# The IBM Professional Data Science Capstone Project

## <u>Severity Code on car accidents</u>

By Oriol Jorba Ferro
Coursera student

The **Jupyter notebook** with all the technical details can be found here.

And the presentation here.

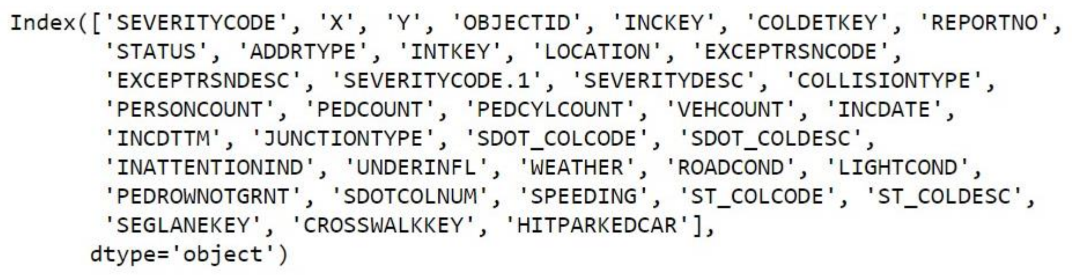
# 1. Introduction

The world suffers due to car accidents, including the USA. National Highway Traffic Safety Administration of the USA suggests that the economical and societal harm from car accidents can cost up to $871 billion in a single year. The project aims to understand, study and predict the severity of accidents and the external factors that can play a role such as the weather, the road and light conditions.

This model can be used for alerting drivers when the probability of having a severe accident is high, due to weather and other external conditions.

# 2. Data understanding and preparation

The data used for this project can be found here: https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv. It consists of all type of collisions on the city of Seattle from 2004 to present. The raw data has 194673 observations (collisions) and the following columns:



```
Index(['SEVERITYCODE', 'X', 'Y', 'OBJECTID', 'INCKEY', 'COLDETKEY', 'REPORTNO',
       'STATUS', 'ADDRTYPE', 'INTKEY', 'LOCATION', 'EXCEPTRSNCODE',
       'EXCEPTRSNDESC', 'SEVERITYCODE.1', 'SEVERITYDESC', 'COLLISIONTYPE',
       'PERSONCOUNT', 'PEDCOUNT', 'PEDCYLCOUNT', 'VEHCOUNT', 'INCDATE',
       'INCDTTM', 'JUNCTIONTYPE', 'SDOT_COLCODE', 'SDOT_COLDESC',
       'INATTENTIONIND', 'UNDERINFL', 'WEATHER', 'ROADCOND', 'LIGHTCOND',
       'PEDROWNOTGRNT', 'SDOTCOLNUM', 'SPEEDING', 'ST_COLCODE', 'ST_COLDESC',
       'SEGLANEKEY', 'CROSSWALKKEY', 'HITPARKEDCAR'],
      dtype='object')
```

## 2.1 Target variable

The target variable is ‘SEVERITYCODE’, being a code that corresponds to the severity of the collision; 1 for property damage and 2 for injuries. We have modified this into a binary variable, for simplicity, being 0 for property damage and 1 for injuries.

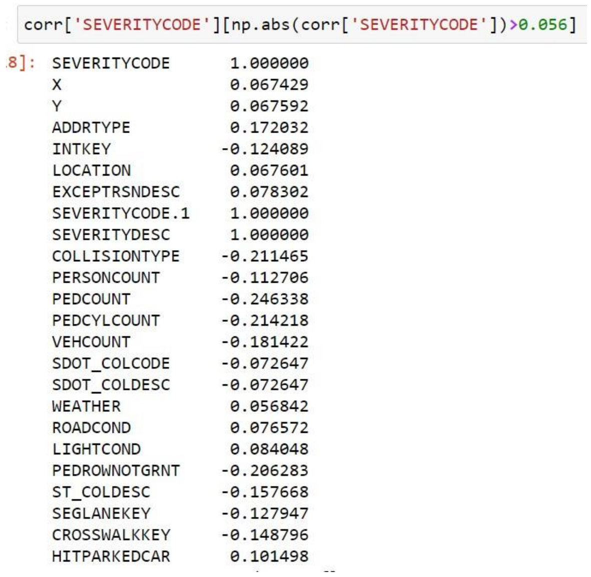
## 2.2 Predictor variables

The predictor variables we have chosen for our model are the following:

I. **Weather:** Categorical variable, describing the weather conditions during the time of the collision.

II. **Road Condition:** Categorical variable, 'ROADCOND' describing the condition of the road during the collision.
III. **Light Condition:** Categorical variable, 'LIGHTCOND', describing the light conditions during the collision.

Initially we thought we could perform our model only using the weather, road and light conditions. Nevertheless, using some feature engineering we have found that their correlation is low, as we can see on figure 1. Therefore, as we can predict that our model will not perform good, we have also included some other variables:



```
corr['SEVERITYCODE'][np.abs(corr['SEVERITYCODE'])>0.056]
```

```
8]:  SEVERITYCODE        1.000000
     X                   0.067429
     Y                   0.067592
     ADDRTYPE            0.172032
     INTKEY             -0.124089
     LOCATION            0.067601
     EXCEPTRSNDESC       0.078302
     SEVERITYCODE.1      1.000000
     SEVERITYDESC        1.000000
     COLLISIONTYPE      -0.211465
     PERSONCOUNT        -0.112706
     PEDCOUNT           -0.246338
     PEDCYLCOUNT        -0.214218
     VEHCOUNT           -0.181422
     SDOT_COLCODE       -0.072647
     SDOT_COLDESC       -0.072647
     WEATHER             0.056842
     ROADCOND            0.076572
     LIGHTCOND           0.084048
     PEDROWNOTGRNT      -0.206283
     ST_COLDESC         -0.157668
     SEGLANEKEY         -0.127947
     CROSSWALKKEY       -0.148796
     HITPARKEDCAR        0.101498
```

*Figure 1: Correlation between some possible predictor variables and the target variable.*

IV. **Speeding:** Binary variable determining if speeding was a factor in the collision.
V. **Under influence:** Binary variable, 'UNDRINFL' determining if the driver was under the influence of drugs or alcohol.
VI. **Inattention:** Binary variable, 'INATTENTIONIND', determining if the driver inattention to the road was a factor in the collision.
VII. **Address type:** Collision address type: Alley, Block or Intersection
VIII. **Collision type:** Different type of collisions: Parked car, angles, head on, etc…
IX. **Pedestrian count:** The number of pedestrians involved in the collision.
X. **Person count:** The total number of people involved in the collision
XI. **Pedcyclcount:** The number of bicycles involved in the collision.
XII. **Vehicle count:** The number of vehicles involved in the collision.
XIII. **PEDROWNNOTGRNT:** Whether the pedestrian right of way was not granted.

# 3. Methodology

## 3.1 Root variables that can generate an accident

Looking at our dataset we can identify some predictor categorical variables, called here 'root variables', that might help to explain why the accident had occurred. These variables are the different factors that can lead to an accident.

- **Weather:**

On figure 2, we can see that there is an important unbalance on the different types of values this categorical variables have. For simplicity, we had modified our weather conditions into several variables:

0. Good weather conditions, overcast or partly cloudy
1. Raining and other bad conditions.

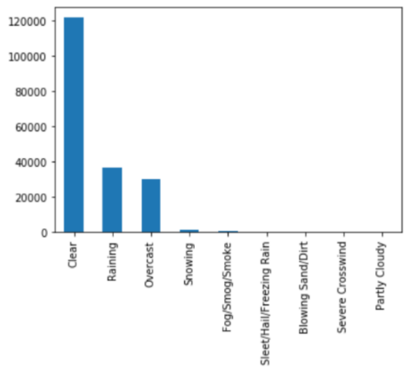


*Figure 2: Different values of the categorical variables weather*

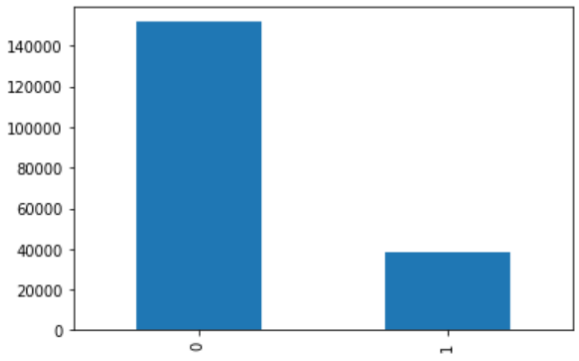


*Figure 3: Values of the new binary variable, Weather. Being 0 good weather, and 1 bad weather*

- **Road and light conditions:**

On a similar way we have done with the weather, we will modify the road and light conditions into binary variables:

a) Road Conditions:
   0. Good road conditions: Dry
   1. Bad road conditions: Wet, Ice, Snow, Slush, etc…
b) Light Conditions:
   0. Good light conditions (Daylight, Dawn, Dusk)
   1. Bad light conditions (Night streetlight on and off)

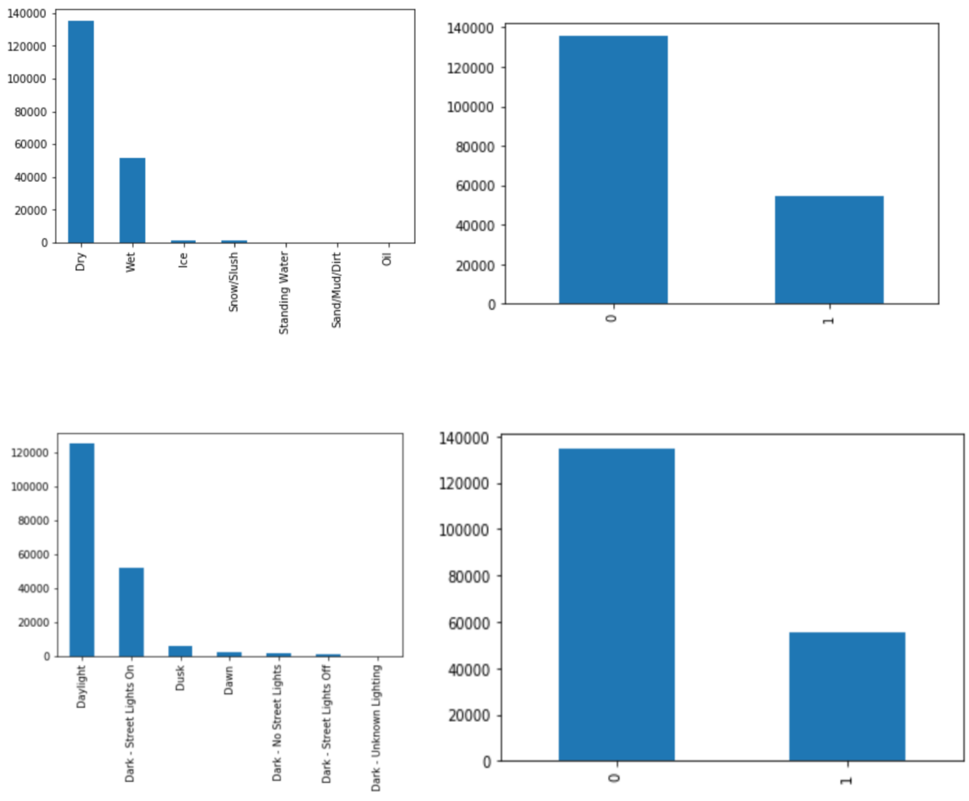


*Figure 4: Different values for road and light conditions, being a categorical variable on the left figures and the new binary variable on the right.*

- **Other root variables:**

The other root variables we consider in our model are all in a binary form (Speeding, Under Influence of drugs/alcohol, inattention to the road and whether the pedestrian right of way was not granted.

All the observations that had all the root values as null values (or other and unknown) had been deleted, as they do not give enough information of the causes of the accident.

- **Overview of root variables:**

Using all these binary variables, we can plot the factors that might led to an accident:

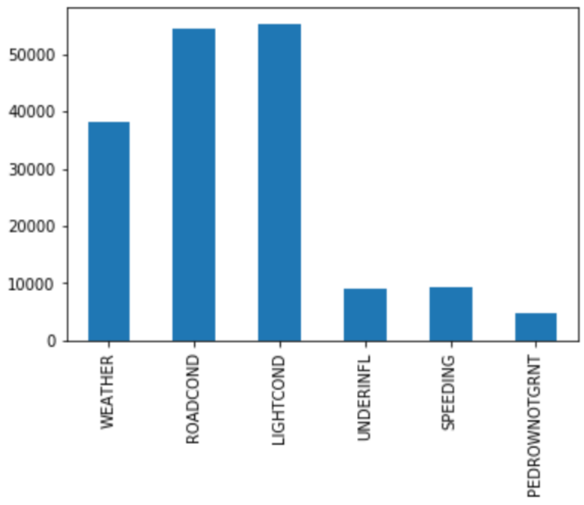


*Figure 2: Causes of car accidents in Seattle*

## 3.2 Other variables that can have some influence on the severity of the accident

The other variables in our model are categorical (Address Type, Collision Type) and the count of persons, vehicles, bicycles and pedestrian on the accidents.  We have seen that these variables do not have information about the causes of the accident but might be important factors in determining which accidents might cause injures (as there exist an important correlation between them).

- **Collision Type:**

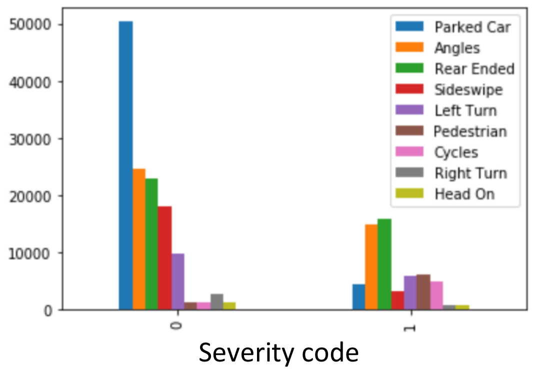


*Figure 5: Frequency of collision type depending on the target value.*

We can see that the collision type can be an important factor determining the types of accidents that can lead into a car accident with injures. Parked car accidents are majorly only property damage accidents, while most pedestrian or cycles accidents are accidents with injuries. Therefore, we split this categorical variable into all the 9 values.

- **Address Type:**

As we can see in the figure below, there are very little alley accidents. Therefore, we decided to convert this variable into a binary variable as well, being:

0. No intersection accidents (block and alley)
1. Intersection accidents

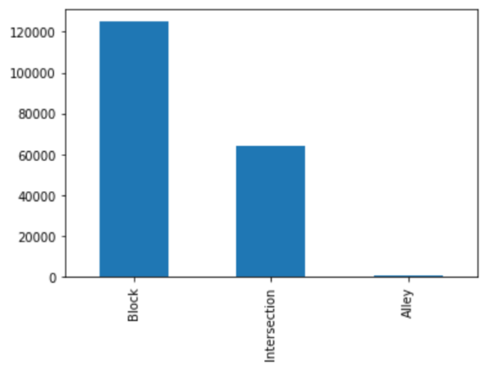


*Figure 6: Different type and frequency of Address Types*

## 3.3 Data unbalance

Our target variable 'SEVERITYCODE' is imbalanced. Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally. In our case we have more than two times more cases with value 0 than with value 1. If we were to fit and train our data, we will probably have an Accuracy Paradox, where our code will probably estimate than having all or almost all our data predict target value equal to 0 for all our data will have a good accuracy score.

For addressing this issue, we have used the imbalanced-learn SMOTE python library, on which we will generate synthetic data into balancing the values.

```
sum(y_unbalanced)/len(y_unbalanced)
```

```
]:  0.30117130076031806
```

```
X_bal, y_bal = SMOTE().fit_resample(X_unbalanced, y_unbalanced)
```

```
sum(y_bal)/len(y_bal)
```

```
]:  0.5
```

## 3.4 Data normalization

We normalize our features matrix using pre-processing sklearn library:

```
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
X_bal = preprocessing.StandardScaler().fit(X_bal).transform(X_bal)
```

# 4. Modelling

The Machine Learning models chosen for this study are:

a) **Decision Tree Analysis**: The Decision Tree Analysis breaks down a data set into smaller subsets while at the same time an associated decision tree is incrementally developed. The result is a tree with decision nodes and leaf nodes.

b) **Logistic Regression**: Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable

c) **k-Nearest Neighbor**: K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (based on distance)

Our data Will be split into a train set and a test set, representing respectively the 70% and 30% of the dataset, using the train_test_split sklearn library:

```
# 30% of test dataset and 70% for train dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_bal, y_bal, test_size=0.25, random_state=3)
```

## 4.1 Decision Tree Analysis

We used the DecisionTreeClassifier sklean library, and we tuned the max depth parameter. We found that for a depth of 10, we had a good accuracy and f1-score. Here we have around 0.7 value for both accuracy and weighted f1-score.

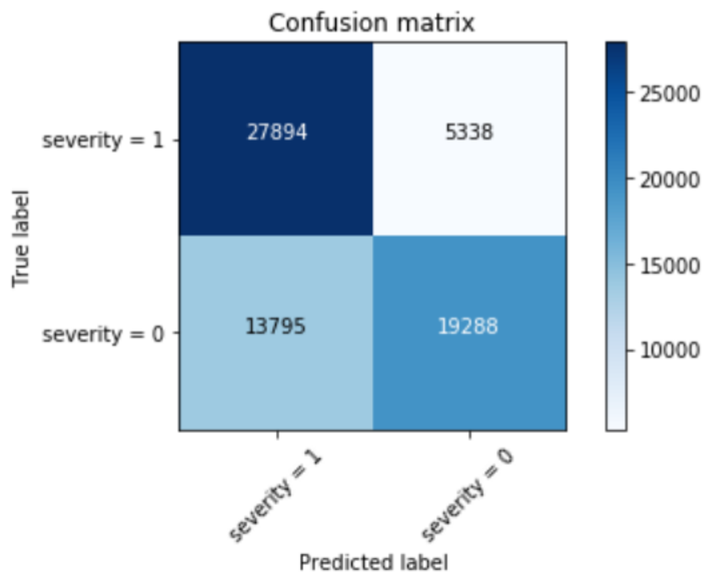
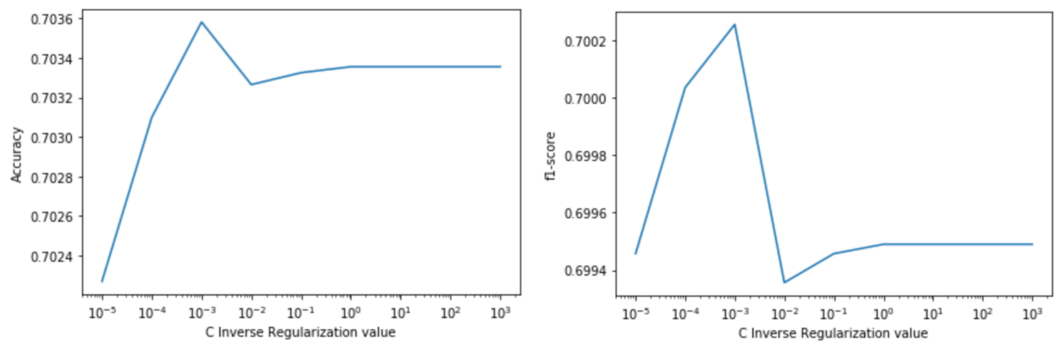


*Figure 7: Confusion matrix from the results of the Decision Tree.*

## 4.2 Logistic Regression

Using the LogisticRegression sklearn library we have computed the accuracy, f1-score and log-loss for several values of C (where C is the inverse of the regularization constant).



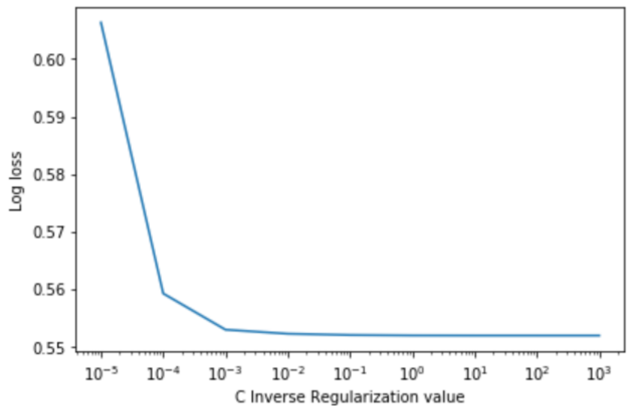


*Figure 8: Values of accuracy, weighted f1-score and log-loss function of inverse of the regularization constant, c*

We can see that with a value of C = 0.001 we have maximum accuracy and f1-score, and it's as well an elbow point on the log-loss plot.

## 4.3 K-Nearest Neighbor

Using the kNeighborsClassifier sklearn library we have computed the accuracy and f1-score for several values of n, being n the number of neighbors.

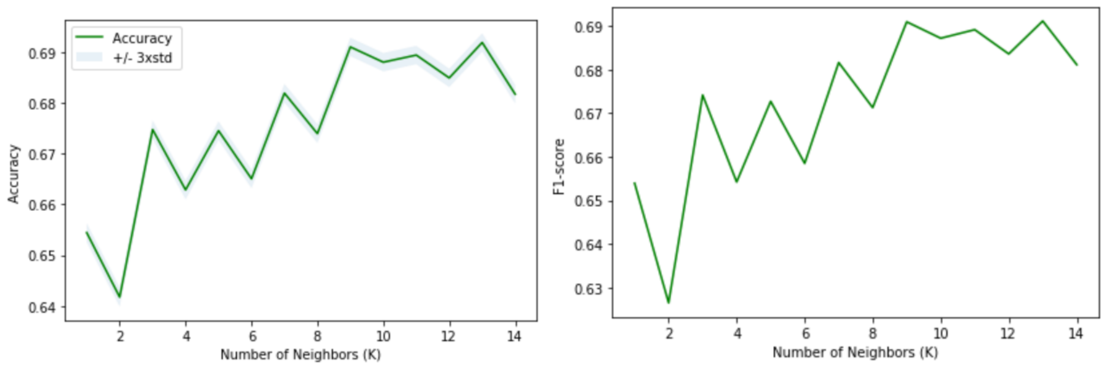


*Figure 9: Accuracy and weighted f1-score computed for different number of neighbours using a KNN algorithms.*

We can see that un of the best values is for k=9, where both values are around a maximum.

# 5. Evaluation and results

Using the from classification_report sklearn library we can access to some basic metrics, to conclude what is the best method.

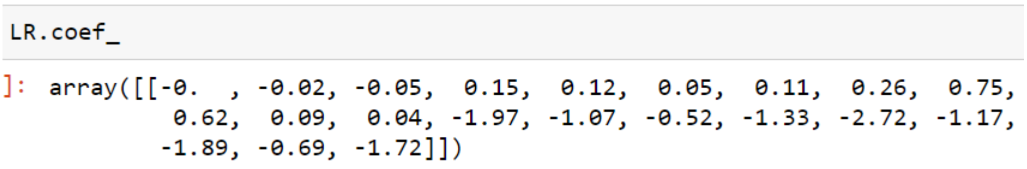
| | Tree Decision Analysis | | | Logistic Regression | | | K-Nearest Neighbors | | |
|---|---|---|---|---|---|---|---|---|---|
| Severity Code | Precision | Recall | F1-score | Precision | Recall | F1-score | Precision | Recall | F1-score |
| 0 | 0.78 | 0.58 | 0.67 | 0.76 | 0.60 | 0.67 | 0.69 | 0.69 | 0.69 |
| 1 | 0.67 | 0.84 | 0.74 | 0.67 | 0.81 | 0.73 | 0.69 | 0.69 | 0.69 |
| Weighted | 0.73 | 0.71 | 0.71 | 0.71 | 0.70 | 0.70 | 0.69 | 0.69 | 0.69 |
| Accuracy | 0.71 | | | 0.70 | | | 0.69 | | |

We can observe that all our 3 methods had around 0.70 accuracy, and approximately the same precision, recall and f1-score, with the Tree Decision Analysis being the best and kNN the worst (and slower), but the more constant in terms of precision-recall.

# 6. Discussion

We think that the correlation between the results and the data is solid, nevertheless, there are some variables that will play a higher role to our model.

Using the Logistic Regression Model, we can find the coefficients used on of our model:



```
LR.coef_
```

```
]: array([[-0.  , -0.02, -0.05,  0.15,  0.12,  0.05,  0.11,  0.26,  0.75,
         0.62,  0.09,  0.04, -1.97, -1.07, -0.52, -1.33, -2.72, -1.17,
        -1.89, -0.69, -1.72]])
```

We can see that the first variables, are very little compared on the last variables, this means that the weight of the first variables (roots variables) are smaller than the other variables in our model.  We can therefore say that **the weather, road and light conditions are not a good estimator for knowing the severity of an accident**. We think that it is more easily to determine this with the other variables, such as type of accident.

After doing this work, we think that a more valuable study will be computing the **frequency of accidents** and compare them with the weather, road and light conditions. On the same line, we could multiply each accident by their severity, and obtaining therefore a value of "dangerousity" depending on weather, road and light conditions.

Moreover, the data imbalance can be approached differently. Using the SMOTE() method we think we are over-weighting the under-representing class (y=1). This can be reflected on the results, as almost all methods had better recall on y=1.

# 7. Conclusion

We have proved that a correlation with the data and the target value, nevertheless, it can be significantly improved. We do not think that weather and road/light conditions are a good estimator in this case and with our data. Probably a different approach can give substantially better results.