# Accident Severity Prediction

## 1. Introduction

Each year millions of people die in traffic accidents, not to mention an additional number of people that are injured or disabled. If the locations of traffic accidents could be predicted, this could have a huge beneficial impact in potentially helping to reduce their number each year. It could also be useful in an insurance context, in order to predict risk, as well as for governments and local road authorities looking to create more efficient systems of road maintenance and improvements. The aim of this project is to predict the severity of road accidents in Seattle, US to help drivers and traffic police department.

Road accident is most unwanted thing to happen to a road user, though they happen quite often. The most unfortunate thing is that we don't learn from our mistakes on road. Most of the road users are quite well aware of the general rules and safety measures while using roads but it is only the laxity on part of road users, which cause accidents and crashes. Main cause of accidents and crashes are due to human errors.

Road accidents are extremely common. Often times they lead to a loss of property and even life. With this analysis, I am attempting to understand these factors and their relation with accident severity. This analysis has multiple applications like an app that will prompt the drivers to be more careful depending on the weather and road conditions on any given day or a way for the police to enforce more safety protocols. In this instance, I am using the data from the City of Seattle’s police department which has been downloaded as a file with name accident_data.csv.

**Different factors of Roads contribute in Accidents:**

Drivers: Over-speeding, rash driving, violation of rules, failure to understand signs, fatigue, alcohol.
Pedestrian: Carelessness, illiteracy, crossing at wrong places moving on carriageway, Jaywalkers.
Passengers: Projecting their body outside vehicle, by talking to drivers, alighting and boarding vehicle from wrong side travelling on footboards, catching a running bus etc. Vehicles: Failure of brakes or steering, tyre burst, insufficient headlights, overloading.

Road Conditions: Potholes, damaged road, eroded road merging of rural roads with highways.
Weather conditions: Fog, snow, heavy rainfall, wind storms, hail storms.

**Preventive measures for accidents:**

- Education and awareness about road safety
- Strict Enforcement of Law
- Engineering:
    - Vehicle design
    - Road infrastructure

**Direct Consequences of Accidents:**

- Fatality (Death)
- Injury
- Property Damage

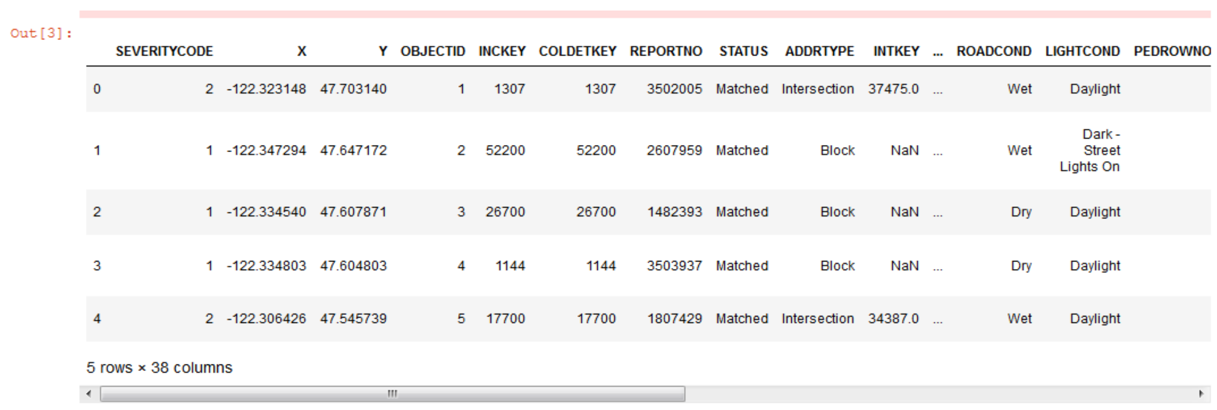
## 2. Data Processing

### Data

This is an extensive data set from the Seattle Police Department with almost 35 parameters like collision type, Person Count, Vehicle count, weather, Speed etc. To accurately build a model to prevent future accidents and/or reduce their severity, I have decided to use few parameters and drop the remaining parameters depend on the data

I used Jupyter Notebook to conduct that analysis and imported all the necessary Python libraries like Pandas, Numpy, Matplotlib and sklearn.

I started by downloading the csv file as given in week 1 detail and saved as accident_data.csv.

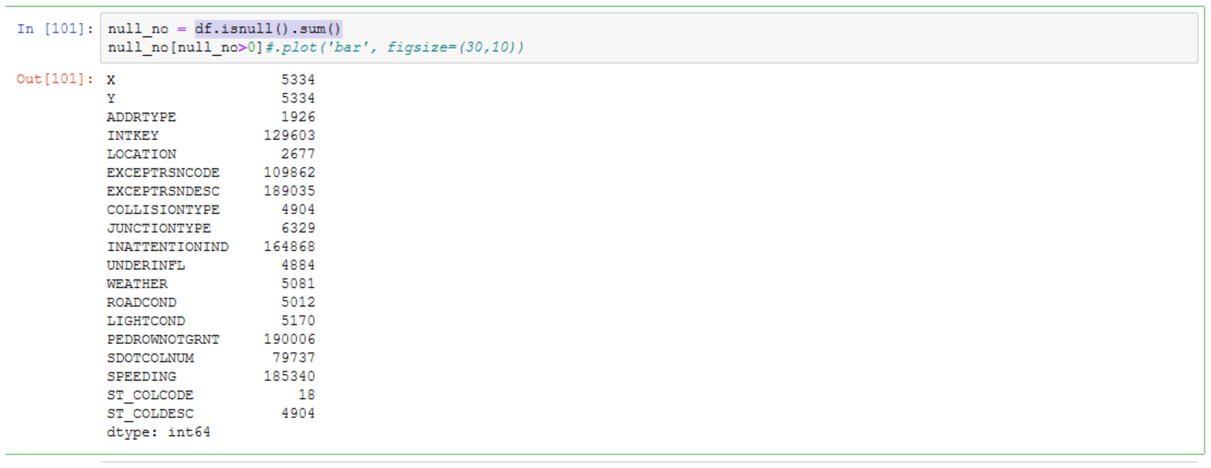

Out[3]:

| | SEVERITYCODE | X | Y | OBJECTID | INCKEY | COLDETKEY | REPORTNO | STATUS | ADDRTYPE | INTKEY | ... | ROADCOND | LIGHTCOND | PEDROWNO |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2 | -122.323148 | 47.703140 | 1 | 1307 | 1307 | 3502005 | Matched | Intersection | 37475.0 | ... | Wet | Daylight | |
| 1 | 1 | -122.347294 | 47.647172 | 2 | 52200 | 52200 | 2607959 | Matched | Block | NaN | ... | Wet | Dark - Street Lights On | |
| 2 | 1 | -122.334540 | 47.607871 | 3 | 26700 | 26700 | 1482393 | Matched | Block | NaN | ... | Dry | Daylight | |
| 3 | 1 | -122.334803 | 47.604803 | 4 | 1144 | 1144 | 3503937 | Matched | Block | NaN | ... | Dry | Daylight | |
| 4 | 2 | -122.306426 | 47.545739 | 5 | 17700 | 17700 | 1807429 | Matched | Intersection | 34387.0 | ... | Wet | Daylight | |

5 rows × 38 columns

The database considered around 35 parameters and the output label was severity code based on these parameters

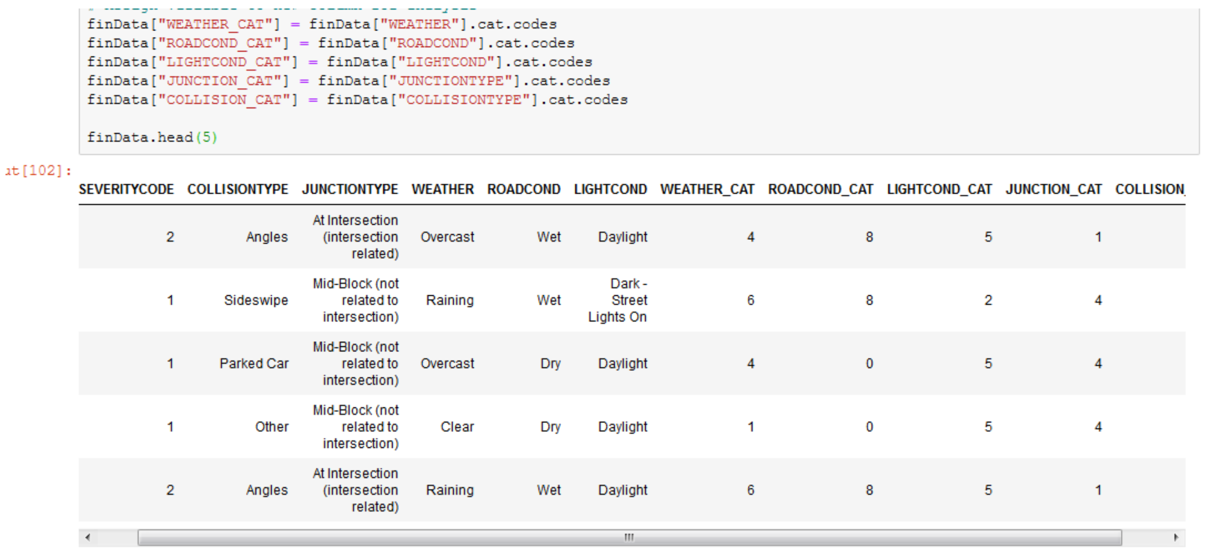
To prepare the data, so that I can  drop columns we do not need from the dataset, i.e., columns that do not have values or where the values are unknown I used the commands df.info( ) , df.describe( )and df.isnull( ).sum( ) .



```
In [101]: null_no = df.isnull().sum()
          null_no[null_no>0]#.plot('bar', figsize=(30,10))

Out[101]: X                   5334
          Y                   5334
          ADDRTYPE            1926
          INTKEY            129603
          LOCATION            2677
          EXCEPTRSNCODE     109862
          EXCEPTRSNDESC     189035
          COLLISIONTYPE       4904
          JUNCTIONTYPE        6329
          INATTENTIONIND    164868
          UNDERINFL           4884
          WEATHER             5081
          ROADCOND            5012
          LIGHTCOND           5170
          PEDROWNOTGRNT     190006
          SDOTCOLNUM         79737
          SPEEDING          185340
          ST_COLCODE            18
          ST_COLDESC          4904
          dtype: int64
```

By looking at the table, I decided to incorporate 5 parameters weather, road condition, light condition, Junction type and collision category. Even though speeding is an important parameter, we have to drop speeding entirely because it is missing over 180,000 values and this can hamper the results.

From the main data frame df, I dropped the unwanted parameters and kept only the necessary parameters and changed their data type to int64

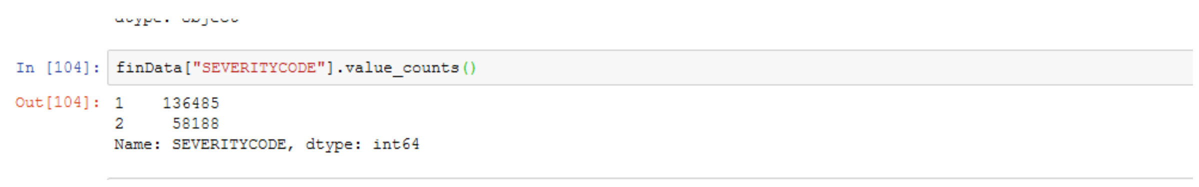


```
finData["WEATHER_CAT"] = finData["WEATHER"].cat.codes
finData["ROADCOND_CAT"] = finData["ROADCOND"].cat.codes
finData["LIGHTCOND_CAT"] = finData["LIGHTCOND"].cat.codes
finData["JUNCTION_CAT"] = finData["JUNCTIONTYPE"].cat.codes
finData["COLLISION_CAT"] = finData["COLLISIONTYPE"].cat.codes

finData.head(5)
```

ıt[102]:

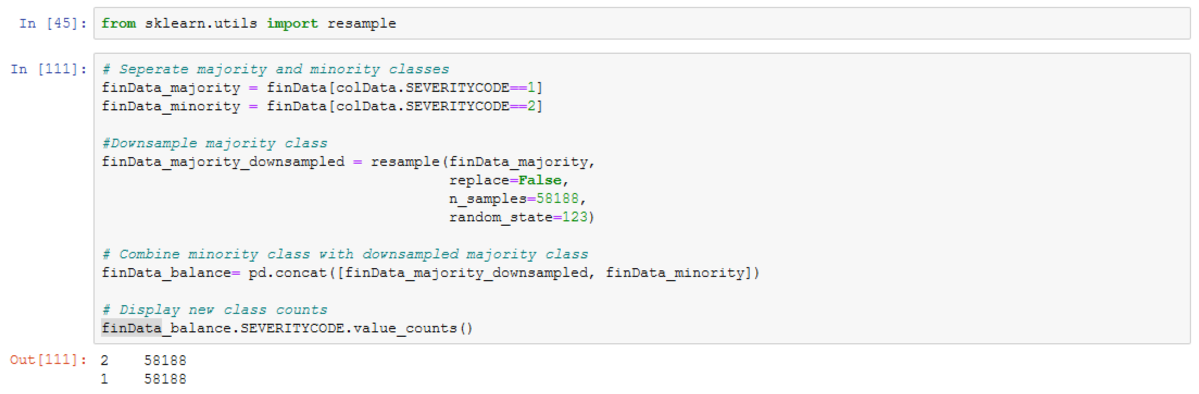
| SEVERITYCODE | COLLISIONTYPE | JUNCTIONTYPE | WEATHER | ROADCOND | LIGHTCOND | WEATHER_CAT | ROADCOND_CAT | LIGHTCOND_CAT | JUNCTION_CAT | COLLISION |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | Angles | At Intersection (intersection related) | Overcast | Wet | Daylight | 4 | 8 | 5 | 1 | |
| 1 | Sideswipe | Mid-Block (not related to intersection) | Raining | Wet | Dark - Street Lights On | 6 | 8 | 2 | 4 | |
| 1 | Parked Car | Mid-Block (not related to intersection) | Overcast | Dry | Daylight | 4 | 0 | 5 | 4 | |
| 1 | Other | Mid-Block (not related to intersection) | Clear | Dry | Daylight | 1 | 0 | 5 | 4 | |
| 2 | Angles | At Intersection (intersection related) | Raining | Wet | Daylight | 6 | 8 | 5 | 1 | |

When we analysed the severity code we found out that data is unbalanced as Severity code 1 was approximately three times larger than Severity code 2.



```
In [104]: finData["SEVERITYCODE"].value_counts()

Out[104]: 1    136485
          2     58188
          Name: SEVERITYCODE, dtype: int64
```

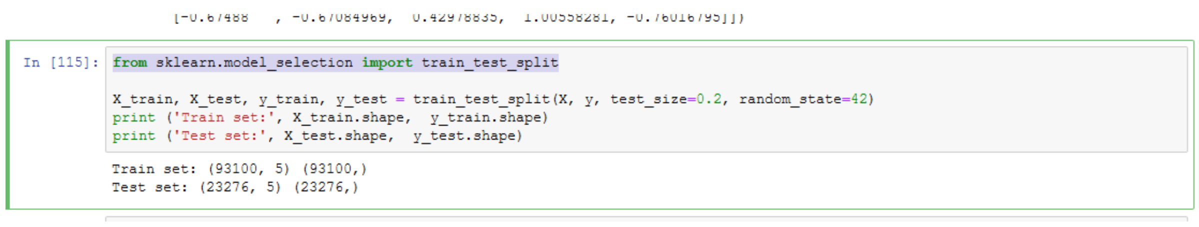
So we have done down sampling for severity code 1 class with sklearn's resample tool. We down sampled to match the severity code 2 exactly with 58188 values each.



```
In [45]: from sklearn.utils import resample

In [111]: # Seperate majority and minority classes
          finData_majority = finData[colData.SEVERITYCODE==1]
          finData_minority = finData[colData.SEVERITYCODE==2]

          #Downsample majority class
          finData_majority_downsampled = resample(finData_majority,
                                                   replace=False,
                                                   n_samples=58188,
                                                   random_state=123)

          # Combine minority class with downsampled majority class
          finData_balance= pd.concat([finData_majority_downsampled, finData_minority])

          # Display new class counts
          finData_balance.SEVERITYCODE.value_counts()

Out[111]: 2    58188
          1    58188
```

With this I have completed data cleaning and balancing work.

Now I have to split the data into training data and testing data with a ratio of 80:20 which I have completed using the command from sklearn.model_selection import train_test_split



```
 [-0.67488   , -0.67084969,  0.42978835,  1.00558281, -0.76016795]])
```

```
In [115]: from sklearn.model_selection import train_test_split

          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
          print ('Train set:', X_train.shape,  y_train.shape)
          print ('Test set:', X_test.shape,  y_test.shape)

          Train set: (93100, 5) (93100,)
          Test set: (23276, 5) (23276,)
```

So the train data set consists of 93100 samples with 5 parameters and 93100 output labels and the test data consists of 23276 samples with 5 parameters and 23276 output labels.

Since we now we are ready with all necessary data, our next step is to build the models and analyse the performance of the system.

## 3. Methodology

This project uses a combination of accident data from the Seattle to predict the accident severity as severe and light. Two algorithms are considered for implementation.

### KNN classifier

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique we generally look at 3 important aspects:

- Ease to interpret output
- Calculation time
- Predictive Power

We can implement a KNN model by following the below steps:

- Load the data
- Initialise the value of  k
- We can use the KNN library for implementation

### Logistic Regression

In statistics, the logistic model is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one. Logistic regression is used in various fields, including machine learning, most medical fields, and social sciences

**Performance Analysis**

For comparison, we have considered two scores Jaccard similarity score and F1score

**Jaccard similarity score**

The Jaccard similarity index (sometimes called the Jaccard similarity *coefficient*) compares members for two sets to see which members are shared and which are distinct. It's a measure of similarity for the two sets of data, with a range from 0% to 100%. The higher the percentage, the more similar the two populations. Although it's easy to interpret, it is extremely sensitive to small samples sizes and may give erroneous results, especially with very small samples or data sets with missing observations.

**F1 score**

In statistical analysis of binary classification, the $F_1$ score (also F-score or F-measure) is a measure of a test's accuracy. It is calculated from the precision and recall of the test, where the precision is the number of correctly identified positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of correctly identified positive results divided by the number of all samples that should have been identified as positive. The F1 Score is the 2*((precision*recall)/ (precision+recall)). F1 score conveys the balance between the precision and the recall.

**4. Result Analysis**

For various values of K, We have calculated the Jaccard similarity score and F1 Score as shown in table 1. We were able to get best F1 score when K =25 for KNN classifier while considering three conditions weather, Road and light.

| K Value | Jaccard similarity score | F1 Score |
|---|---|---|
| 5 | 0.5451 | 0.5247 |
| 10 | 0.54704 | 0.5309 |
| 15 | 0.5500 | 0.5296 |
| 20 | 0.55108 | 0.5315 |
| 25 | 0.55198 | 0.5337 |
| 30 | 0.5319 | 0.52182 |

After finalising the K value as 25, we have repeated the process with one more condition junction type so that total four conditions and again with another condition collision type. The results are listed as given below

| Parameters | KNN classifier | | Logistic Regression | |
|---|---|---|---|---|
| | Jaccard similarity score | F1 Score | Jaccard similarity score | F1 Score |
| 3 | 0.5451 | 0.5247 | 0.5218 | 0.5079 |
| 4 | 0.6117 | 0.6109 | 0.5947 | 0.5946 |
| 5 | 0.69419 | 0.69346 | 0.5849 | 0.5844 |

## 5. Discussion

At the start of our analysis, I was trying to figure out the severity of road accidents based on weather conditions, road conditions, and other factors. Even though our data was a good size, there were a number of missing elements and we needed to clean the data in order to get a good result. We had to drop many conditions  because there were too many missing From the analysis, it is clear that most accidents involve wet roads, bad weather, Poor light and are minor in nature.. When weather conditions are bad at the junction intersection point, this model can alert drivers to remind them to be more careful. Also could be helpful to the travel police department in deciding where to install more traffic signs, or maybe adding CCTV systems.

## 6. Conclusion and Future Work

This analysis has given us some good insight. It seems like a lot of these accidents are minor and avoidable. These findings can be helpful to the Seattle PD in enforcing some new measures to prevent future accidents. To improve the accuracy the following future work can be considered

- Data size can be increased.
- Latest Data can be considered
- Multiple models like decision tree could be trained and then compared.
- More conditions can be included to train the model

.