Traffic Accident Severity Prediction

Team Name: The Collective

Team Members and Email id's:

1. Susmitha Srirangam - 00764501 - ssrir5@unh.newhaven.edu
2. Kaivalya Reddy Maddireddy - 00760616 - kmadd3@unh.newhaven.edu
3. Thatikonda Akanksha - 00764338 - athat2@unh.newhaven.edu

Research Question: Our aim is to predict the severity of accident based on the data collected by applying various data mining techniques.

Dataset information:

This dataset contains 11 attributes with values recorded based on time while considering hours of travel like rush hour, workdays, weekends, surface conditions, weather conditions, speed limits and predicting the severity of accidents weather they are fatal or non-fatal.

Attribute Information:

- **Rush Hour:** Shows weather the hour of travel is comes under rush hour or non-rush hour so, we can get to know the traffic conditions of the area.
- **Work Zone:** Shows if the area of travel is coming under work zone or not from this, we can decide how the timings impact traffic in this zone
- **Workday:** Shows if the day is workday or not because of which we can determine the which zones can see traffic during the working days.
- **INT_HWY**
- **LGTCON_day**
- **Level:** The level of road from the surface which can also tend to accidents when having the impact of weather
- **Speed Limit:** Travel speed of vehicles during the traffic and while having the impact of other conditions mostly lead to accidents
- **Surface Conditions:** Shows weather the surface condition of the road is good or wearied off including the climate conditions like snow, rainy or greasy roads

- **Traffic Two Way:** This shows if the traffic is one way or two ways to rule out and analyze the conditions opposite side
- **Maximum Severity:** With analysis of all these conditions the severity of the accident is determined weather it is fatal or non-fatal.

List of data mining techniques used:

Until today the data mining techniques used are:

- Decision Trees: It is predictive machine learning model that is used for classification of data based on considered attributes
- Logistic regression: This is a statistical model used for classification and predictive analytics, it estimates the probability of event occurring based on a given dataset of independent variables.
- Naïve Bayes Algorithm: This provides us with probability of a prediction from the underlying evidence, as observed in the data.
- K Nearest Neighbor Algorithm: This is a classifier algorithm where the learning is based on how similar a data from other available attributes is.

Description of parameters and hyperparameters:

Decision Trees: Hyper parameter are responsible for controlling the model. In decision tree the hyper parameters are most basic ones are:

- **Criterion:** This function is used to calculate the uncertainty on the rule selected
- **Max Features:** The number of features to consider when searching for the best split rule.
- **Max Depth:** To determine the maximum depth of the tree to get best outcome possible.
- **Cpp_alpha:** Here the node with high complexity is pruned and less than cpp value will be pruned.

Logistic Regression:

Logistic regression is a classification algorithm that is used to predict a outcome where there are only two possible scenarios that is either the event happens or it does not happen. To calculate the outcome we can equally split the data into intervals. We can continue to split the data until we find appropriate outcomes. These intervals are termed as iterations. In our model it took 4 iterations to get to an optimized solution.

Naive Bayes:

Naive Bayes is a probabilistic algorithm which can provide us prediction real quick, In general Naive Bayes doesn't have any hyper parameters to tune, however we can control the data that is used by the algorithm by setting parameters like Maximum_Input_Attributes, Maximum_Output_Attributes and Minimum_States.

K Nearest Neighbor :

The KNN technique, which is used to address classification model problems, establishes an illogical border to classify the data. The program will try to forecast the closest border line as additional data points are received.

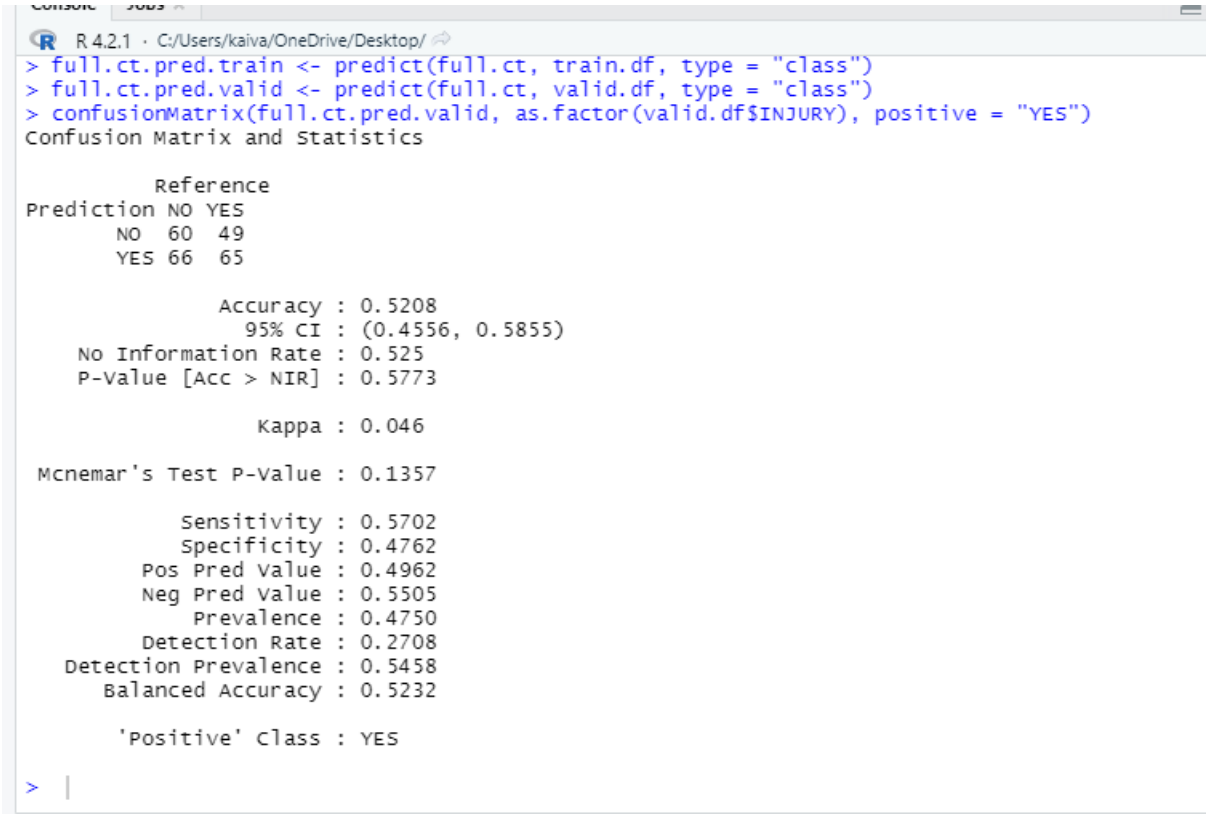
The adjustable parameters for KNN are:

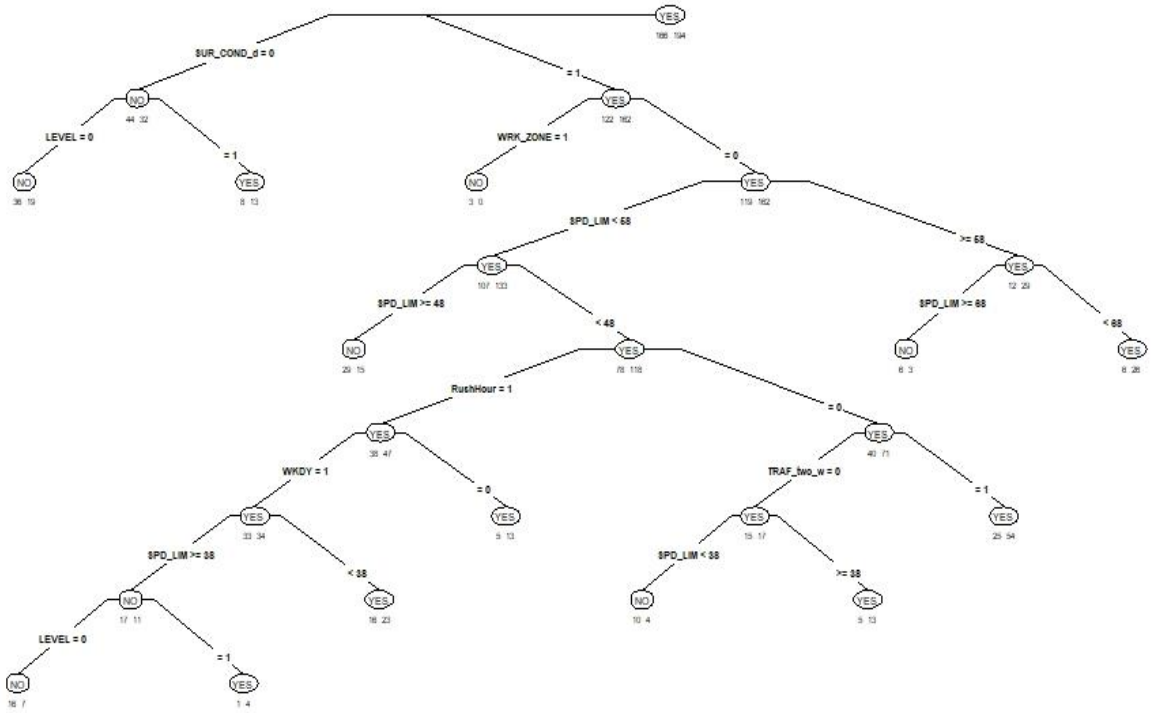
- N_neighbor: This regulates number of neighbors that are checked when an item is being classified. The default value is 5. This is also known as 'K' Value.
- Weights: This determines how weights are distributed among the neighbor values. The default value is uniform.

List of optimization techniques:

By comparing the values of accuracy, sensitivity, and specificity of all the techniques used. By reimplementing and changing the hyperparameter values we tried optimizing the values.

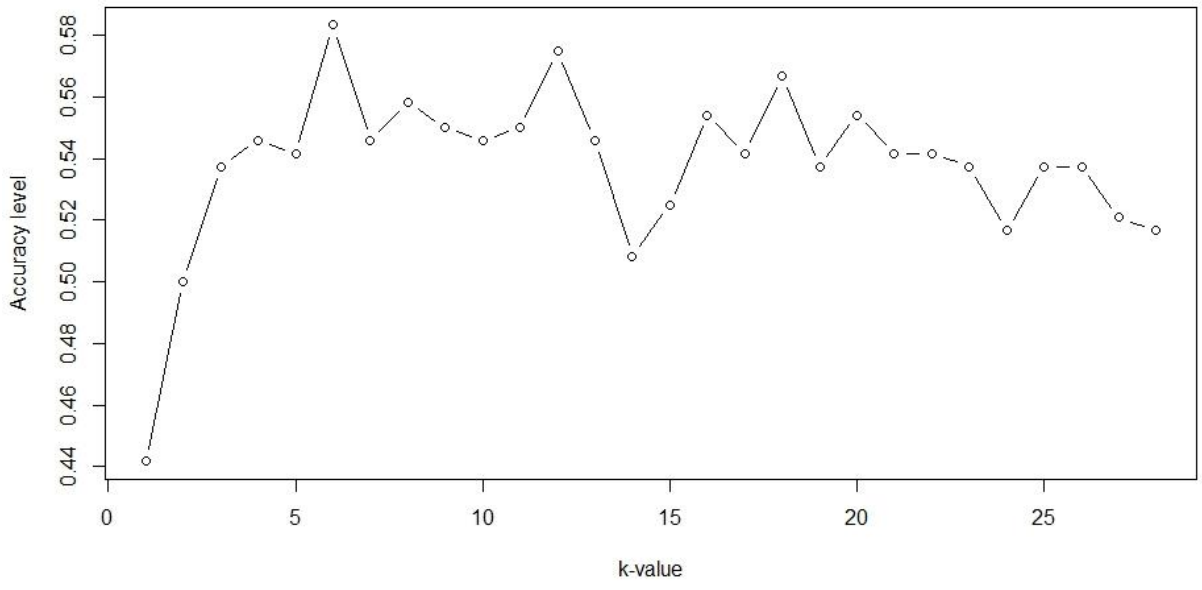
Firstly, for decision tree, we can try to optimize it by changing the depth values so we have added the cp value which gives impact on accuracy and specificity values, this when compared to the previous execution without mentioning the cp value.





Similarly, we can calculate accuracy values in the KNN Algorithm for different values of K. When implemented we have noticed the accuracy value between K value 5 and K value 10 is high so when implemented By changing the K values, observing the changes in accuracy values. We observed that the accuracy value at K=6 is highest.

```
> k.optm =1
> for (i in 1:28){
+   knn.mod <- knn(train = train.df, test = valid.df, cl = outcome[train.rows,], k= i)
+   kl <- confusionMatrix(knn.mod, outcome[valid.rows,], positive = "YES")
+   k=i
+   cat(k, '=', kl$overall["Accuracy"], '\n')
+   k.optm[i]<- kl$overall["Accuracy"]
+ }
1 = 0.4416667
2 = 0.5
3 = 0.5375
4 = 0.5458333
5 = 0.5416667
6 = 0.5833333
7 = 0.5458333
8 = 0.5583333
9 = 0.55
10 = 0.5458333
11 = 0.55
12 = 0.575
13 = 0.5458333
14 = 0.5083333
15 = 0.525
16 = 0.5541667
17 = 0.5416667
18 = 0.5666667
19 = 0.5375
20 = 0.5541667
21 = 0.5416667
22 = 0.5416667
23 = 0.5375
24 = 0.5166667
25 = 0.5375
26 = 0.5375
27 = 0.5208333
28 = 0.5166667
> plot(k.optm, type = "b", xlab= "k-value", ylab= "Accuracy level")
>
```

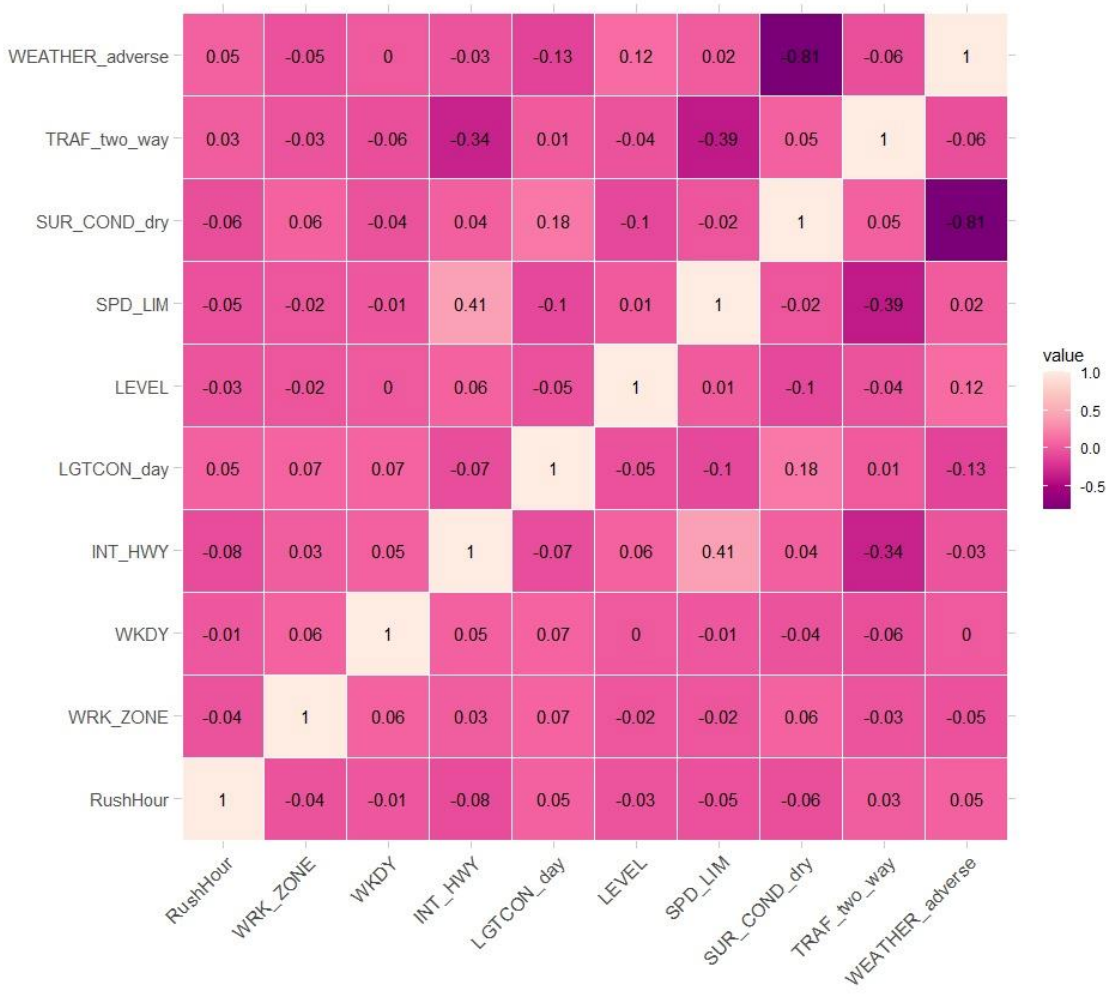


So the optimization values after the K value is changed to 6 is:

```
R 4.2.1 · C:/Users/kaiva/OneDrive/Desktop/
> valid.df <- d.norm.df[valid.rows, ]
> knn6 = knn(train = train.df, test = valid.df , cl = outcome[train.rows,] ,
E)
> confusionMatrix(knn6, outcome[valid.rows,], positive = "YES")
Confusion Matrix and Statistics

          Reference
Prediction NO YES
       NO  87  61
       YES 39  53

               Accuracy : 0.5833
                 95% CI : (0.5182, 0.6464)
    No Information Rate : 0.525
    P-Value [Acc > NIR] : 0.04019

                  Kappa : 0.1568

 Mcnemar's Test P-Value : 0.03573

            Sensitivity : 0.4649
            Specificity : 0.6905
         Pos Pred Value : 0.5761
         Neg Pred Value : 0.5878
             Prevalence : 0.4750
         Detection Rate : 0.2208
   Detection Prevalence : 0.3833
      Balanced Accuracy : 0.5777

       'Positive' Class : YES
```

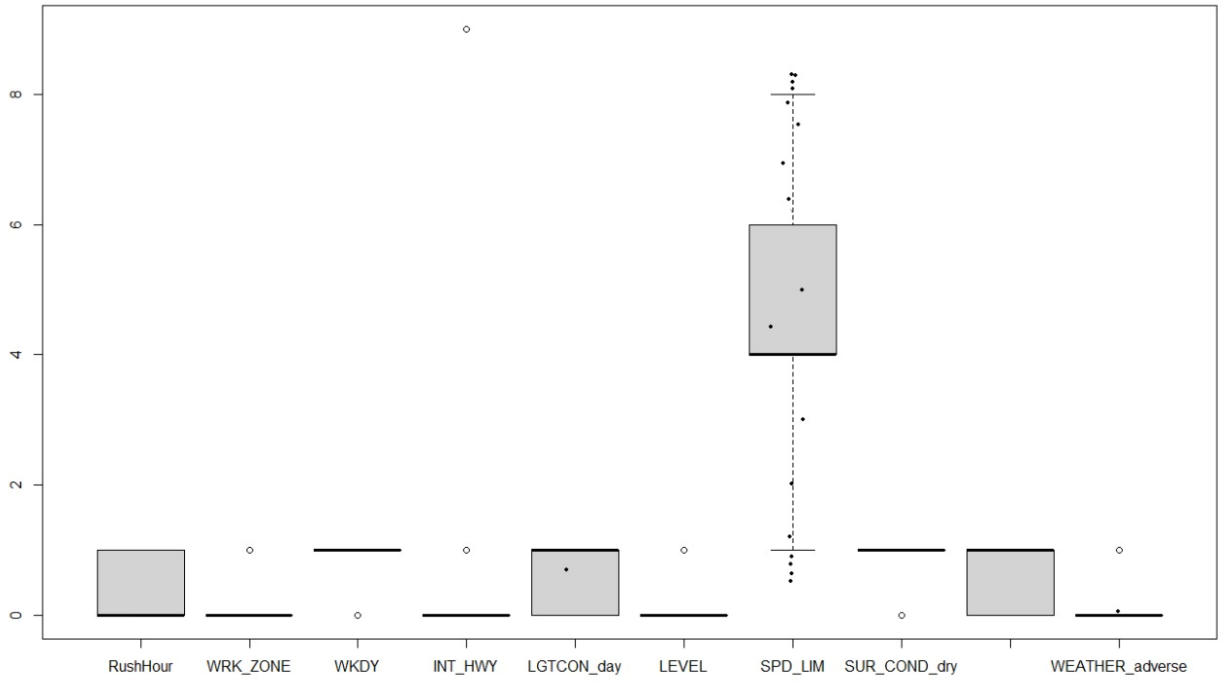
Visualization techniques used:

Here we have applied correlation to find out similarity between the unique features so, we can eliminate exploring any similar attributes and in turn narrowing down the search.

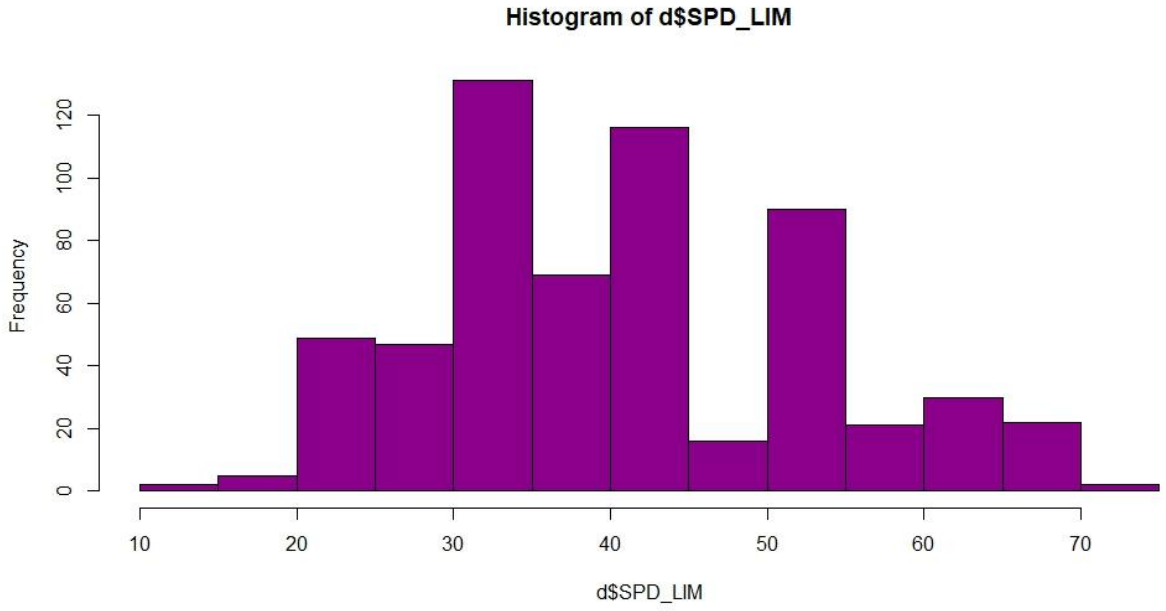


According above graph the similarities between the attributes is exceptionally low so, we can use all the attributes in predicting the severity of the accidents occurred. By considering all these features our goal is to predict the future possibility of accidents that might occur.

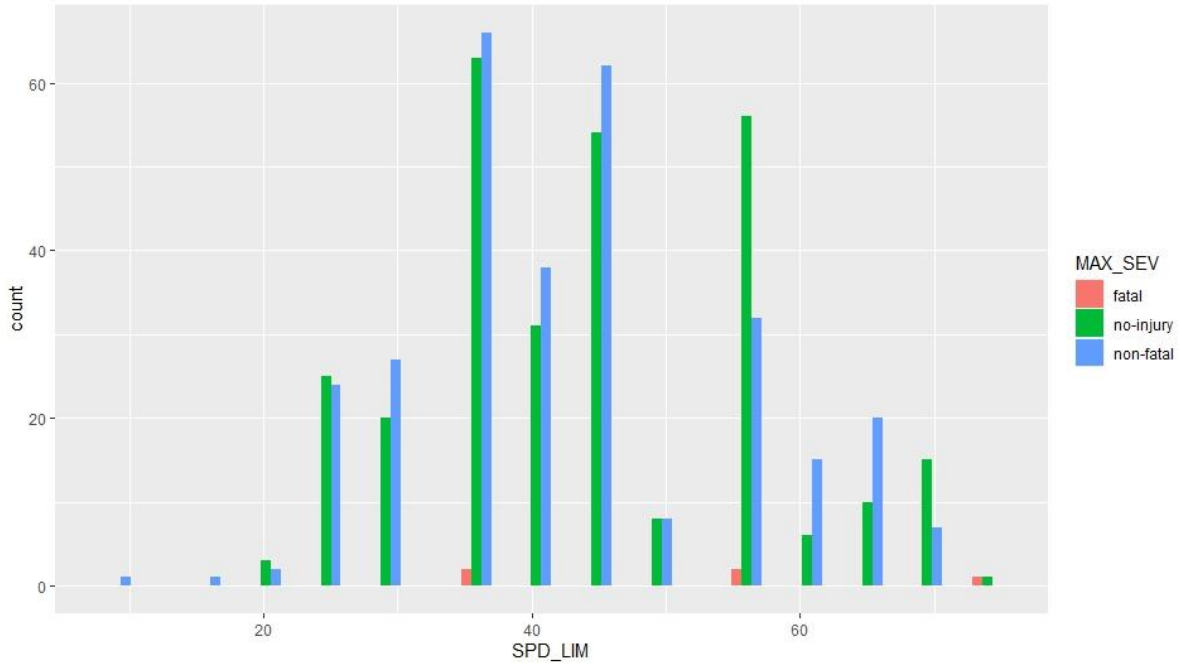
Outliers Detection: Our next process is to find if there are any outlining data present in the dataset. Outlining data might occur due to some conditions like road works or any other blockings that might occur due to accidents happened on that day on the road.



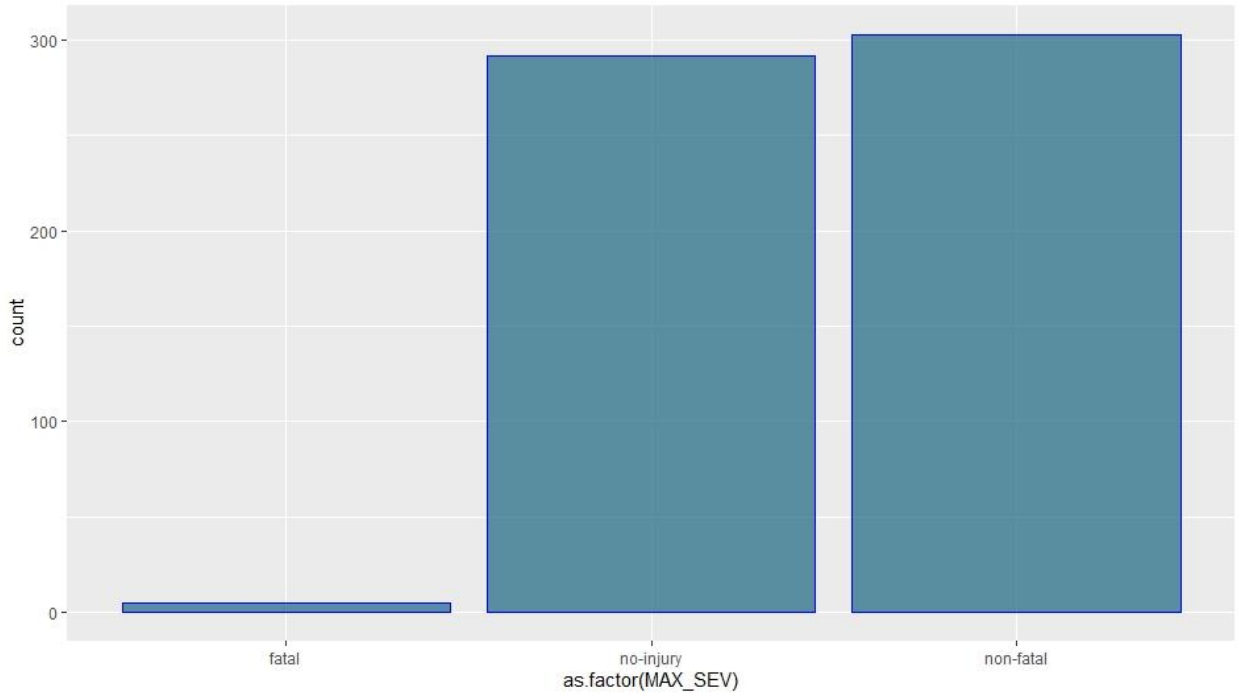
Histogram Analysis: Here we have compared the happening of injuries because of speed limit from below diagram the conclusion is that the frequency of accidents occurring is from the speed above 40



This histogram gives count of frequency of the accident severity when compared to speed limit



Bar Chart Analysis:



Optimization Conclusion:

After exploring and implementing various optimization techniques like by changing the hyper parameter values. We can get the highest accuracy by implementing the KNN algorithm with K value 6 but here the sensitivity the specificity values don't seem to be good. But at the decision tree with cp specified all the values seem to be regulated.

Git Repository:

https://github.com/susmitha7599/TheCollective