**Introduction**:

Car driving seems to be a normal activity, something that can be done on a regular base, every day, to reach your job place, to go to the gym, to go shopping or even something you do to live in the case that driving represents the core activity of your job (think about taxi drivers, bus drivers and so on).

Nevertheless, making an activity so frequently does not mean that this activity lacks danger. Instead, making it so many times multiplies the possibility to encounter some unexpected events and some of them could be very dangerous.

Take a taxi driver as an example. His job is to take care of people transportation, doing the best route to reach their destination and last but not least to make the travel experience as relaxing and smooth as possible. What if you take a street and after a while you become stuck into the traffic caused by an incident? For the client it will not be for sure a good experience, it will become agitated and you will be able to serve less customers since you are going to spend much of your time among other cars, making very few progress towards your client destination.

And even worst, what if you decide to meet one friend of yours in the evening, you take your car to reach the bar where you are supposed to meet but unfortunately you make an incident. You do not know why it happened, maybe for the weather conditions, or maybe road conditions were not perfect.

Would not be great if there is something that will warn you based on some parameters? Something that will take into account some factors such as the weather and road conditions, warning you about the possibility of getting into a car accident and how severe it would be. So you will be able to drive more carefully or even change your travel if possible.

This is exactly the aim of this project capstone. The idea is to analyze many incidents, finding which variables are the most impacting in determining the event and to predict also the severity of it. With this tool in your hand, before getting into your car you will be able to analyze the actual conditions and discover how secure the travel could be. Isn’t it fantastic?

**Data Description**

The dataset is composed by 38 attributes (columns) and a total of 194673 entries (rows)

The Dtype of the attributes is mainly characterized by int64 (12) and object (22), with just some float64 type (4).

Some of the columns unfortunately are mainly composed by NaN values (EXCEPTRSNDESC, INATTENTIONIND, PEDROWNOTGRNT, SPEEDING and EXCEPTRSNCODE)

First of all attributes that are composed of mainly NaN will be discarded in order to not compromise the data replacing with some values like mean in case of replacement, and not to reduce too much the dataset in case of drop rows choice.

Many columns of the object type could be considered as categorical variables, meaning that to be useful they will be transformed into Boolean variables (dummies variables).

Then the selection of the variables that will be considered to train and test the model will be selected among the variables that will show to have higher correlation with the target variables. For this scope, data visualization tools and data analysis will be the main tasks in this phase.

**Methodology**

First I tried to have an overall idea of the dataset, so once opened the file and saved it into a dataframe, I used the .info() function to understand the type of the columns and their amount of missing values.

Many columns had the majority of their population composed by NaN, so I decided to drop these columns (as written in the previous chapter).

Once the dataset was cleaned by all the missing values (I preferred not to replace them by mean of the column in order to avoid any data manipulation and distortion) using .dropna function, then I tried to understand which columns could have more impact on the target value.

After this stage, I dropped many columns. This allows me to get the final dataframe in terms of columns to use (ADDRTYPE, SEVERITYDESC, COLLISIONTYPE, PERSONCOUNT, PEDCOUNT, PEDCYLCOUNT, VEHCOUNT, UNDERINFL, WEATHER, ROADCOND, LIGHTCOND).

Another thing that has to be considered is whether or not the variable could be available before in order to predict the result. For example, variables such as SEVERITYDESC could not be used since they mean that the incident was already occurred (since you already have a description of the severity). For this reason, I decided to keep only variables that could be processed directly at the beginning (such as weather and light condition at the moment of the beginning of the drive session) or variables that can be guessed considering where you are going. For instance, if you are going to drive through the city, pedestrians could be encountered. Instead, if you are going to drive through campaigns, probably there will be very few pedestrians or even none.

One assumption was made for the column UNDERINFL. By looking at its population with the function .groupby() followed by the function .size(), the result was 4 different elements. There were N, Y, 0, 1. Since the proportions of N-0 and Y-1 were similar, I supposed that they could be considered as typing errors that were made by who populate the dataset. For this reason, using the function .replace I replace all the Ns and Ys with 0 and 1 respectively. And it was not over with this column. In fact, since the previous 0s and 1s were str type, I change them into int. In this way, in the end the population of the column UNDERINFL was composed only by two values, 0 and 1.

Then, in order to have a smaller number of values in the dataframe (otherwise the time to train the algorithms would have been much higher) I reduced the size of it. I decided to drop the 85% of the instances, thus the final dataframe will be of about 22000 rows. Still a good sample!

The next step was to split the dataset into train and test. To make this I used the function train\_test\_split form sklearn. As parameters, I set the test size as 25%, I stratify according to the target variable in order to have both training and test set populated by the same % of categories of the target label.

After obtaining the training and test datasets, I decided to use them into 3 different algorithms. At the end I will select the one that perform better.

The first one is the KNeighborsClassifier. I decided to select a number of neighbors between 11 and 17.  
Then, I used GridSearchCV with cv=3 and scoring method the accuracy. I then fit the model and print the best result.

The second algorithm that I selected is the Logistic Regression. For this, I tried with values of C of 0.1, 1 and 10. Then pretty much the same as the first algorithm.

The third and last algorithm that was used is the Decision Tree, more precisely, the Decision Tree Classifier.

**Results**

At the end, the 3 models gave me the results as the best results the values of 81.8%, 83.1% and 84.6% respectively. For this reason, I decided to select the third one, the Decisional Tree Classifier.

**Discussion**

The main point that I would like to discuss is about the sample reduction that I made. Probably, disposing of a more powerful machine, I will recommend to try with the entire dataset in order to compare the results. Having more data does not always mean better performances, since overfitting could appear. But for sake of completeness, it would be reasonable to give a try to the whole dataset.

**Conclusion**

As a result of this process, after analyzing the data, modifying the dataset in order to get the most out of the data, 3 different algorithms were used. The one that shows slightly higher results is the Decisional Tree Classifier that, with a prediction accuracy of 84.6% on the test set, is selected as the best model.