# DATA SCIENCE PROJECT

Guilherme Pires
Miguel Ferreira
João Cruz

# Data Science project

## Introduction

This project objective is for us to critically analyse two datasets. We are challenged to understand the data we are dealing with, process it, create models and then hypothesize what could be done to improve them. Since we are following the **CRISP-DM** process for each dataset, this report is divided into two major parts, one for each dataset, **NYC Motor Vehicle collisions to Person** and **Air Quality in China**, respectively. **Let's start!**

## Dataset 1

### Data profiling

For the first dataset, **NYC Motor Vehicle Collisions to Person**, regarding supervised classification methods, we are going to use as target the variable **PERSON_INJURY**, and, regarding forecasting, we will be using **NR_COLLISIONS**. We want to be especially alert to these features when analysing our data.

### Data Dimensionality

As it is possible to see in [Figures 1-3 in the appendix], we have almost **6568** records with missing values for the **VECHICLE_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 for us to deal with: how to treat the high number of records with missing values (can they be simply dropped, or should we don't something else? Regarding the symbolic variables, how are we going to dummify 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.

### Data Granularity

Before analysing the data granularity, we need to understand how our data is distributed.

The first thing we are able to notice is that there are some **incorrect values** (not noise!) on the PERSON_AGE feature, as the maximum value is 9999, and, in fact, we have 29 records with PERSON_AGE > 140 & < 0. For the remaining of the report, these incorrect values were simply removed. Nonetheless, a note must be done that these removed records were all 'Injured' in the target variable. In a bit we are going to see that this is the majority class, and that this removal doesn't impact - at all - our work (the features distributions).

This first work, of analysing the distributions, is particularly important for us to understand what values for each feature may be further divided or grouped. For each feature we compare its distribution when the records are Injured and when the records are Killed (from the target feature). Note that on the appendix is possible to see this analysis for all the features, here we are just going to present some.

Regard time hierarchy, some ideas arise:
- CRASH_DATE records could be grouped into 'Weekday', 'Weekend' and 'Holiday'. (We discovered two python libraries that could do exactly this for both USA and Chinese holidays, for both datasets).
- CRASH_TIME could also be grouped into 'Dawn', 'Morning', 'Lunch time', 'Afternoon', 'Dinner time' and 'Night'.

Both suggestions didn't arise from nothing. They are quite intuitive. We may think that, for example, we would find more crashes on Weekends, Holidays and Fridays. The same for the CRASH_TIME, probably there would be more crashes at Dinner and Night time slots. As we will figure out during this report, what one may think is **intuitive**, more often than not, isn't what we think.

Regarding granularity, there is a lot of ways for us to tackle it: for example, BODILY_INJURY could be grouped depending on the severity of the issue. SAFETY_EQUIPMENT could be grouped as well, depending on the part (or parts) of the body that a given equipment protects. For the PERSON_TYPE, Bicyclists and Motorised could also be grouped, if we view them as individuals who are using a transportation vehicle with only two heels. For all other variables (for instance CONTRIBUTING_FACTOR_1 and _2 which have many unique values), we don't really see any way of grouping the data and gathering more substantial and completer groups...

As we can see in [Figure 1], there really isn't any way for us to group the CRASH_TIME in any meaningful way that preserves the correct distribution. If we do it, we obtain [Figure 2], which won't give us, we think, enough detail. We must also be aware that reducing unique values on Symbolic variables, just because we can, isn't necessarily what we should do. Obviously, one possible grouping could be to make two groups, one from 00h-> until 11h and another with the remaining hours. This would give us the distinction we are looking for on Figure 1 for the yellow figure. However, this wouldn't be correct as we would simply be manipulating our data to lead to the results we desire. For example, regarding the feature BODILY_INJURY we observe its distribution in [Figure 2]. As we can see there's clearly a majority of records Killed (read) with injures on "Head" and "Entire Body". For Injured individuals, the distribution is much more well behaved, in sense that there isn't a majority of injured individuals with injuries just on Head or Entire Body. This means that we could try to make a grouping between body parts like: Head, UpperBody_Front, UpperBody_Back, LowerBody and Entire Body. However, we won't proceed with this since the amount of granular detail we would loose is not a good trade-off.

Similarly, for SAFETY-EQUIPMENT, [Figure 3], we are able to observe that clearly a majority of the records where there's no use of any safety equipment, the target feature is Killed. Same for who uses "Helmet (Motorcycle)". This **may** point to the fact that a majority of motorcycle accidents are much more deadly than car accidents (since for Injured, represented with Green colour, a majority of the people used Lap Belt). A possible grouping , and that we will do, would be to group everything that has 'Helmet' to the group 'Helmet', and everything with 'Air Bag' to 'AirBag'. Still regarding the distributions, there's a lot of curious information that should be retained! For example, if the outcome was 'Killed' then it is much more probable for the PERSON_TYPE to be 'Occupant' or 'Pedestrian', if it is 'Injured' then it is much more problem for the individual to be 'Occupant'. Similarly, for the EJECTION feature we can see that 'Ejected' and 'Not Ejected' is much more probable for 'Killed', and 'Not Ejected' is what is more probable for 'Injured'.

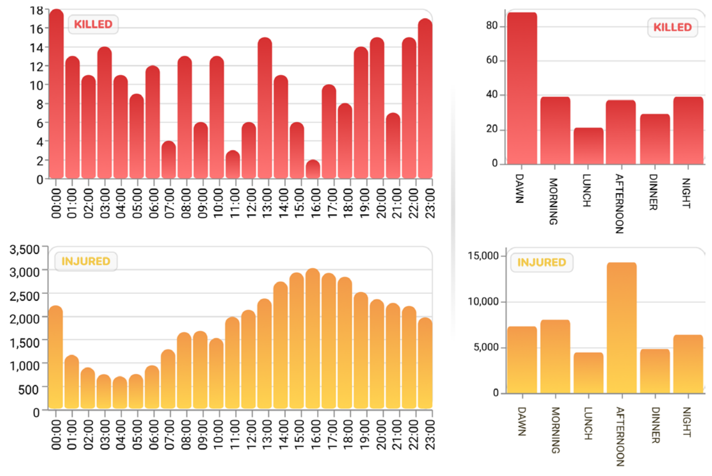


**Figure 1:** Left: Difference on distributions between Injured and Killed individuals regarding CRASH_TIME. Right: Difference on distributions between Injured and Killed individuals after grouping.

Where we can also retrieve valuable information is in the COMPLAINT and EMOTIONAL_STATUS features. For example, for Killed records, 60% and 20% were respectively for 'Internal' and 'Crush Injuries'. For 'Injured', 60% is for 'Pain or Nausea', and 10% for 'None visible'. One possible grouping would be to classify these different complaints depending on their severity. After we talked with a colleague of ours, that is on the 4th year of the medical course, it is his opinion that there isn't any clear way of grouping these values...

For the EMOTIONAL_STATUS, for Killed, almost 90% of the records are for 'Apparent Death' and 'Unconscious'. For the Injured records, 92% are for 'Conscious'. However, we can't find any reasonable grouping to be made.

We took almost 2 pages and an half on data granularity and its distribution, because not only this allows us to **understand** much better the possible problems and results that will appear along the report, but also understanding what are the most important values in each feature allows us to do a much more informed dummification of these **nominal** symbolic variables.

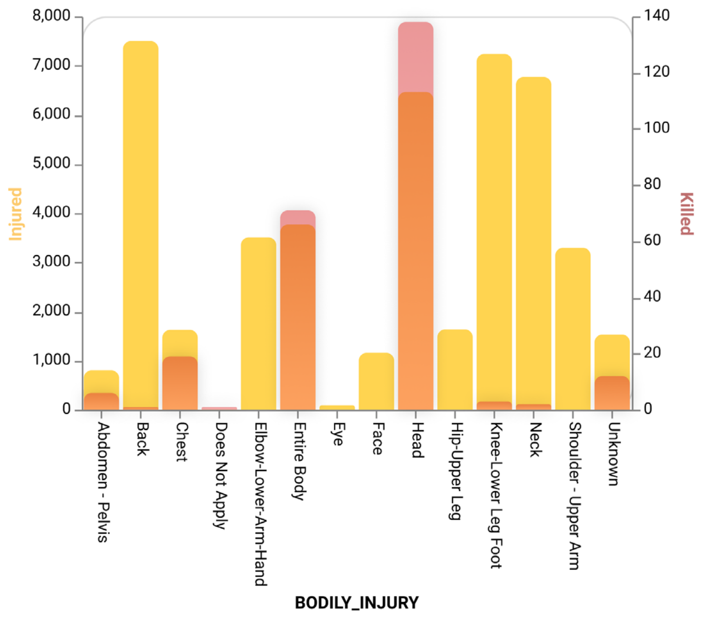


**Figure 2:** Difference on distributions between Injured and Killed individuals regarding BODILY_INJURY.

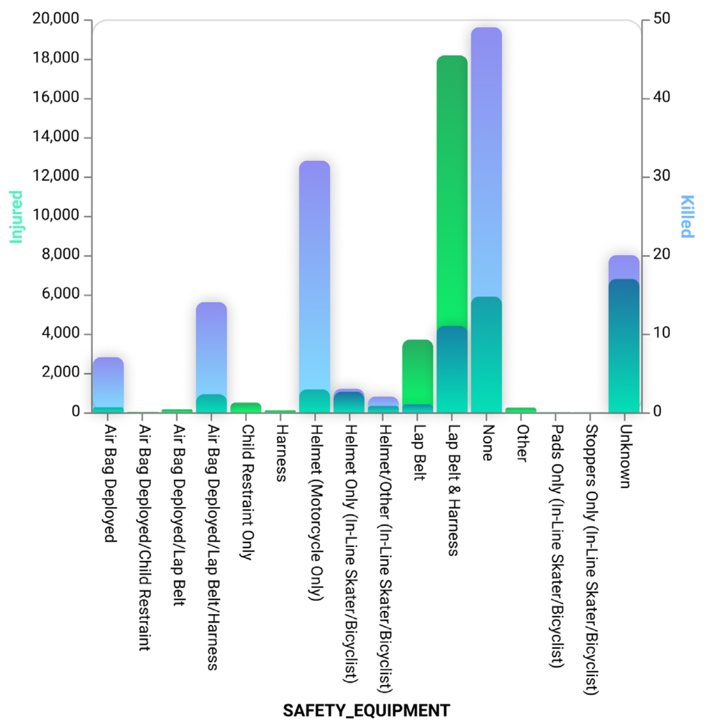


**Figure 3:** Difference on distributions between Injured and Killed individuals regarding SAFETY_EQUIPMENT.

In sum, we have that:
**-** We will use time hierarchy only for CRASH_TIME and not for CRASH_DATE since we don't **think** it would give us enough detail - or at least the amount of detail we want. **ALTERAR EM CIMA, DISSE QUE NAO IAMOS FAZER**

**-** We perform granularity for the SAFETY_EQUIPMENT since we will loosing almost no detail and gain generality, and, it **may** facilitate our predictive models work.

**-** Regarding the data distribution, type, domain and range, the images needed to perform the respective analysis are all present in the appendix **ADD THE NUMBERS**. There's not much to say: for the symbolic variables, the distribution is, for all variables, very similar to a log-normal function. For the PERSON_AGE the distribution is similar to a Gaussian one. In terms of ranges and domain, there's also nothing of much importance to describe.

**-** Regarding **outliers**, we won't be detecting the outliers now, since we think it will be better to do it after the dummification step.

## Data Sparsity

Regarding the numeric variables, there's no relevant information that we can retrieve from their sparsity analysis. Remember that the numeric variables are PERSON_AGE and all other features that are ID's. In fact we don't gain any valuable type of information with the ID's features. In some cases, ID's may withhold some type of structure, in the case of this dataset we don't observe it, remarking this features as useless.
For the symbolic features, its sparsity analysis give us the information that we have presented on the previous section (Data Granularity and Dimensionality). Being pragmatic, the sparsity graphs are quite dense, making it more difficult to do any type of analysis. Even more since this dataset is so unbalanced (more on that later).

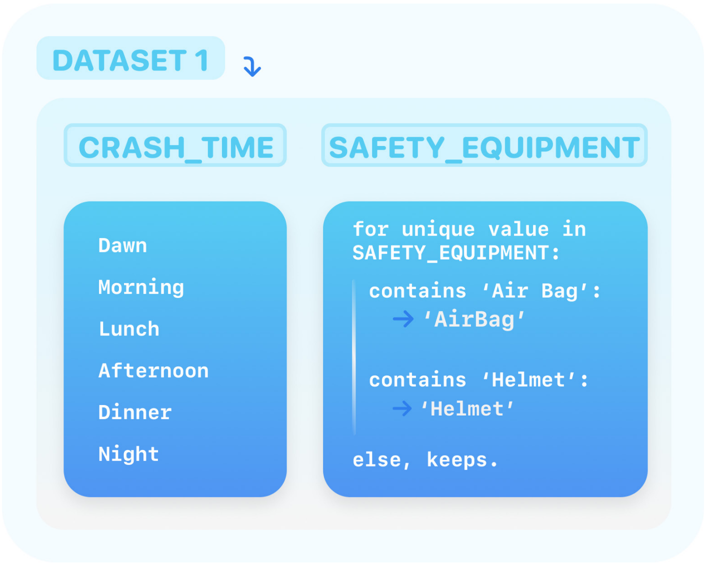
What we can, though, observe in the sparsity graphs [**IN THE APPENDIX**], is that our data is quite sparse for both Injured and Killed values of our target feature for almost all relevant (excluding ID's) pairwise combination of features. Even though the data is sparse, we have linear 'spots' /clusters of data points for both outcomes. This already tells us that we will have positive correlations between some of the features. What we can also conclude by this is that we have, even though we are dealing with a lot of features and records, the data isn't uniform, and still has 'structure', so not every point is the same statistically speaking. We are not under the curse of multidimensionality. **AREN'T WE?**

Regarding correlations, there's two curious approaches: analyse the correlation matrix for the entire dataset (both Injured and Killed records) and analyse two different correlation matrices, one for Killed and another for Injured records.

Starting by analysing the correlation matrix for the entire dataset, we identify little correlation between most of the features. Only identifying it for (PED_LOCATION x PED_ACTION), (PED_ROLE x POSITION_IN_VEHICLE), (PERSON_TYPE x EJECTION), and a few more similar combination of features. Namely, we observe correlation between the following set of features: [PED_ROLE, PED_ACTION, CONTRIBUTING_FACTOR_1, EJECTION, CONTRIBUTING_FACTOR_2, PERSON_TYPE, PED_LOCATION, SAFETY_EQUIPMENT]. This already points out that when we do feature selection, probably some of the features can be dropped since they can be described by others with little loss of information. More importantly two notes must be taken:

**1)** There's little to no correlation between the target feature and all other features.
**2)** The majority of the high correlations found, and described early, have quite intuitive domain knowledge reasons to appear. For example, for the (PED_ROLE x POSITION_IN_VEHICLE), there is the PED_ROLE 'Driver' and also the POSITION_IN_VEHICLE 'Driver'. Similar reasoning can be made for some of the other correlations.

The second part of this analysis that is important is to understand that when we separate the dataset in two - for Killed and Injured -, the correlation matrices are quite the opposite. There's a lot more high valued correlations for the Killed (minority class) values. This is due to the fact that the data is not only much less sparse, we have a low number of records but also because, intrinsically, there are indeed intuitive patterns for the features of these records. For example, that it is much more likely for one to be Killed if driving a Motorcycle, and that this probability will be even greater if there's no use of Helmet (this is one of the many correlations we can find). This corroborates our findings in the previous section.



DATASET 1
CRASH_TIME
SAFETY_EQUIPMENT
Dawn
Morning
Lunch
Afternoon
Dinner
Night
for unique value in SAFETY_EQUIPMENT:
contains 'Air Bag': → 'AirBag'
contains 'Helmet': → 'Helmet'
else, keeps.

**Figure 4:** Granularity transformations for the CRASH_TIME and SAFETY_EQUIPMENT features.

## Data preparation: Missing Values Imputation

As it is possible to see in [], we have a quite substantial number of records with missing values. From all features with missing values, four of them stand out: PED_LOCATION, CONTRIBUTING_FACTOR_2 and 1 and PED_ACTION.

After analysing and understanding our data, we were able to understand the reason behind the missing values. For the PED_LOCATION, there are missing values only for records where PERSON_TYPE is **not** 'Pedestrian'. And, well, this makes sense. Since you aren't a pedestrian, you can't have a pedestrian location. Every record that the PERSON_TYPE isn't 'Pedestrian', if the PED_LOCATION is null, we simply put 'NotApplicable'. Note that some records already are 'Does Not Apply', we will be transforming them (234 records) to 'NotApplicable'.

For the CONTRIBUTING_FACTOR_1 and CONTRIBUTING_FACTOR_2, every empty record is present when PERSON_TYPE is **not** 'Pedestrian'. In fact, only a few records (+/- 130) that aren't Pedestrian have CONTRIBUTING_FACTOR_1 and 2 filled. For those that haven't, we put 'NotApplicable',. Lastly, for PED_ACTION, if the PERSON_TYPE is Pedestrian, then we fill missing values with 'Unknown', otherwise, if it isn't Pedestrian, we fill with 'NotApplicable'.

For all other features( where there are few missing values), if the PERSON_TYPE is Pedestrian, excluding the features we've talked before, when there is a missing value we put 'NotApplicable'. The exception is for POSITION_IN_VEHICLE, where we fill missing values also with 'NotApplicable'.

For all other PERSON_TYPEs, if SAFETY_EQUIPMENT, EJECTION or POSITION_IN_VEHICLE is missing we put 'Unknown'. If PED_LOCATION is missing, we fill with 'NotApplicable'.

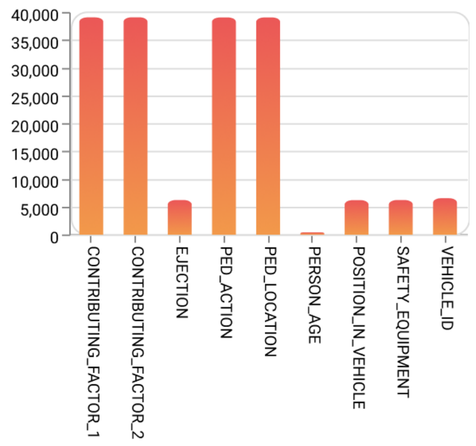


**Figure 5:** Number of missing values per variable.

Lastly, regardless of the PERSON_TYPE, if the VEHICLE_ID is missing, we fill it with '-1'.

We end up also dropping all values where PERSON_AGE is missing and where PERSON_SEX is 'U'. When AGE is missing, the target class variable is always 'Injured', as such we aren't losing information in our minority class nor changing our distributions (we are talking about 421 records).

## Dummification:

This dataset has 15 Symbolic variable which turns this process, dummification, not only more important but harder. It is true that not all algorithms, for classification, need every variable to be numeric (there's Categorical Naive Bayes), however, the majority can't handle symbolic data. Another difficulty that arises is the differentiation between nominal and ordinal variables. Most of our symbolic variables are, indeed, nominal and not ordinal. Or, at least, we would need to be quite creative to turn them into ordinal ones. For example, for safety equipment, we could think of a natural order compreenhending the 'area' of protection of the equipment, the bigger the more important... However we may be stretching this concept too much!

The problem with dummification, or one-hot-encoding, is that, unless we group our variables way more than what we did, we would end up with hundreds of columns, given the large number of unique values for each  feature. What we are going to do is use the Ordinal Encoding method. This was we will be doing a map between the categorical values and an integer (that later will be scaled). We know, however, that this integer value matters! Let's imagine a feature with 20 unique values, one of the values will be mapped to 1 and the other to 20 (or 0/19), and for some of the classification algorithms, the value 20 may have more weight than the value 1. This way, and by using the sklearn OrdinalEncoder class, for each feature that we encode, we are also going to pass a variable `categories`. This variable contains the order that we want to give to our variables. This way we can, by observing the data distribution for both Injured and Killed records, have an encoding that embodies the distribution that we observe. Needless to say that this is not perfect. Features that appear less, and that therefore we may attribute a lower integer value, may loose some of its importance, something that with OneHotEncoding wouldn't happened since the distance between each and every one of the columns of the unique values would be 1. However, we think that this approach is indeed the best trade-off between the number of features and the maintenance of the natural distribution and importance. This importance is, for example, saying that, for the BODILY_INJURY feature, [figure 2], the 'Head' is the most important value (or one of), because of the number of occurrences in the dataset.

One remark that must be done is that there is a maybe large pitfall to this approach: we may end up over-fitting our data. That's something we will need to be cautious about.

We perform ordinal encoding to: BODILY_INJURY, PERSON_TYPE, EJECTION, COMPLAINT, PED_ROLE, PED_ACTION, EMOTIONAL_STATUS and SAFETY_EQUIPMENT. The mapping we use for each unique value for each feature can be seen in the appendix [**IN THE APPENDIX**].

To PED_LOCATION and POSITION_IN_VEHICLE, we perform simple integer mapping with OrdinalEncoding. Note that even thought there is no order between the elements (even by trying to give more importance to the values more observed in either the killed or Injured distributions), but, performing One Hot Encoding would add 14 new columns/features. Ultimately, after we tested both combinations, either using Ordinal Encoding or One Hot Encoding, on four distinct metrics: Precision, Recall, Area Under ROC and F-1, Ordinal Encoding always performed the better.

## Outliers Imputation

We now have a problem regarding outliers imputation. Since we have performed OrdinalEncoding where, for some of the columns in each feature, we've manually defined an higher value for some of the features, it may happen that some features, due to the higher value they have, are now regarded as outliers. The fact is that some points that are now detected as outliers with both the IQR and stdev criteria, in reality aren't.

The problem with our approach, without using OneHotEncoding, is that IQR and stdev aren't sufficient to handle our data anymore. Let's not forget that this first dataset had a majority of symbolic features. For the second dataset, since it won't be the case, we can use these more traditional and very robust methods. For now, let's be creative.

We will tackle this in the following manner: for the previous existing numeric variables (like PERSON_AGE), we are going to use the stdev criteria or the IQR. For the remaining features, the symbolic that are now numeric, we are going to use the IsolationForest by sklearn. Why this method and not another? Great question! Since our data, even though was continuized, it still is discrete in terms of the values it has (they are all natural numbers), IsolationForest is the best model for this type of data due to the partition style method used.

**1)** Firstly, for the numeric variables, we aren't doing outlier detection for ID's since we won't be working with them and since there is not proper a concept of outlier for an ID. Secondly, for the remaining feature, PERSON_AGE, if we observe [Figure 6], and using IQR, 289 records are classified as outliers. With stdev the number jumps to 2163 records. What we observe is that, with IQR the lowest detected age is 83. With stdev, from every record between 0 and 3 of age, is detected, as well as every record starting from age 70. More importantly for us, we would be loosing, with stdev, 42 records where PERSON_INJURY is Killed. With IQR just 14 of the 289 are from Killed records.

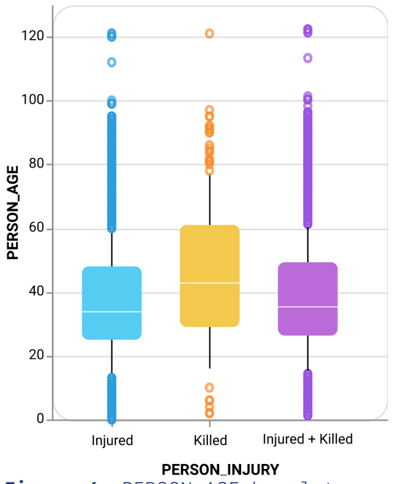


**Figure 6:** PERSON_AGE boxplot with outliers for PERSON_INJURY.

Obviously this could not be a problem depending on the Imputations methods we use, however, and since we are talking about a small percentage of data, and since the distribution is left-skewed, we are going to use the IQR method. Not only less records are detected as outliers, but also less Killed records are manipulated.

We won't be discarding the records since we would loose, even if only 14, records from the Killed class. Instead we are going to impute them with the mean from their respective classes. We could also do this to the stdev criteria, however, the 40 Killed records that we would be manipulating, either by replacing them with the mean value or by dropping them, would represent 16% of that class, furthermore we would also be loosing that class distribution towards higher aged records. This way we have a good trade-off with only 289 being manipulated, and only 14 from the Killed class.

**2)** For the other features, the symbolic ones that were continuized, we thought of using either the Winsorizing method, or the LocalOutlierFactor from sklearn. Using the LocalOutlierFactor we get that 118 Killed records are classified as Outliers, as such we won't be proceeding with this method, since we would be loosing too much information. We could also apply the outlier detection algorithm just for the 'Injured' and 'Killed' records, separately, however doing it for the Injured records, we obtain 6264 records classified as outliers. The thing is that if we look at [**APPENDIX**] we can clearly see very similar distributions to those on the full dataset, meaning that it didn't fully catch the meaning of an outlier for this dataset. Obviously, we could do a much more fine and detailed analysis to these results, however, we are afraid it might be out of scope of this project to go into such detail. The main remark is that since these were categorical features transformed into numeric ones, we can't assess outliers using normal methods, as such we need to use methods like this one that does assesses the distance between records. This is not robust as we want to preserve fine details and observations.

## Scaling:

As a rule of thumb, we know that the Standard Scaler is useful when our features follow a Normal distribution, and that the Min-Max scaling is useful when we need values in a bounded interval. What we also know is that due to the way weights are initialized in many machine learning algorithms, there are some advantages in using the StandardScaler (or Standardization).

For our numeric variables, excluding ID's, we only have PERSON_AGE. And its distribution is a normal one, as such we are going to use the standard scaler.

For all other features, namely the ones we've dummified in the previous section, this is trickier. The thing is that, since we've defined our categories (the order of the mapping), we've created distributions that are basically Beta distributions (a log-normal but to the right, with a negative skewness). For all these features, since the distribution is quite different from the Normal one, we are going to be using the MinMaxScaler from sklearn.

A note must be done that, on the Pre-processing data from sklearn, there are a multitude of methods and ways of tacking the scaling problem. However, since we don't have the time nor the space in this report, we just focus on StandardScaler and MinMaxScaler.

In the appendix, as requested, is possible to observe the difference between using **only** the StandardScaler and the MinMaxScaler.

## Balancing:

Balancing our data is of the up most importance since, as we have pointed out in the beginning, we have almost 180 times more Injured records than Killed ones. If we don't balance our data, what would happen is that any model that always predict injured, would be almost 100% correct. Indeed the model would have amazing performance, but, as in Medical classification problems, we are more concern about classifying correctly what is 'worse' or more important, which, in this case, is correctly classifying the Killed instances. If we classify as 'True' or 'Positive' being Killed, and as 'False' or 'Negative' being Injured, in this case, **Recall** is what we want to preserve the most. It is better to predict a Injured record as Killed than the opposite! The same way it is better to predict someone, for example, with a disease than not, because, even if the person doesn't have it, at least we didn't miss the opportunity to assess it.

With this said, we now have some challenges ahead of us. We may use SMOTE, Undersampling or Oversampling to balance our data. We can probably assume that between Oversampling and SMOTE, SMOTE would be the better choice. Between Undersampling and oversampling with SMOTE, both have problems.

With Undersampling we are going to keep only 0.0055% of our Injured records, which doesn't look promising at all (unless we observe that indeed with this data the distributions keep the same!). With oversampling we are going to try to artificially reproduce 45 thousand Killed records (with SMOTE) data from only 250. Neither option is good. They are both either loosing too much information, or over fitting the data. Even though SMOTE tries to create new records that aren't duplicates of the others, it is inevitable that it will happen since we are doing a 180 times magnification.

What we are going to do, however, is a mix of both options. We are, in one hand, going to perform DownSampling on our Injured records, and going to do SMOTE oversampling in our Killed records. Having special attention to keep the Injured and killed distributions, without adding any bias to any of the features. There's no rule of thumb in how many records should we keep for the undersampling of the Injured records. For that, we are going to keep 30% of the Injured records (approximately 13500). In the appendix [**ADD THE NUMBERS**], is possible to see two images, [**ADD THE NUMBERS**], regarding the distributions before and after the undersampling.

To conclude, in order to prevent overfitting and duplicating records (or near duplicated ones) we perform SMOTE oversampling for the Killed records, passing from 250 to 13500 records. And we also perform random undersampling for the Injured records, passing from 45000 to 13500 records.

Following the suggested methodology from the work assignment, we've test this against the other natural option: keep all records and perform oversampling, however we got worse results [**APPENDIX**]. A good advantage of our proposed balancing method is that if the results of the models are good, it means that we are less prone to overfitting, and we are indeed generalizing the 'idea' of being in an accident and resulting on injuries or deaths.

# nyc