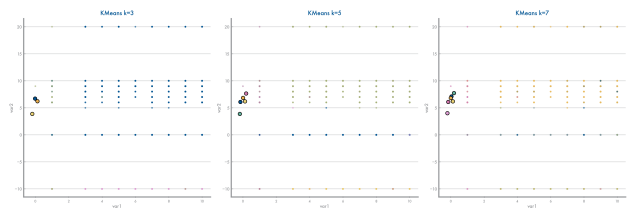
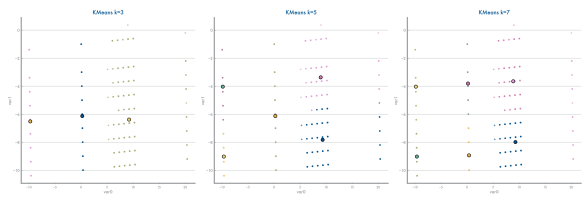
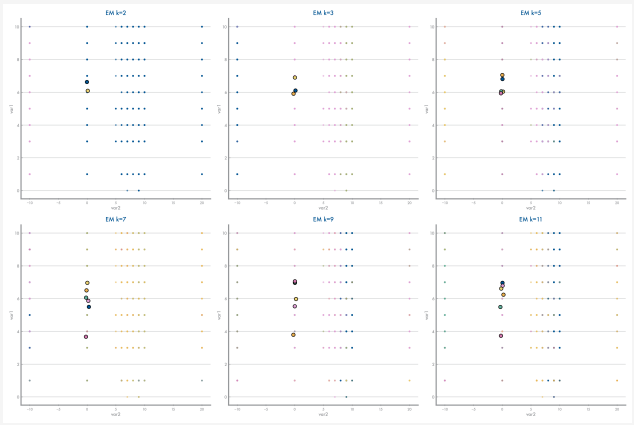
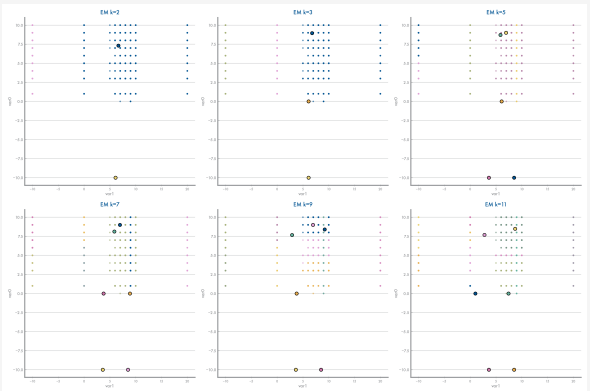
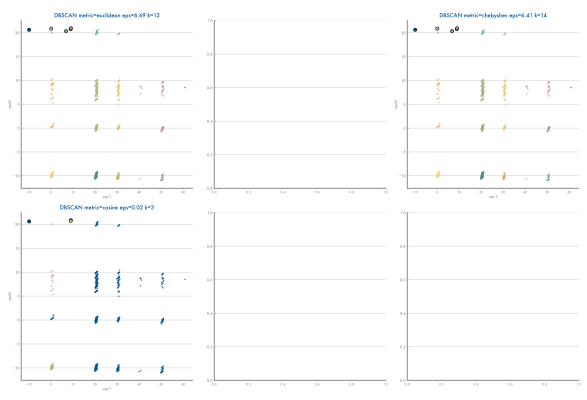
- POSITION_IN_VEHICLE grouped depending on the risk given the position. Higher the value, higher the risk;
- EMOTION-STATUS, EJECTION, PED_LOCATION, PERSON_TYPE grouped depending on the health risk;
- For the EJECTION feature, EJECTED and PARTIALLY_EJECTED values were grouped into 'EJECTED'.
- BODILY_INJURY grouped with 'major areas'. For example, ['Shoulder - Upper Arm', 'Elbow-Lower-Arm-Hand'] were grouped into 'Arm'. 'Head', 'Face' and 'Eye' were grouped into Head. For each one of these groups, a value was given according to its health risk.
- PED_ACTION grouped into major groups, for example ["Crossing With Signal", "Crossing, No Signal, Marked Crosswalk"] -> "Crossing With Signal / Crosswalk". For each one of these groups, a value was given according to its health risk.
- SAFETY_EQUIPMENT were also grouped and valued as PED_ACTION and BODILY_INJURY. For example, every unique value with the sentence 'Air Bag', was grouped into just 'Air Bag'. In this case, the values were given depending on the protection given (the higher the value, higher the protection).
- CONTRIBUTING_FACTOR_1 we were also able to create major groups, passing from 46 unique values to 23 (similar to CONTRIBUTING_FACTOR_2). In this case, the values are also attributed depending on the risk presented to the population.
- For **COMPLAINT**, we talked with a 4th year medicine student, from the Faculty of Medicine of the University of Lisbon (thank you João Graterol!) and we were able to group each one of the complaints into major groups using official guidelines (dges, and ebm guidelines): ["Green", "Yellow", "Red", "WHEN PATIENT MUST BE TAKEN TO THE SITE OF CARE WITH URGENCY", "CRITICAL WITH ON-SITE EMERGENCY TREATMENT"]. Then, for each group, we gave a value depending on its health impact.

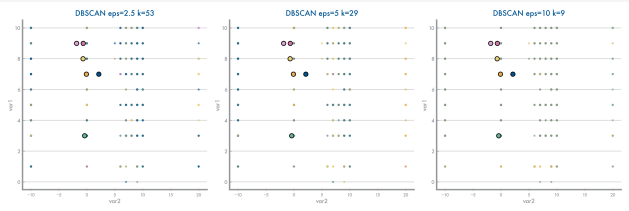
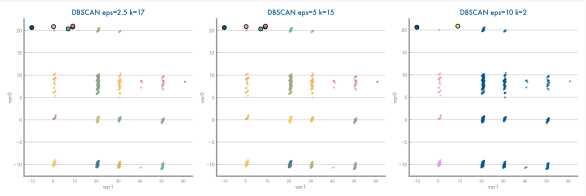
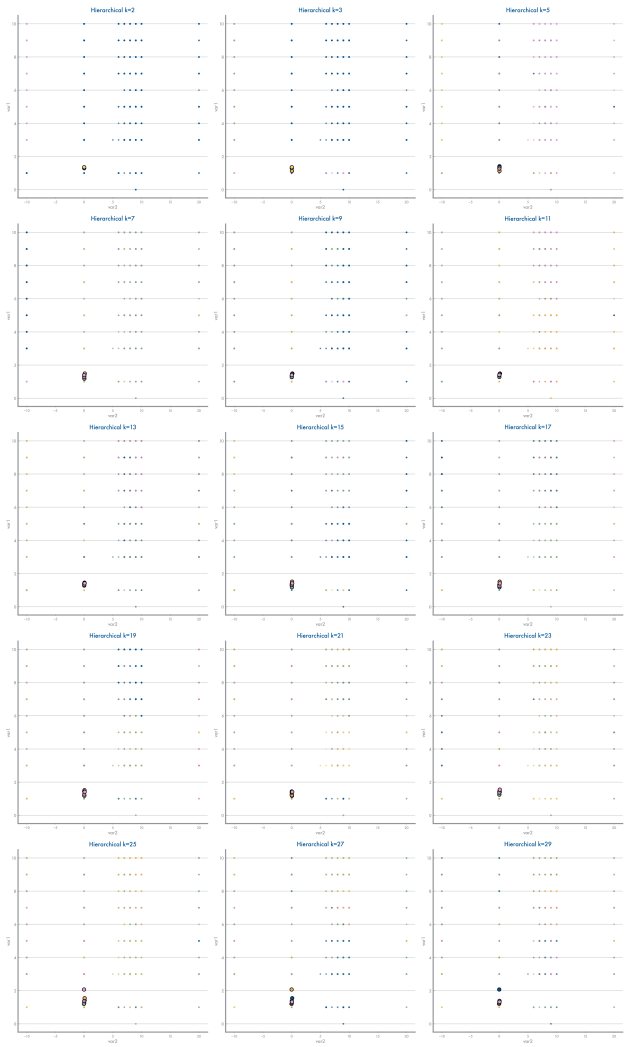
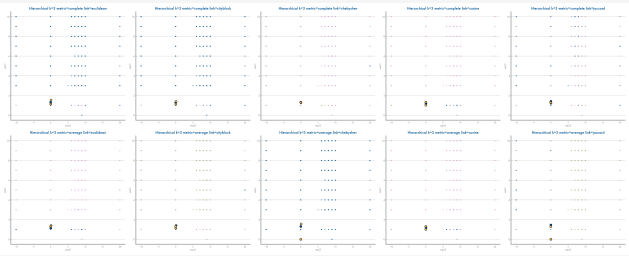
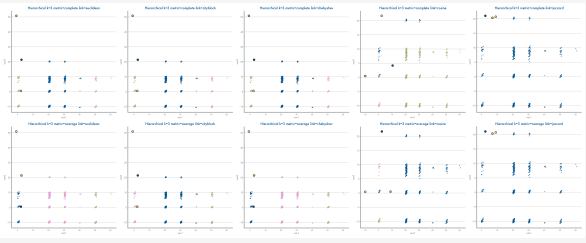
Google sheets file with all the details about the encoding: ***link***.

Custom missing value imputation

```python
def apply_set1_imputation(data):
    for index, row in data.iterrows():

        if row["PERSON_TYPE"] == "Pedestrian":
            if pandas.isna(row["SAFETY_EQUIPMENT"]):
                data.at[index, "SAFETY_EQUIPMENT"] = "Unknown"

            if pandas.isna(row["EJECTION"]):
                data.at[index, "EJECTION"] = "Unknown"

            data.at[index, "POSITION_IN_VEHICLE"] = "Does Not Apply"

            if pandas.isna(row["PED_ACTION"]):
                data.at[index, "PED_ACTION"] = "Unknown"

        else:
            if pandas.isna(row["SAFETY_EQUIPMENT"]):
                data.at[index, "SAFETY_EQUIPMENT"] = "Unknown"

            if row["PERSON_TYPE"] != "Occupant":
                data.at[index, "PED_LOCATION"] = "Does Not Apply"

            if pandas.isna(row["CONTRIBUTING_FACTOR_2"]):
                data.at[index, "CONTRIBUTING_FACTOR_2"] = "Unknown"
            if pandas.isna(row["CONTRIBUTING_FACTOR_1"]):
                data.at[index, "CONTRIBUTING_FACTOR_1"] = "Unknown"

            if pandas.isna(row["EJECTION"]):
                data.at[index, "EJECTION"] = "Unknown"

            if pandas.isna(row["POSITION_IN_VEHICLE"]):
                data.at[index, "POSITION_IN_VEHICLE"] = "Unknown"

            if pandas.isna(row["PED_ACTION"]):
                data.at[index, "PED_ACTION"] = "Does Not Apply"

            if pandas.isna(row["PED_LOCATION"]):
                data.at[index, "PED_LOCATION"] = "Unknown"
```

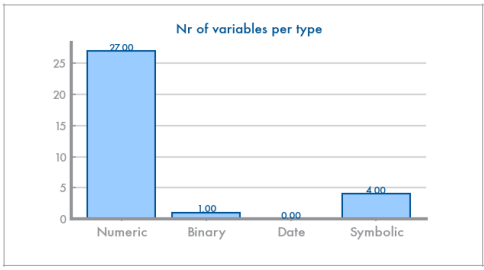
Clustering

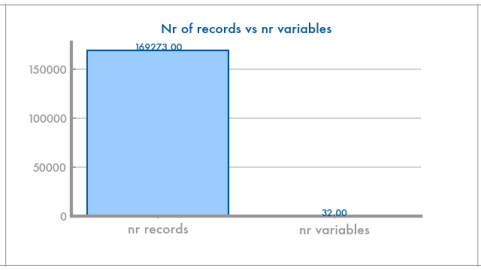
	Before PCA	After PCA
K-Means		
EM		
DBSCAN - Metrics		

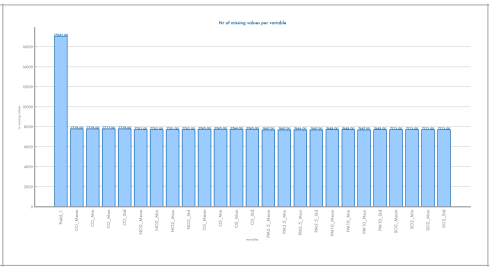
	Before PCA	After PCA
EPS		
Hierarchical		
Hierarchical - Metrics		

Dataset 2 - Other images and informations:

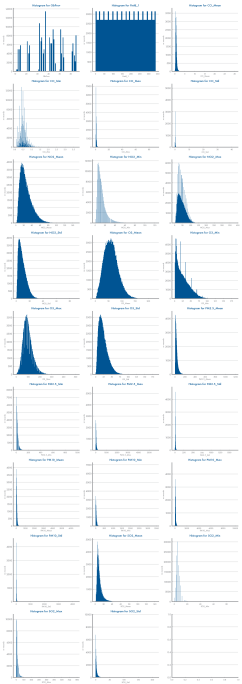
Dimensionality

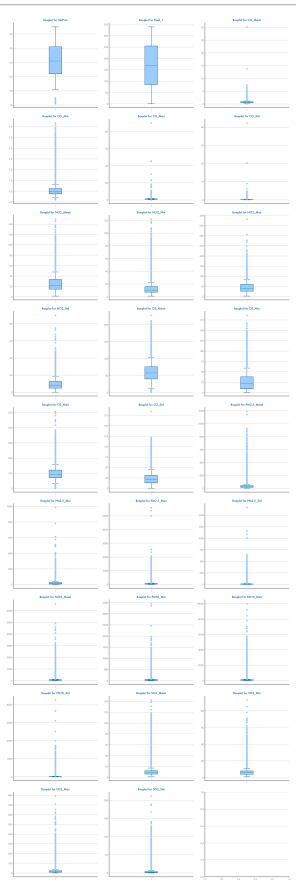


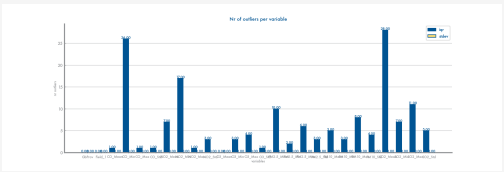


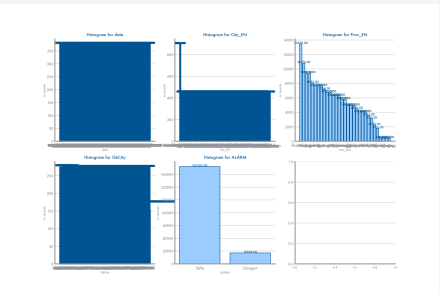


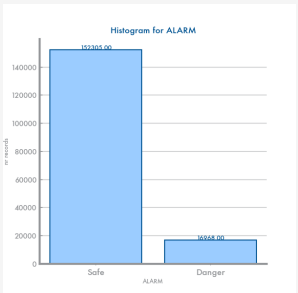
Distribution

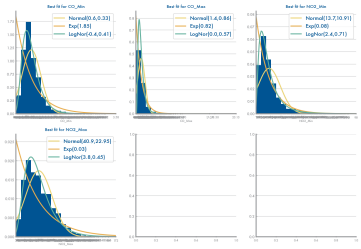


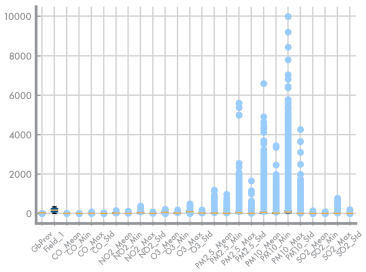




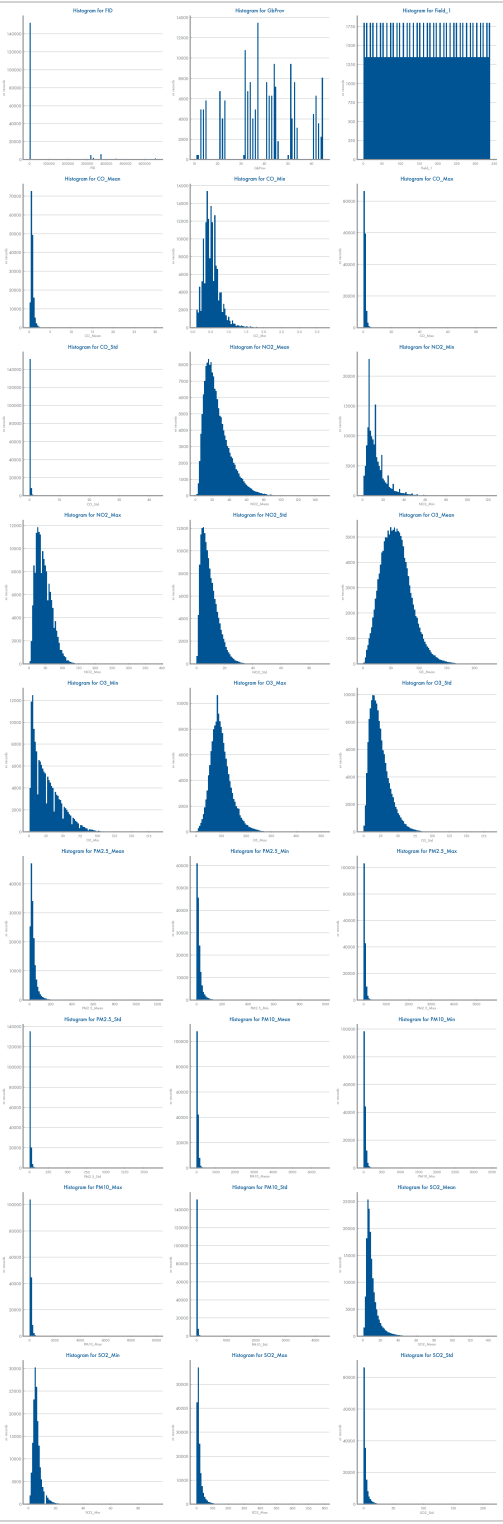


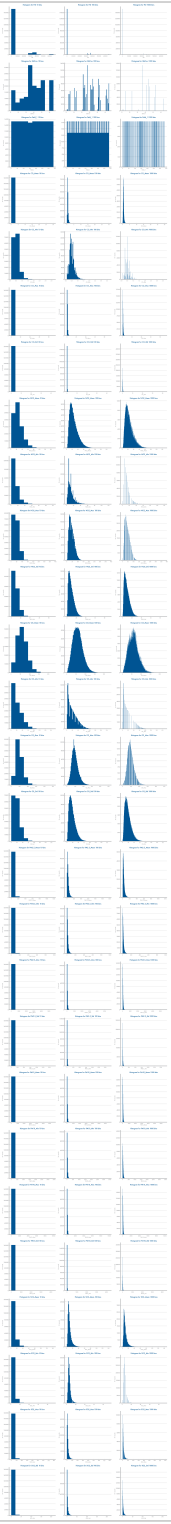


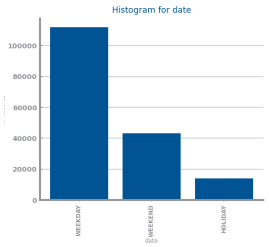




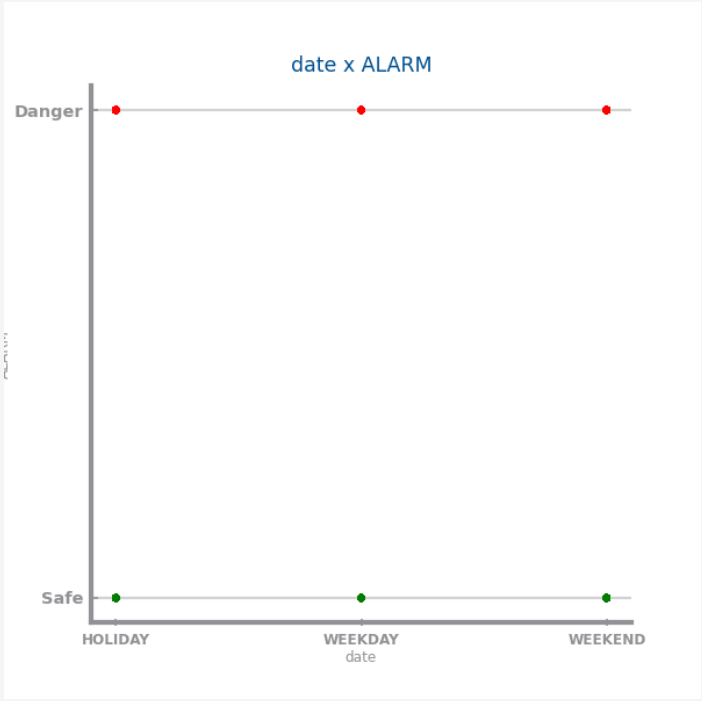
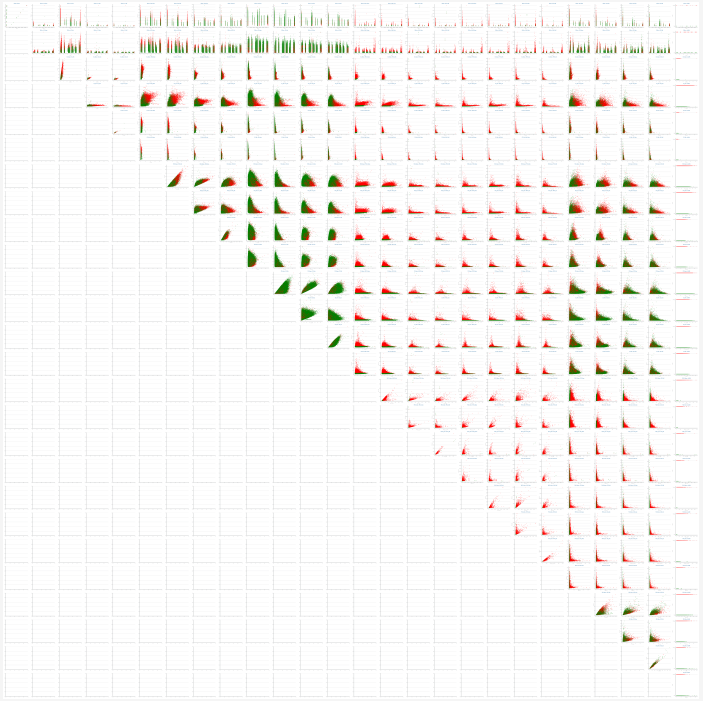
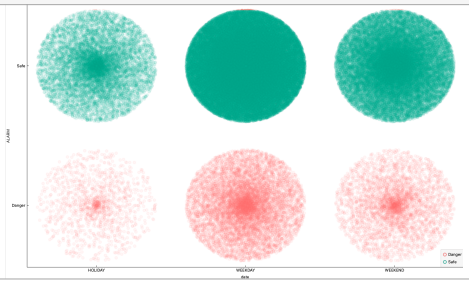
Granularity



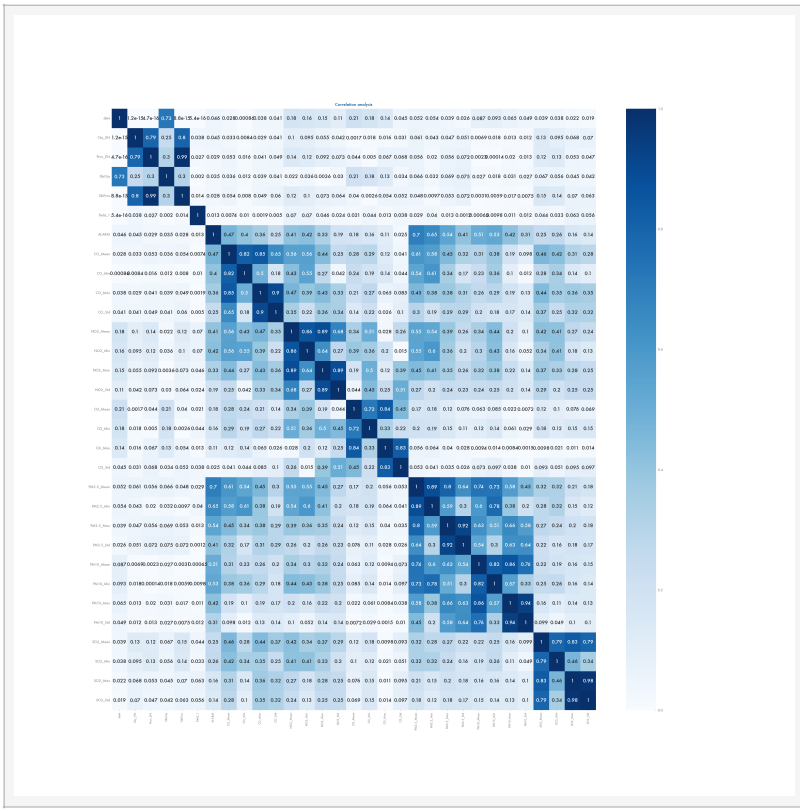




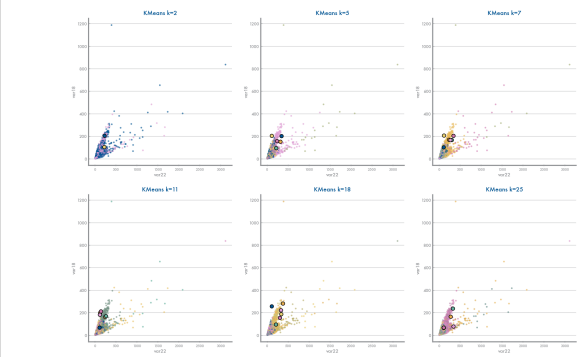
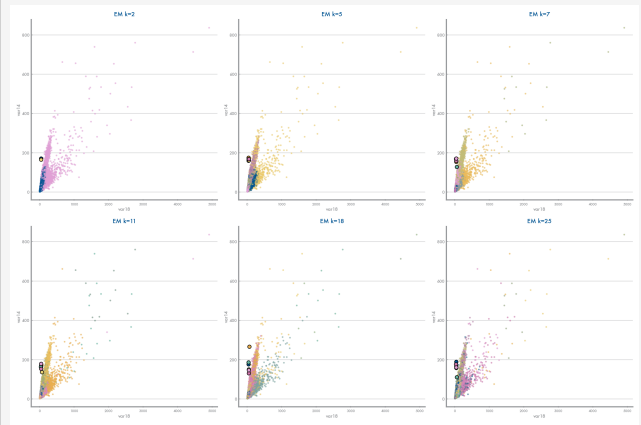
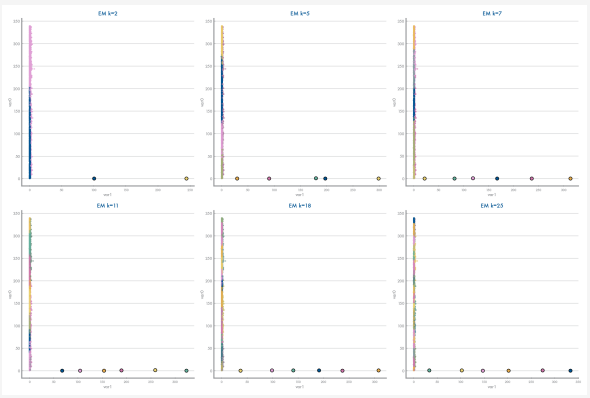
Sparsity and correlation

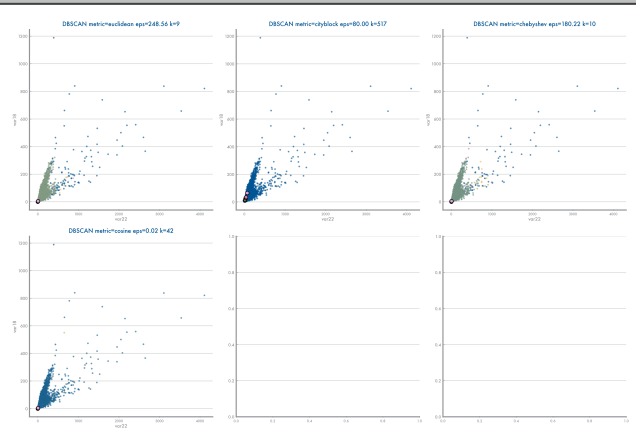
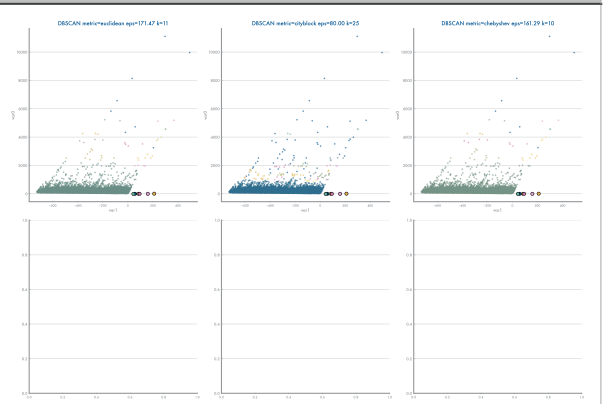
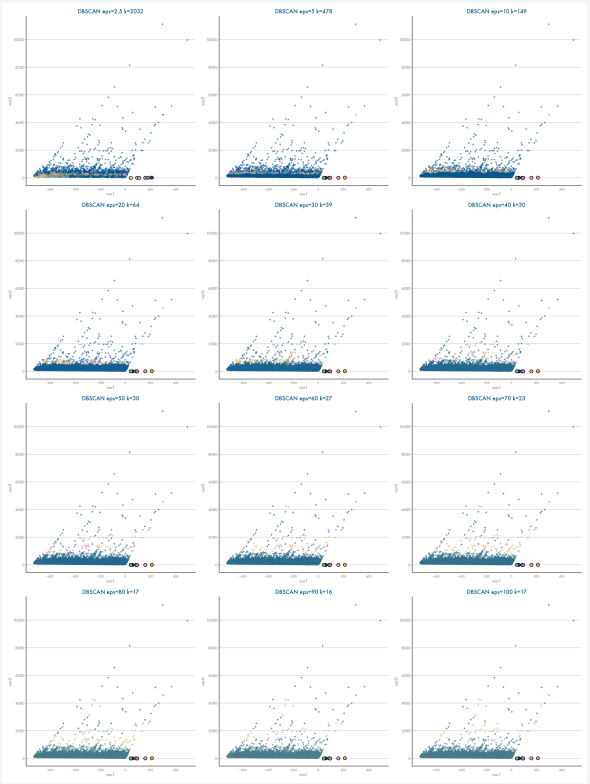
Sparsity symbolic	Sparsity numeric
	
Without using jitter, it is not possible to understand the differences between the multiple dates for each one of the values of the Target feature. This we also present, on the right, the plot with jitter!	

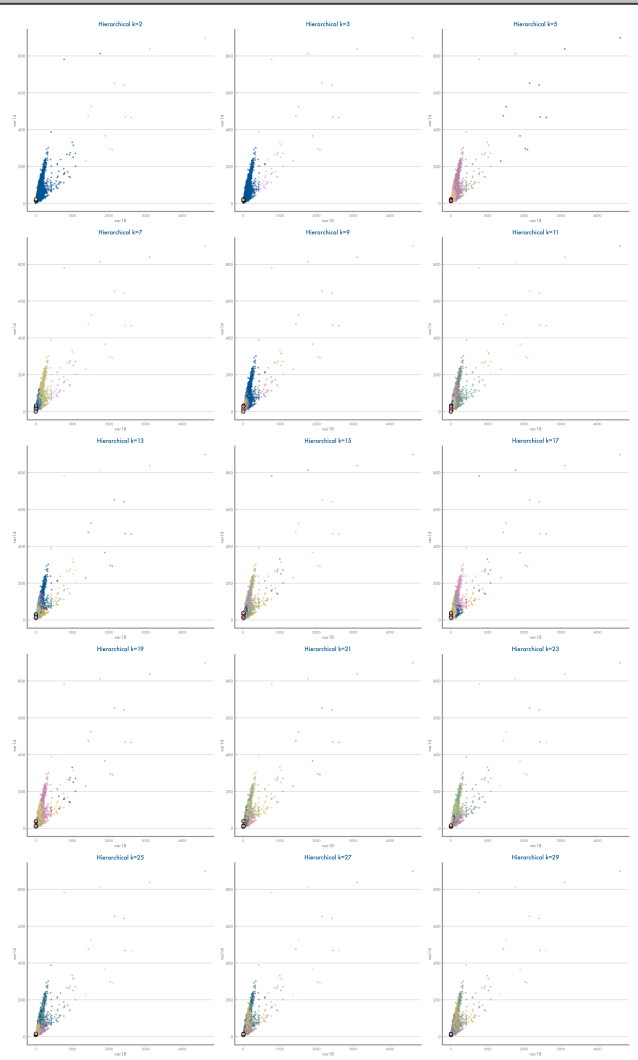
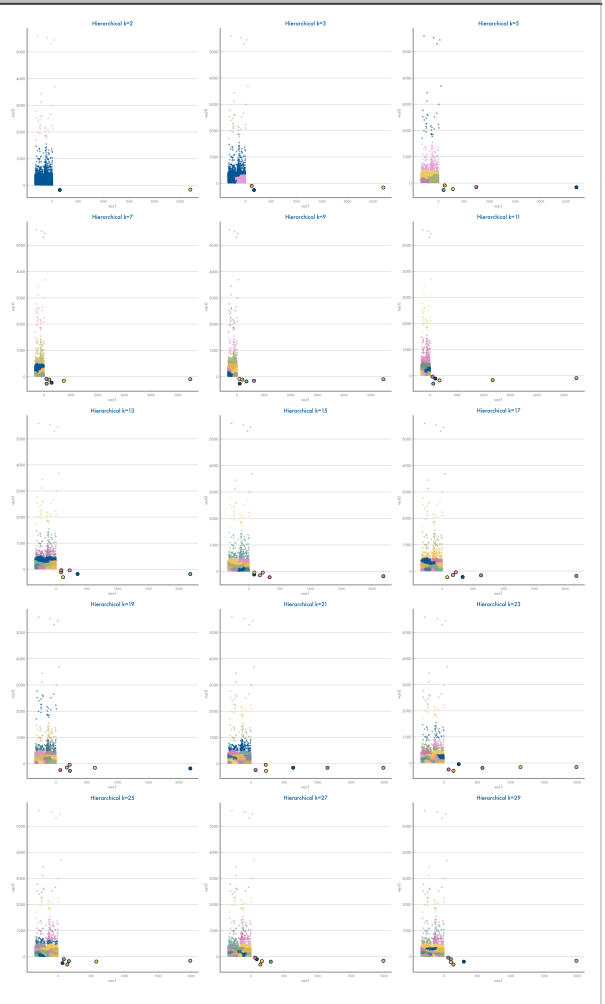
Correlations:



Clustering

	Before PCA	After PCA
K-Means		
EM		

	Before PCA	After PCA
DBSCAN - Metrics		
DBSCAN EPS		

	Before PCA	After PCA
Hierarchical		
Hierarchical - Metrics