# Predicting the Severity of Road Accidents

Prathmesh Tokekar

September 08, 2020

## 1. Introduction

### 1.1 Background

Every year around 1.9 Million people die in road accidents. At times of harsh weather conditions people are more prone to road accidents. In order to minimise accidents it would be a good idea to let people travelling through that route know how severe accidents can happen on that route.

As a data scientist, our job is to analyze the data of collisions that took place in similar weather conditions on that route and forecast the severity of accidents.

### 1.2 Problem

To help people travelling on highways during bad weather days, to know how risky it is to drive now. I need to provide the user with a measure of how severe an accident is he likely to get if he / she travels through the road at a particular instance.

## 2. Data acquisition and cleaning

### 2.1 Data sources

I am provided with the data of the collision from the Seattle Authorities. This is a publicly accessible database.

### 2.2 Data cleaning

There are several columns with missing entries and thus in order to move ahead to the analysis phase, I need to first clean the data.

For this I first check what are the data types of each column. Then I analyzed the number of unique entries present in the column. This gives us an idea of whether I can replace the “NaN” values and if yes, then what should I use.

Next, I compute the number of “NaN” values in each column. If I find that over 70% of the entries in a particular column are “NaN” then I drop it. If the NaN value exists in the column which has data type as integer or float, with more than 15 unique entries; I replace them with the

mean. If the column has data type as string or object and has less than 15 unique values, I replace them with the mode of the column. I also need to check the distribution of the values and check whether median can be a better central tendency for the NaN values or not.

### 2.3 Feature selection

After data cleaning, I move on to the process of feature selection. The first step of this process involves finding out the correlation of columns amongst each other.

For this I compute the correlation matrix and the results are as shown in the image below.

| | SEVERITYCODE | X | Y | OBJECTID | INCKEY | COLDETKEY | INTKEY | SEVERITYCODE.1 |
|---|---|---|---|---|---|---|---|---|
| SEVERITYCODE | 1.000000 | 0.010309 | 0.017737 | 0.020131 | 0.022065 | 0.022079 | 0.006553 | 1.000000 |
| X | 0.010309 | 1.000000 | -0.160262 | 0.009956 | 0.010309 | 0.010300 | 0.120754 | 0.010309 |
| Y | 0.017737 | -0.160262 | 1.000000 | -0.023848 | -0.027396 | -0.027415 | -0.114935 | 0.017737 |
| OBJECTID | 0.020131 | 0.009956 | -0.023848 | 1.000000 | 0.946383 | 0.945837 | 0.046929 | 0.020131 |
| INCKEY | 0.022065 | 0.010309 | -0.027396 | 0.946383 | 1.000000 | 0.999996 | 0.048524 | 0.022065 |
| COLDETKEY | 0.022079 | 0.010300 | -0.027415 | 0.945837 | 0.999996 | 1.000000 | 0.048499 | 0.022079 |
| INTKEY | 0.006553 | 0.120754 | -0.114935 | 0.046929 | 0.048524 | 0.048499 | 1.000000 | 0.006553 |
| SEVERITYCODE.1 | 1.000000 | 0.010309 | 0.017737 | 0.020131 | 0.022065 | 0.022079 | 0.006553 | 1.000000 |
| PERSONCOUNT | 0.130949 | 0.012887 | -0.013850 | -0.062333 | -0.061500 | -0.061403 | 0.001886 | 0.130949 |
| PEDCOUNT | 0.246338 | 0.011304 | 0.010178 | 0.024604 | 0.024918 | 0.024914 | -0.004784 | 0.246338 |
| PEDCYLCOUNT | 0.214218 | -0.001752 | 0.026304 | 0.034432 | 0.031342 | 0.031296 | 0.000531 | 0.214218 |
| VEHCOUNT | -0.054686 | -0.012168 | 0.017058 | -0.094280 | -0.107528 | -0.107598 | -0.012929 | -0.054686 |
| SDOT_COLCODE | 0.188905 | 0.010904 | -0.019694 | -0.037094 | -0.027617 | -0.027461 | 0.007114 | 0.188905 |
| SDOTCOLNUM | 0.004226 | -0.001016 | -0.006958 | 0.969276 | 0.990571 | 0.990571 | 0.032604 | 0.004226 |
| SEGLANEKEY | 0.104276 | -0.001618 | 0.004618 | 0.028076 | 0.019701 | 0.019586 | -0.010510 | 0.104276 |
| CROSSWALKKEY | 0.175093 | 0.013586 | 0.009508 | 0.056046 | 0.048179 | 0.048063 | 0.018420 | 0.175093 |

*Fig1. Correlation matrix*

As can be clearly seen, the OBJECTID , INCKEY , COLDETKEY and SDOTCOLUMN are highly intercorrelated. Hence it is advisable to drop 3 columns from them and just keep 1. OBJECTID was the column that still remained and other columns were removed.

## 3. Exploratory Data Analysis

### 3.1 Target variable

Our goal with this project was to calculate the severity of any accident. This is represented by the column named SEVERITYCODE, It contains 2 values 1 & 2 where 1 means less severe accident and 2 means more severe accident.

### 3.2 Relationship between SEVERITY and the column SEVERITYDESC

The column SEVERITYDESC contained 2 values “Property Damage” and “Injury Collision” . When they are splitted into dummy variables I find that they highly correlate with our Target variable that is SEVERITY . Hence to make the dataset more unbiased I dropped the column of SEVERITYDESC

### 3.3 Relationship between SEVERITY and DATE

I understand that date is useful data for the prediction but it is neither a categorical variable and nor an Integer / Float. If I had to analyze the relationship of Date with our target variable then it would have required TIME SERIES analysis which is out of the scope of the course. Hence I thought to drop this column.

### 3.4 Relationship between SEVERITY and REPORTNO

The column of Report number just contained the data as a random combination of letters & numbers used to denote the ID of the report and hence I chose to drop this column.

## 4. Predictive Modeling

Our aim with this project was to build a predictive model for finding the severity of accidents that occur on roads. Our target variable in this case is a binary object with value 0 or 1. Hence we can use the following algorithms to predict :

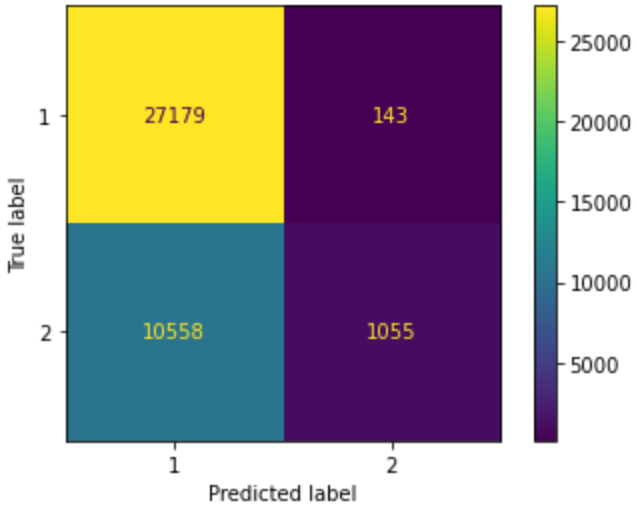
1. Logistic Regression
2. Decision Tree Classification
3. Random Forest Classification
4. XGBoost Classification

### 4.1 Models Used

#### 4.1.1 Logistic Regression

Here are the results of the Logistic regression model when applied to the filtered data.

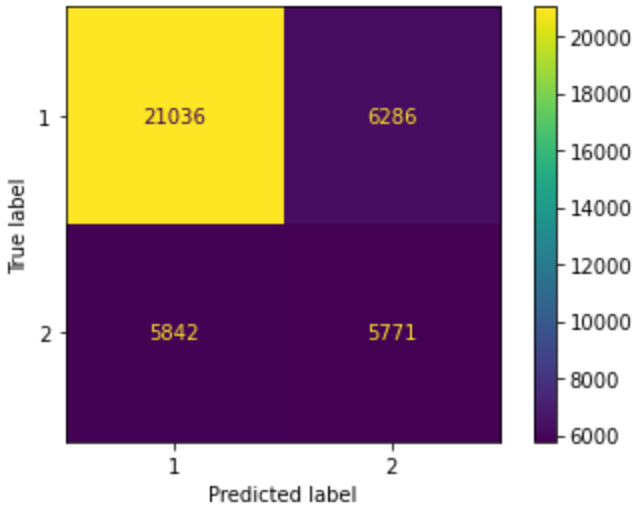
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 0.72 | 0.99 | 0.84 | 27322 |
| 2 | 0.88 | 0.09 | 0.16 | 11613 |
| accuracy | | | 0.73 | 38935 |
| macro avg | 0.80 | 0.54 | 0.50 | 38935 |
| weighted avg | 0.77 | 0.73 | 0.64 | 38935 |




### 4.1.2 Decision Tree Classification

Here are the results of the Decision Tree Classification model when applied to the filtered data.

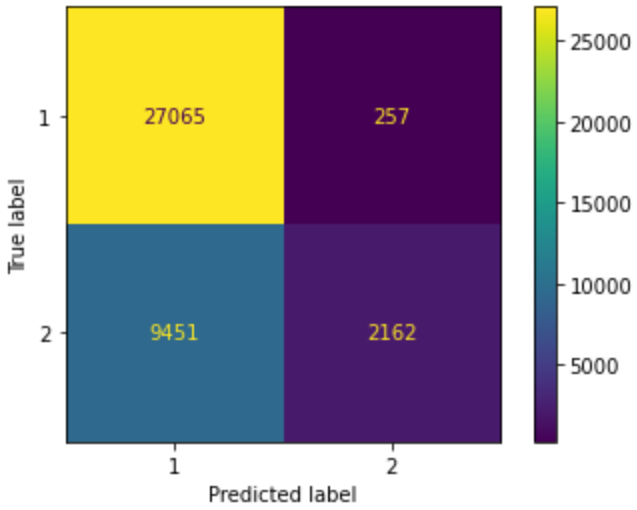
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 0.78 | 0.77 | 0.78 | 27322 |
| 2 | 0.48 | 0.50 | 0.49 | 11613 |
| accuracy | | | 0.69 | 38935 |
| macro avg | 0.63 | 0.63 | 0.63 | 38935 |
| weighted avg | 0.69 | 0.69 | 0.69 | 38935 |




### 4.1.3 Random Forest Classification

Here are the results of the Random Forest Classification model when applied to the filtered data.

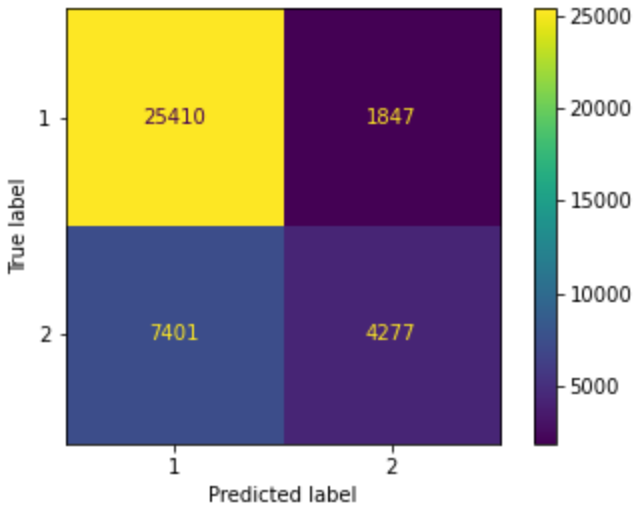
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 0.74 | 0.99 | 0.85 | 27322 |
| 2 | 0.89 | 0.19 | 0.31 | 11613 |
| accuracy | | | 0.75 | 38935 |
| macro avg | 0.82 | 0.59 | 0.58 | 38935 |
| weighted avg | 0.79 | 0.75 | 0.69 | 38935 |



### 4.1.4 XGBoost Classification

Here are the results of the XGBoost Classification model when applied to the filtered data.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 0.77 | 0.93 | 0.85 | 27257 |
| 2 | 0.70 | 0.37 | 0.48 | 11678 |
| accuracy | | | 0.76 | 38935 |
| macro avg | 0.74 | 0.65 | 0.66 | 38935 |
| weighted avg | 0.75 | 0.76 | 0.74 | 38935 |




## 5. Comparing Models and Conclusion

The models used were compared to find out which algorithm gives us the most accurate predictions of the severity of road collisions that may take place. Here it is important to know that each algorithm has drawbacks and shortcomings. Logistic Regression gives the lowest number of False negatives and highest number of True positives but its accuracy is also low. XGBoost has the highest accuracy but on a single field basis it doesn't outperform any other model.

Table 3. Performance of classification models. Best performance labeled in red.

| | Logistic Regression | Decision Tree | Random Forest | XGBoost |
|---|---|---|---|---|
| Accuracy | 0.7251 | 0.6885 | 0.7506 | 0.7624 |
| No. of True Positives | 27132 | 21034 | 27009 | 25410 |
| No. of False Positives | 10577 | 5937 | 9557 | 7410 |
| No. of False Negatives | 125 | 6233 | 248 | 1847 |
| No. of True Negatives | 1101 | 5741 | 2121 | 4277 |