Car Accident Severity Prediction

Ravinder Kumar

September 9, 2020

1. Business Problem:

With increase in population, and the buying capacity, people are using more and more of private means of transport instead of public transport. This results in two huge problems which most of the countries are facing.

a. **Increase in number of accidents.**
b. Pollution

In this project, we will be focused on point "a". Once an accident occurs, the most important goal is to minimize the loss of human lives and to make sure we decrease the probability of it happening again. The most important factor to achieve this is "**if we can predict the severity of an accident**?"

Now, for every problem we can have either **Reactive** approach or **Proactive** approach. Using reactive approach, we can analyze the historical data and make improvements on accident-prone sites. Proactive approach is if you predict the severity of an accident, and then try to prevent the accident from happening.

Important : The model will be useful only if we can predict serious accidents better than the non-serious ones.

In this project, we will go with the Proactive approach to increase road safety. This can help many departments such as healthcare authorities, road transport authorities. Some of the benefits using this model will be :

i. Government can take proactive approach and provide ambulance near the sites where severity is very high. This will help in saving lives and reducing healthcare cost involved.
ii. Road transport authorities can use this model and improve the road conditions, give warnings etc to prevent accidents.
iii. Government can use this model to make better traffic management systems.

2. Data Acquisition and cleaning:

a. Data Sources:

The most important requirement in solving any business/ data science problem is the **Data.** We will be using a labeled dataset provided by Coursera. Our dataset consists of 36 features, a target variable "accident-severity", and 194673 observations. Each accident event can be identified by a unique key. The data is spanned across years 2004 to 2020. Accident severity can have values unknown, property damage(1), injuries(2).
Some of the features in the dataset are :

a. the address, junction type, location
b. Collision types : Different type of collisions such as angled, parked car, right turn etc
c. Features such as count of people, pedestrians, vehicles involved in the collision.
d. Features such as total injuries, serious injuries, fatalities.
e. Date, time of the accident
f. If collision caused was because of not paying attention.
g. Driver was under influence of any alcohol or drug.
h. Conditions such as weather, road, light conditions.

i. If parked car was hit or not.
j. Was the car speeding or not.

b. Data Cleaning:

Major issue after looking at the data was, it had lot of missing values for many features. It is always difficult to replace the missing values for categorical features. For some features, which was just yes/no flag and { "y", null} were stored , it was easy to convert it into {1,0}. We directly mapped "y" to 1 and null "0". These features were mostly Boolean. The target consisted of only 2 value (1,2) which was mapped to (0,1).
There were some duplicate features such as incident data and timestamp, severity code. We dropped the date and kept timestamp for further analysis.
There were some features, where we had actual missing values as it was not captured or was unknown. These records were from 1% to 3% of total records. As our dataset was pretty large and missing data was small, we decided to drop these records.

c. Feature Selection:

After data cleaning, there were 182660 samples and 35 features. Upon examining the names of the features, it was clear that there is redundancy and it can be dropped.
For example, there were many features with codes, and the other one with code description. It is good for analysis or business understanding, but our model will consider both as duplicates. So we dropped all the description features which had corresponding "code" feature like SDOT_COLDESC, ST_COLDESC, SEVERITY_DESC etc.
There are some features which just convey an index like INC_KEY, COLDETKEY, REPORT_NO etc. These were also dropped. We also added a new feature based on the incident date i.e. weekday which is {0, 1, 2, 3, 4, 5, 6}.

dropped features	reason
X','Y','OBJECTID','INCKEY','COLDETKEY','REPORTNO','INTKEY','EXCEPTRSNCODE', 'SDOTCOLNUM','SEGLANEKEY', 'CROSSWALKKEY']	indexes , keys only, doesn't convey any meaning
STATUS','INTKEY','EXCEPTRSNDESC', 'SEVERITYDESC','SDOT_COLDESC','ST_COLDESC'	descriptors, kept the corresponding codes

d. Categorical features encoding :

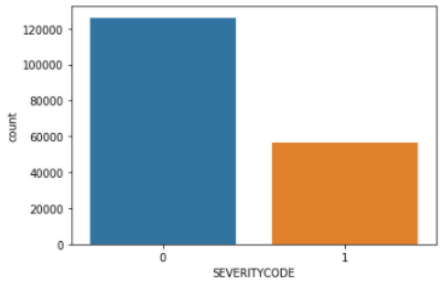
As, almost all our features are categorical, there was some encoding required for some features.

- Features like "INATTENTIONIND", "UNDERINFL", "PEDROWNOTGRNT", "SPEEDING", "HITPARKEDCAR", "ADDRTYPE" contains only two distinct values. So these features were mapped to {0,1}
- The other features doesn't follow any order among itself like "WEATHER", "ROADCOND", "LIGHTCOND" etc, so we chose to use one hot encoding instead of label encoding .

3. Exploratory Data Analysis

a. Calculation of Target Variable :

In our dataset, we had two labels – Property Damage, only collision(1) and Injury Collision (2). To predict severity code, we deleted the description and mapped severity code from 1 to 0 and 2 to 1.



From the above chart, it can clearly be seen that non-serious accidents are almost 2.5 times than the serious ones.

Note : For us predicting serious accident is much important, but we have less data, we will need to use algorithms available for imbalanced data.

b. Correlation Matrix :

After cleaning , deleting unnecessary, duplicate features, we calculated a correlation matrix for the remaining features, to see if we still have redundant features.

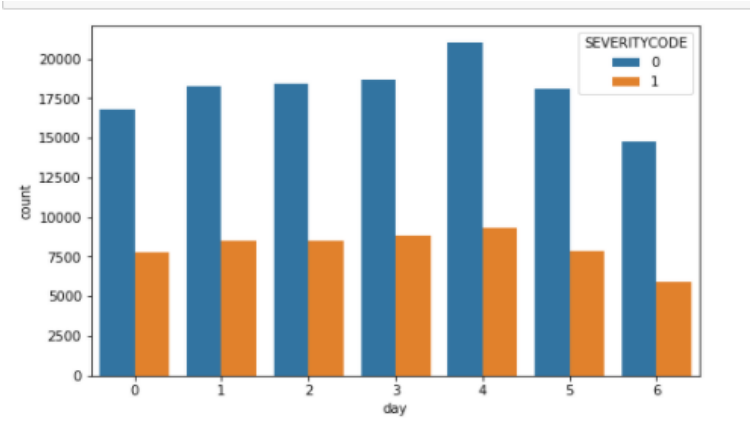
	SEVERITYCODE	ADDRTYPE	LOCATION	COLLISIONTYPE	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT	JUNCTIONTYPE	SDOT_
SEVERITYCODE	1	0.2	-0.04	-0.1	0.1	0.2	0.2	-0.08	-0.2	
ADDRTYPE	0.2	1	-0.3	-0.5	0.06	0.1	0.08	-0.09	-0.9	
LOCATION	-0.04	-0.3	1	0.2	-0.01	-0.04	-0.02	0.01	0.2	
COLLISIONTYPE	-0.1	-0.5	0.2	1	0.02	0.09	-0.2	0.1	0.5	
PERSONCOUNT	0.1	0.06	-0.01	0.02	1	-0.03	-0.04	0.4	-0.07	
PEDCOUNT	0.2	0.1	-0.04	0.09	-0.03	1	-0.02	-0.3	-0.1	
PEDCYLCOUNT	0.2	0.08	-0.02	-0.2	-0.04	-0.02	1	-0.3	-0.09	
VEHCOUNT	-0.08	-0.09	0.01	0.1	0.4	-0.3	-0.3	1	0.09	
JUNCTIONTYPE	-0.2	-0.9	0.2	0.5	-0.07	-0.1	-0.09	0.09	1	
SDOT_COLCODE	0.1	-0.1	0.05	0.06	-0.2	0.3	0.4	-0.5	0.1	
NATTENTIONIND	0.04	-0.08	0.03	0.1	0.07	-0.007	0.001	0.05	0.07	
UNDERINFL	0.04	-0.05	0.007	0.005	0.02	0.01	-0.02	-0.01	0.06	
WEATHER	-0.09	-0.07	0.009	0.02	-0.05	-0.006	-0.05	-0.01	0.09	
ROADCOND	-0.04	-0.02	0.003	-0.005	-0.03	0.009	-0.05	-0.02	0.03	
LIGHTCOND	-0.04	-0.04	0.01	0.03	-0.03	-0.04	0.02	0.03	0.03	
DROWNOTGRNT	0.2	0.2	-0.03	-0.02	-0.03	0.5	0.3	-0.3	-0.2	
SPEEDING	0.04	-0.06	0.03	-0.003	-0.007	-0.04	-0.02	-0.05	0.07	
HITPARKEDCAR	-0.09	-0.1	0.01	0.03	-0.05	-0.03	-0.03	0.07	0.1	
day	-0.02	-0.01	0.0004	-0.02	0.06	-0.02	-0.03	0.004	0.02	

From the above chart, we can see few of the variables are giving a good relation such as ADDRTYPE, COLLISIONTYPE, PERSONCOUNT, PEDCOUNT, PEDCYLCOUNT, JUNCTIONTYPE, SDOT_COLCODE etc.

Let’s drop the column which has very less correlation less than 0.05.

c. Severity Code – Day of incident :

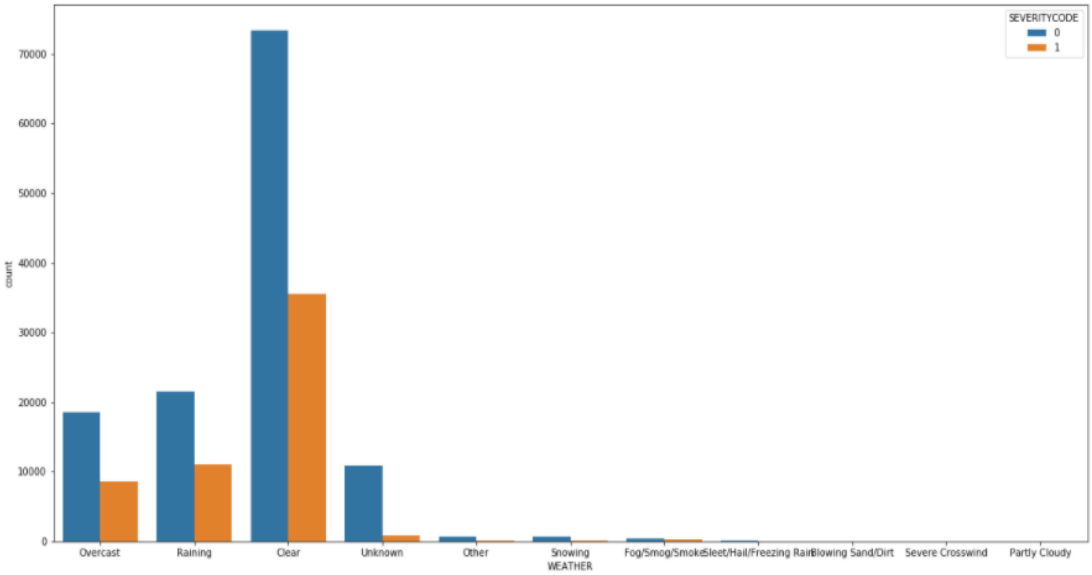
We wrote a function to get timestamp and day of the week from incident date. Days are mapped from 0 to 6 (Mon, Sun, Tue till Sat).



If you see the distribution of accidents with severity code on every date is quite constant. There is no drastic change in events on a particular day. We can remove day column, as there is no information from this feature.

d. Severity Code – Weather Type:

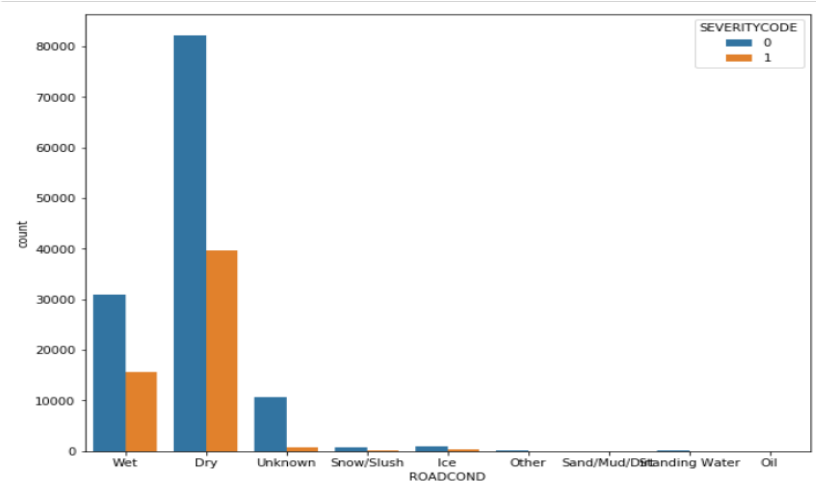
We have total 11 weather types (2 can be combined together as they are unknown and other). Weather value contains Raining, overcast etc.



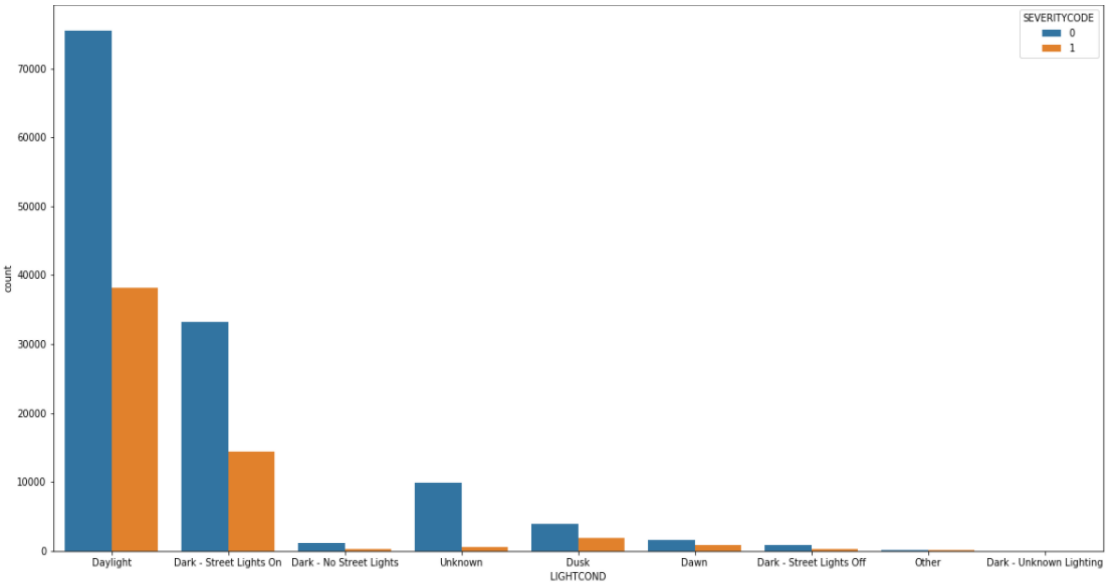
Now from the above graph, it is clearly visible that accidents occur mostly during overcast, raining, clear weather. We will apply one hot encoding on this variable only on 3 weather categories and club others or ignore them, as they behave like noise.

e. Severity Code – Road condition:

From the graph, it is clearly visible that accidents occur mostly on wet and dry road conditions. We can do one hot encoding for this and ignore the others, which are acting as a noise.



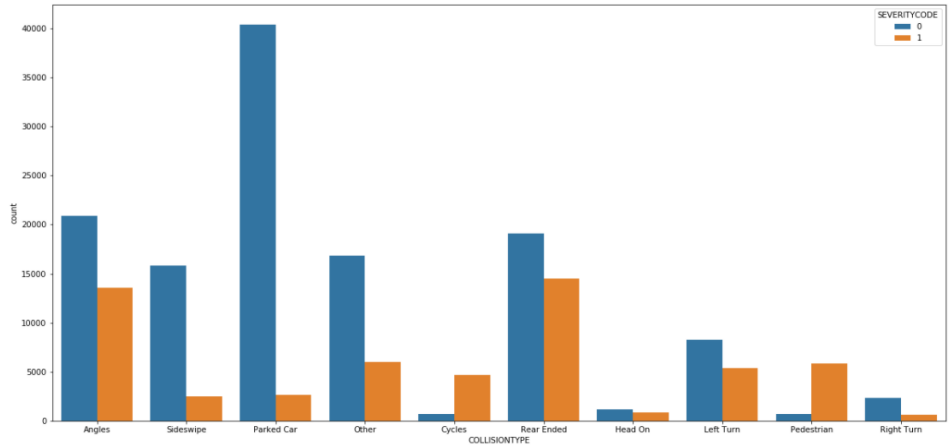
f. Severity Code – Light condition:



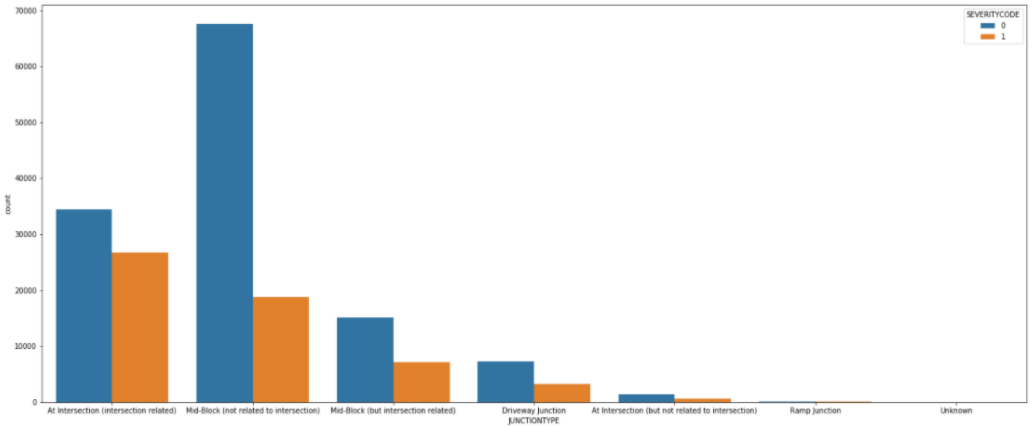
From the above graph, we can conclude that, most observations are captured during daylight, dark street-lights on, dusk. We can ignore other conditions.

g. Severity Code – Collision Type:

From the below graph, we can see there is a strong relation between severity and collision type. Example : if a pedestrian, cycles are involved it will mostly be more severe. If parked car, side swipe is involved, it will mostly will less severe.

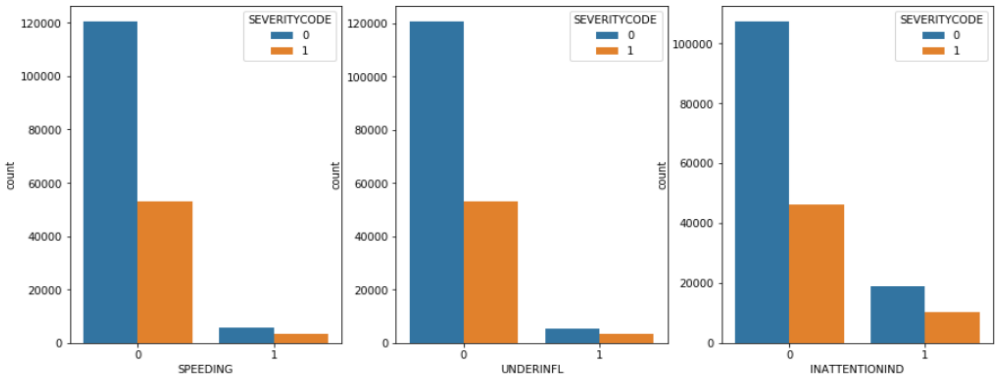


h. Severity Code – Junction Type:



We can remove unknown, ramp junction as they are just acting like noise while doing one hot encoing.

i. Distribution with other less related variables:



4. Predictive Modeling:

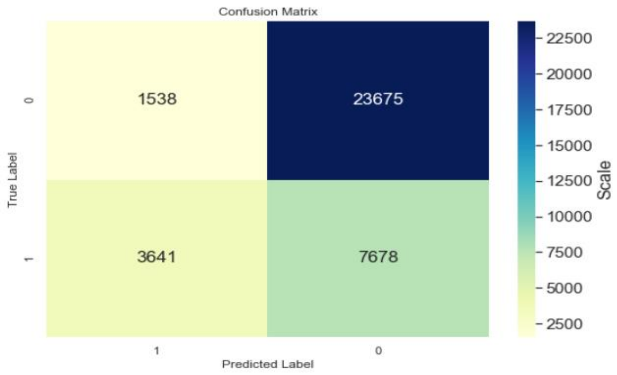
This is a binary classification problem as we have severity code 0 and 1. We used models like decision tree, random forest, knn, logistic regression and finally XGBoost.

Without weights:

Below is the matrix showing accuracy, f1_score, log-loss for classification algorithms:

algorithm	accuracy	f1_score	log loss
decisionTree	75%	45%	
KNN	74%	47%	
Logisitic Regression	75%	44%	49%

Confusion matrix for almost all algorithms without weight looks like this:



It is clearly visible this model can not be used. It is only predicting non-serious accidents, which is not our priority. We want the reverse.

XGBoost without weight, with hyper parameter tuning:

Classification report :

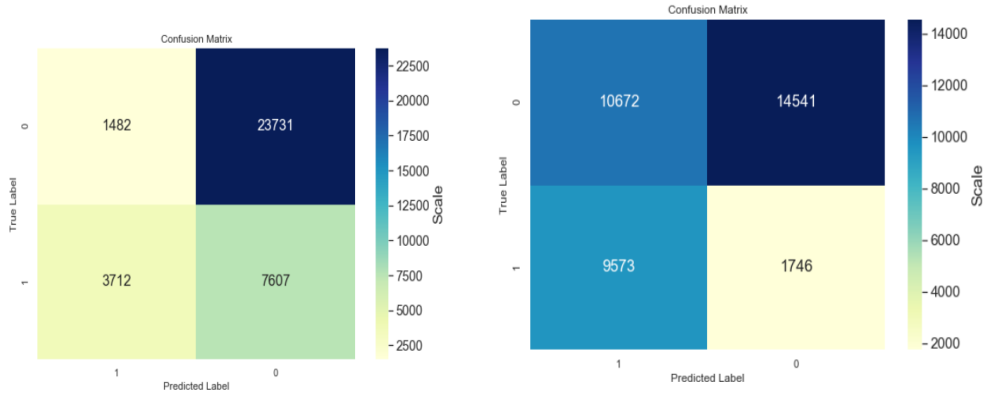
	precision	recall	f1-score	support
0	0.76	0.94	0.84	25213
1	0.71	0.33	0.45	11319
accuracy			0.75	36532
macro avg	0.74	0.63	0.64	36532
weighted avg	0.74	0.75	0.72	36532

XGBoost with weight, with hyper parameter tuning:

	precision	recall	f1-score	support
0	0.89	0.58	0.70	25213
1	0.47	0.85	0.61	11319
accuracy			0.66	36532
macro avg	0.68	0.71	0.65	36532
weighted avg	0.76	0.66	0.67	36532

From the above classification reports, we can see how adding weight to the lesser data (imbalanced) has increased the recall from 33% to 85% and f1_score from 45% to 61%.

Confusion matrix comparison without/with weights:



Without weights **With weights**

Above confusion matrix shows how the number of accurate severe accidents predicted have drastically increased from low range to above medium range in the spectrum.

5. Conclusion:

From the accuracy, f1_score, we got from all the above models, it was observed that we can get decent accuracy (approx. 78%). But the f1_score was very less (approx. 48%). But does accuracy really matters for our problem? The answer is no! In real world scenario, **we want our model to predict more accurately if the accident is more serious**. Also as our data was little imbalanced, it contained less observations for serious accidents, that's why model's accuracy (for serious accidents was very low, but for non-serious accidents, it worked well, but it is not important for us).

As our goal is to get maximum accuracy for predicting serious accidents, we chose XGBoost but with the weight option. Ration of non-serious/serious accidents is almost 2.5:1. So our weight parameter will be roughly around that. We used **GridSearch from sklearn to tune hyperparameters for XGboost.**

XGboost with weight parameter and hyper parameters tuning worked far better than the models without weights. Our model's f1_score increased from 48% to 64%. Although overall accuracy reduced from 75% to 68%. **But accuracy of model predicting serious accidents was approx. 84%, which is perfect for our real world scenario.** We could get better predictions for serious accidents but then we have to have a trade off with non-serious accidents also.

6. Future work:

We have seen that with current approach of adding weight and some feature selection, we can get 80-85 % accuracy for prediction severe accidents. While doing our research, we found out that, there is still lot more we can achieve. We need to collect more data on severe accidents. Also, **Location** is a very important feature which can help a lot in increasing accuracy. We skipped Location, as it had lot of possible values. But there is a research paper, in which we can do one hot encoding on such variables (picking up the most frequent). All real life classification problems are mostly imbalanced, but if we can collect more data, we can also use sampling techniques. It was also observed that, current dataset has most of the value for some important features like road conditions, weather, etc contained "unknown". If we make our data collection process better, it can add great value to this model. There is another area which we can explore, Logisitic regression with weights, it can give additional probabilities also which can further be used in real life scenarios.