Car Accident Severity Prediction

Takao

2022-12-05

TAKAO OBA

Car Accident Prediction Project

(https://www.kaggle.com/competitions/predicting-car-accidents-severity/overview) According to kaggle.com, "This is a countrywide car accident dataset, which covers 49 states of the USA. The accident data are collected from February 2016 to Dec 2021, using multiple APIs that provide streaming traffic incident (or event) data. These APIs broadcast traffic data captured by a variety of entities, such as the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road-networks. Currently, there are about 2.8 million accident records in this dataset."

Further, this is the description for the data sets:

" The data is split into two data sets; the training data set which contains 35,000 observations with 44 variables. While the testing data contains 15,000 observations with 43 variables (No target variable).

- The purpose of this project is to be able to classify accidents as "SEVERE" or "MILD" based on the rest of the data attributes.
- The target variable is "SEVERITY" which is a categorical variable with two categories: "SEVERE" or "MILD". The "SEVERE" accidents are around 10% of the accidents, while "MILD" is the other 90% of the accidents in the training data.
- This means, your classifier misclassification rate has to beat 10% to be considered a better than a chance classifier."

Ultimately, our goal is to undergo a systematic process to classify rather an accident is "SEVERE" or "MILD"

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.2
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
```

```
## Warning: package 'ggplot2' was built under R version 4.1.2
```

```
## Warning: package 'tibble' was built under R version 4.1.2

## Warning: package 'tidyr' was built under R version 4.1.2

## Warning: package 'readr' was built under R version 4.1.2

## Warning: package 'purrr' was built under R version 4.1.2

## Warning: package 'dplyr' was built under R version 4.1.2

## Warning: package 'stringr' was built under R version 4.1.2

## Warning: package 'forcats' was built under R version 4.1.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(car)
```

```
## Warning: package 'car' was built under R version 4.1.2

## Loading required package: carData

## Warning: package 'carData' was built under R version 4.1.2

##
## Attaching package: 'car'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```
library(dplyr)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.1.2
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:gridExtra':
##
##     combine
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.1.2
```

```r
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.1.2
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.2
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(class)
```

```
## Warning: package 'class' was built under R version 4.1.2
```

```r
library(leaps)
```

```r
test <- read.csv("/Users/takaooba/Downloads/predicting-car-accidents-severity/AcctestNoYNew.csv")
test <- test[,-1]

train <- read.csv("/Users/takaooba/Downloads/predicting-car-accidents-severity/Acctrain.csv")

# head(train)
```

The dimensions can be found by the following

```r
dim(test)
```

```
## [1] 15000    43
```

```r
dim(train)
```

```
## [1] 35000    44
```

Use below to examine the two data sets:

```r
# head(test)


# head(train)

# Column names of the two data sets
colnames(test)
```

```
##  [1] "Start_Time"            "End_Time"              "Start_Lat"
##  [4] "Start_Lng"             "End_Lat"               "End_Lng"
##  [7] "Distance.mi."          "Description"           "Street"
## [10] "Side"                  "City"                  "County"
## [13] "State"                 "Zipcode"               "Country"
## [16] "Timezone"              "Airport_Code"          "Weather_Timestamp"
## [19] "Temperature.F."        "Wind_Chill.F."         "Humidity..."
## [22] "Pressure.in."          "Visibility.mi."        "Wind_Direction"
## [25] "Wind_Speed.mph."       "Weather_Condition"     "Amenity"
## [28] "Bump"                  "Crossing"              "Give_Way"
## [31] "Junction"              "No_Exit"               "Railway"
## [34] "Roundabout"            "Station"               "Stop"
## [37] "Traffic_Calming"       "Traffic_Signal"        "Turning_Loop"
## [40] "Sunrise_Sunset"        "Civil_Twilight"        "Nautical_Twilight"
## [43] "Astronomical_Twilight"
```

```r
colnames(train)
```

```
##  [1] "Severity"              "Start_Time"            "End_Time"
##  [4] "Start_Lat"             "Start_Lng"             "End_Lat"
##  [7] "End_Lng"               "Distance.mi."          "Description"
## [10] "Street"                "Side"                  "City"
## [13] "County"                "State"                 "Zipcode"
## [16] "Country"               "Timezone"              "Airport_Code"
```

```
## [19] "Weather_Timestamp"     "Temperature.F."        "Wind_Chill.F."
## [22] "Humidity..."           "Pressure.in."          "Visibility.mi."
## [25] "Wind_Direction"        "Wind_Speed.mph."       "Weather_Condition"
## [28] "Amenity"               "Bump"                  "Crossing"
## [31] "Give_Way"              "Junction"              "No_Exit"
## [34] "Railway"               "Roundabout"            "Station"
## [37] "Stop"                  "Traffic_Calming"       "Traffic_Signal"
## [40] "Turning_Loop"          "Sunrise_Sunset"        "Civil_Twilight"
## [43] "Nautical_Twilight"     "Astronomical_Twilight"
```

The numerical predictors are Start_Lat, Start_Lng, End_Lat, End_Lng, Distance.mi., Temperature.F., Wind_Chill.F., Humidity…, Pressure.in., Visibility.mi., Wind_Speed.mph. There are a total of 11 numerical predictors. This is both for the training and testing data.

The categorical predictors are Street, Side, City, Country, State, Zipcode, Country, Timezone, Airport_Code, Wind_Direction, Weather_Condition, Amenity, Bump, Crossing, Give_Way, Junction, No_Exit, Railway, Roundabout, Station, Stop, Traffic_Calming, Traffic Signal, Turning_Loop, Sunrise_Sunset, Civil_Twilight, Nautical_Twilight, Astronomical_Twilight There are a total of 29 categorical predictors. This is both for the training and testing data.

Next, I aim to impute missing values.

```
# Testing Data
sum((is.na(test)))
```

```
## [1] 5842
```

```
# Training Data
sum(is.na(train))
```

```
## [1] 13211
```

```
# Total NA's in both data sets
sum((is.na(test))) + sum(is.na(train))
```

```
## [1] 19053
```

As a temporary step, omit the NA's value using na.omit and further assess the best predictors that will be used when constructing the model.

```
train.1 <- na.omit(train)
# head(train.1)
train.1$SeverityNum <- ifelse(train.1$Severity == "MILD", 0, 1)
numericalpredictor <- train.1[,c(4,5,6,7,8,20,21,22,23,24,26,45)]
cor(numericalpredictor)
```

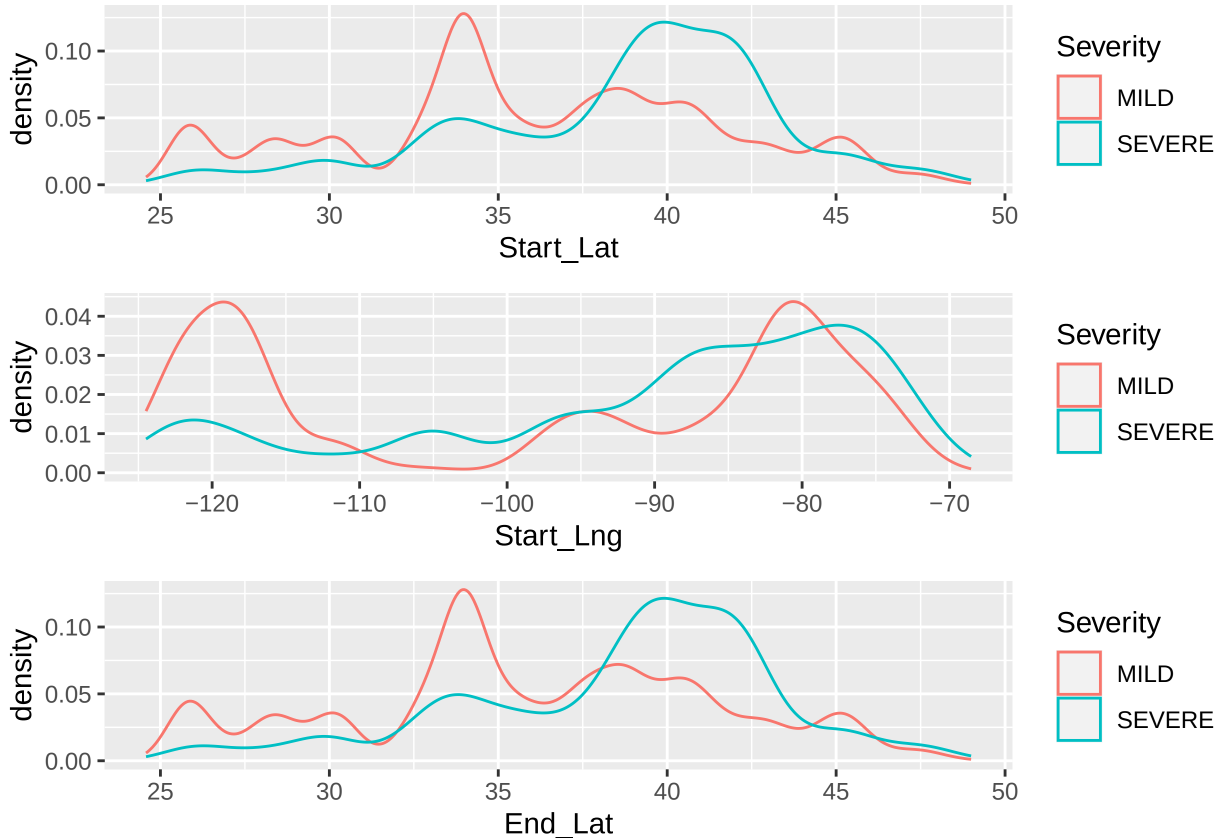
```
##                    Start_Lat   Start_Lng      End_Lat     End_Lng Distance.mi.
## Start_Lat        1.000000000 -0.16112099  0.999995357 -0.16111539   0.07759972
## Start_Lng       -0.161120989  1.00000000 -0.161127433  0.99999911   0.03032042
## End_Lat          0.999995357 -0.16112743  1.000000000 -0.16112227   0.07770666
## End_Lng         -0.161115387  0.99999911 -0.161122270  1.00000000   0.03045833
## Distance.mi.     0.077599716  0.03032042  0.077706656  0.03045833   1.00000000
```

```
## Temperature.F.  -0.493369968  0.02776738 -0.493354441  0.02776064  -0.05213097
## Wind_Chill.F.   -0.498391152  0.01064582 -0.498376917  0.01063811  -0.05731444
## Humidity...      0.007476358  0.16206716  0.007467002  0.16204723   0.02770460
## Pressure.in.    -0.261356822  0.22581863 -0.261341877  0.22580454  -0.07119964
## Visibility.mi.  -0.094135159  0.03483039 -0.094125943  0.03483449  -0.03946985
## Wind_Speed.mph.  0.033185546  0.11498774  0.033177598  0.11500546   0.02530208
## SeverityNum      0.122739891  0.10216806  0.122719408  0.10216720   0.04163956
##                 Temperature.F. Wind_Chill.F.  Humidity... Pressure.in.
## Start_Lat           -0.49336997  -0.498391152  0.007476358  -0.26135682
## Start_Lng            0.02776738   0.010645816  0.162067156   0.22581863
## End_Lat             -0.49335444  -0.498376917  0.007467002  -0.26134188
## End_Lng              0.02776064   0.010638109  0.162047225   0.22580454
## Distance.mi.        -0.05213097  -0.057314440  0.027704596  -0.07119964
## Temperature.F.       1.00000000   0.993757045 -0.374606921   0.11656740
## Wind_Chill.F.        0.99375704   1.000000000 -0.356542306   0.12255842
## Humidity...         -0.37460692  -0.356542306  1.000000000   0.15126431
## Pressure.in.         0.11656740   0.122558422  0.151264314   1.00000000
## Visibility.mi.       0.21708604   0.219076422 -0.369635702   0.02173128
## Wind_Speed.mph.      0.06196195   0.005712984 -0.170450319  -0.05529463
## SeverityNum         -0.08497974  -0.094599872  0.022847681  -0.03724430
##                 Visibility.mi. Wind_Speed.mph.  SeverityNum
## Start_Lat          -0.094135159     0.033185546  0.122739891
## Start_Lng           0.034830391     0.114987744  0.102168059
## End_Lat            -0.094125943     0.033177598  0.122719408
## End_Lng             0.034834492     0.115005458  0.102167198
## Distance.mi.       -0.039469847     0.025302077  0.041639564
## Temperature.F.      0.217086037     0.061961950 -0.084979743
## Wind_Chill.F.       0.219076422     0.005712984 -0.094599872
## Humidity...        -0.369635702    -0.170450319  0.022847681
## Pressure.in.        0.021731275    -0.055294629 -0.037244301
## Visibility.mi.      1.000000000     0.025227688  0.006555872
## Wind_Speed.mph.     0.025227688     1.000000000  0.060132993
## SeverityNum         0.006555872     0.060132993  1.000000000
```

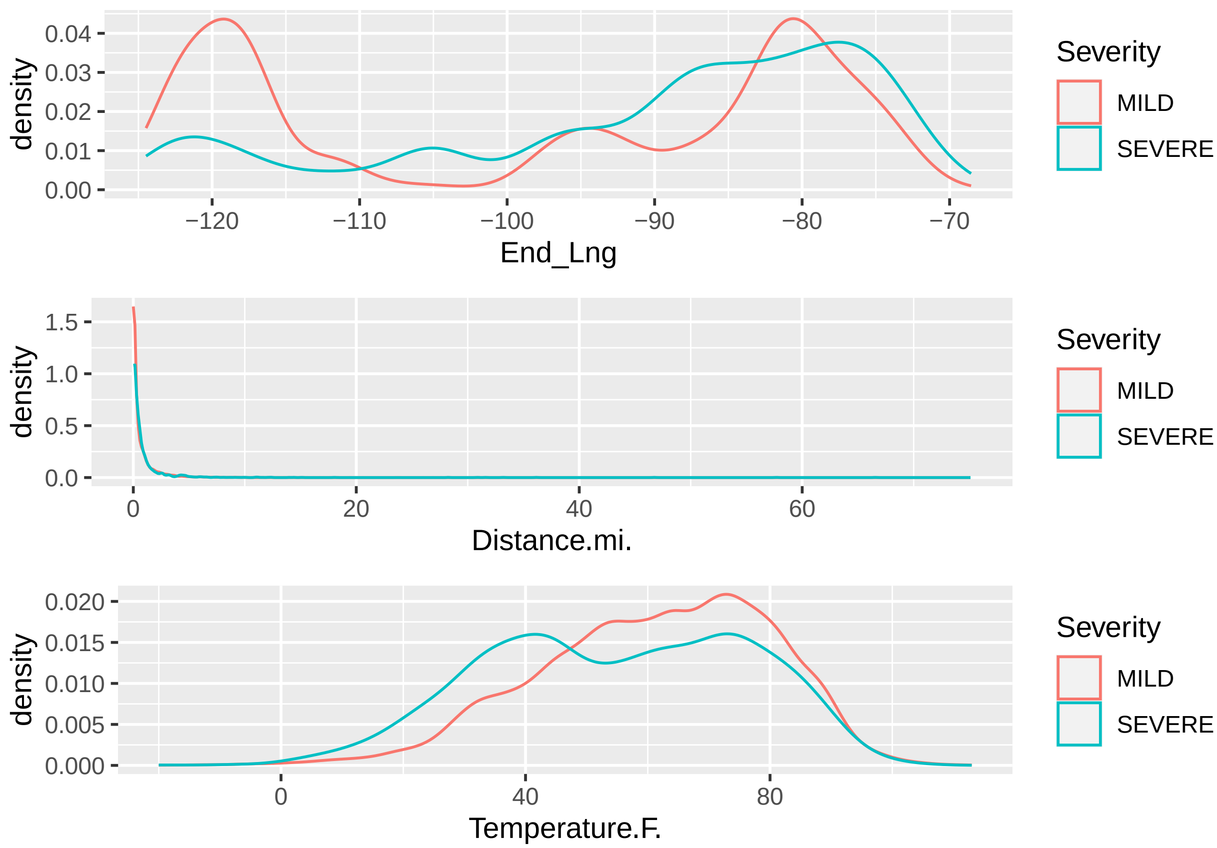
Based on the correlation plot that I have just created above, I have that the best predictors are Start_Lat, End_Lat, Start_Lng, End_Lng, Wine_Chill.F., Wind_Speed.mph.

Next, I plot graphs to see which predictors have distinct humps based on if the Severity is "MILD" or "SEVERE"

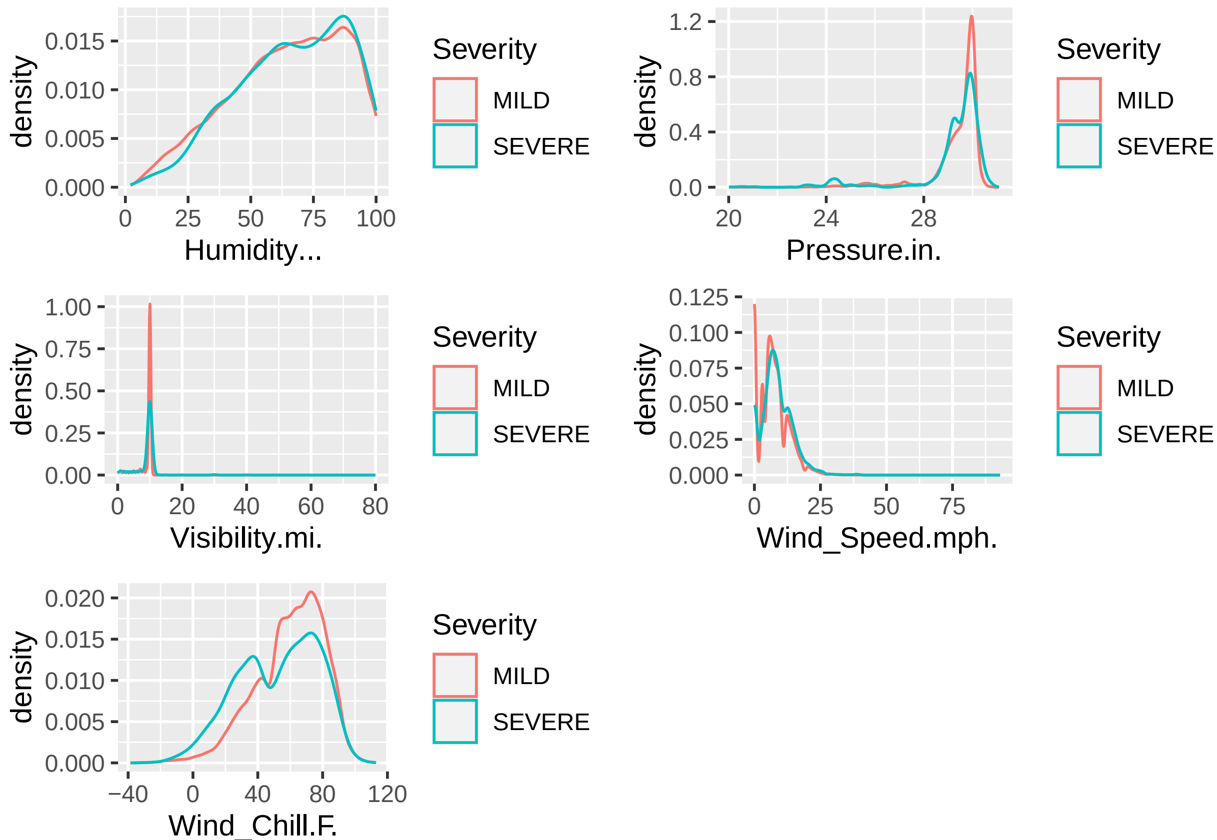
```
library(ggplot2)
ggstart_lat <- ggplot(train.1, aes(Start_Lat, group = Severity, color = Severity  )) + geom_density()
ggstart_lng <- ggplot(train.1, aes(Start_Lng, group = Severity, color = Severity  )) + geom_density()
ggend_lat <- ggplot(train.1, aes(End_Lat, group = Severity, color = Severity  )) + geom_density()
ggend_lng <- ggplot(train.1, aes(End_Lng, group = Severity, color = Severity  )) + geom_density()
ggdistance <- ggplot(train.1, aes(Distance.mi., group = Severity, color = Severity  )) + geom_density()
ggtemperature <- ggplot(train.1, aes(Temperature.F., group = Severity, color = Severity  )) + geom_densi
gghumidity <- ggplot(train.1, aes(Humidity..., group = Severity, color = Severity  )) + geom_density()
ggpressure <- ggplot(train.1, aes(Pressure.in., group = Severity, color = Severity  )) + geom_density()
ggvisibility <- ggplot(train.1, aes(Visibility.mi., group = Severity, color = Severity  )) + geom_densi
ggwind_speed <- ggplot(train.1, aes(Wind_Speed.mph., group = Severity, color = Severity  )) + geom_densi
ggwind_chill <- ggplot(train.1, aes(Wind_Chill.F., group = Severity, color = Severity  )) + geom_density

library(gridExtra)
grid.arrange(ggstart_lat, ggstart_lng, ggend_lat)
```




```
grid.arrange(ggend_lng, ggdistance, ggtemperature)
```



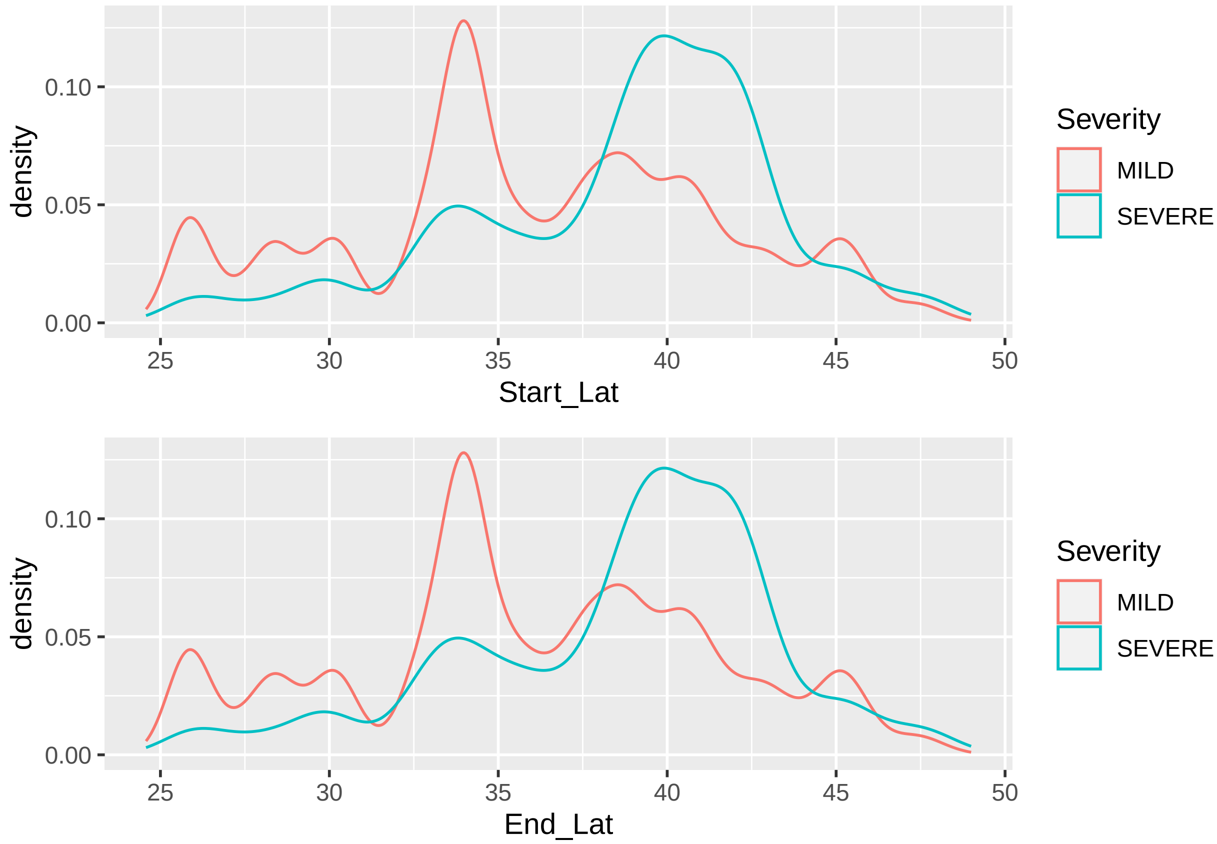

```
grid.arrange(gghumidity, ggpressure, ggvisibility, ggwind_speed, ggwind_chill)
```



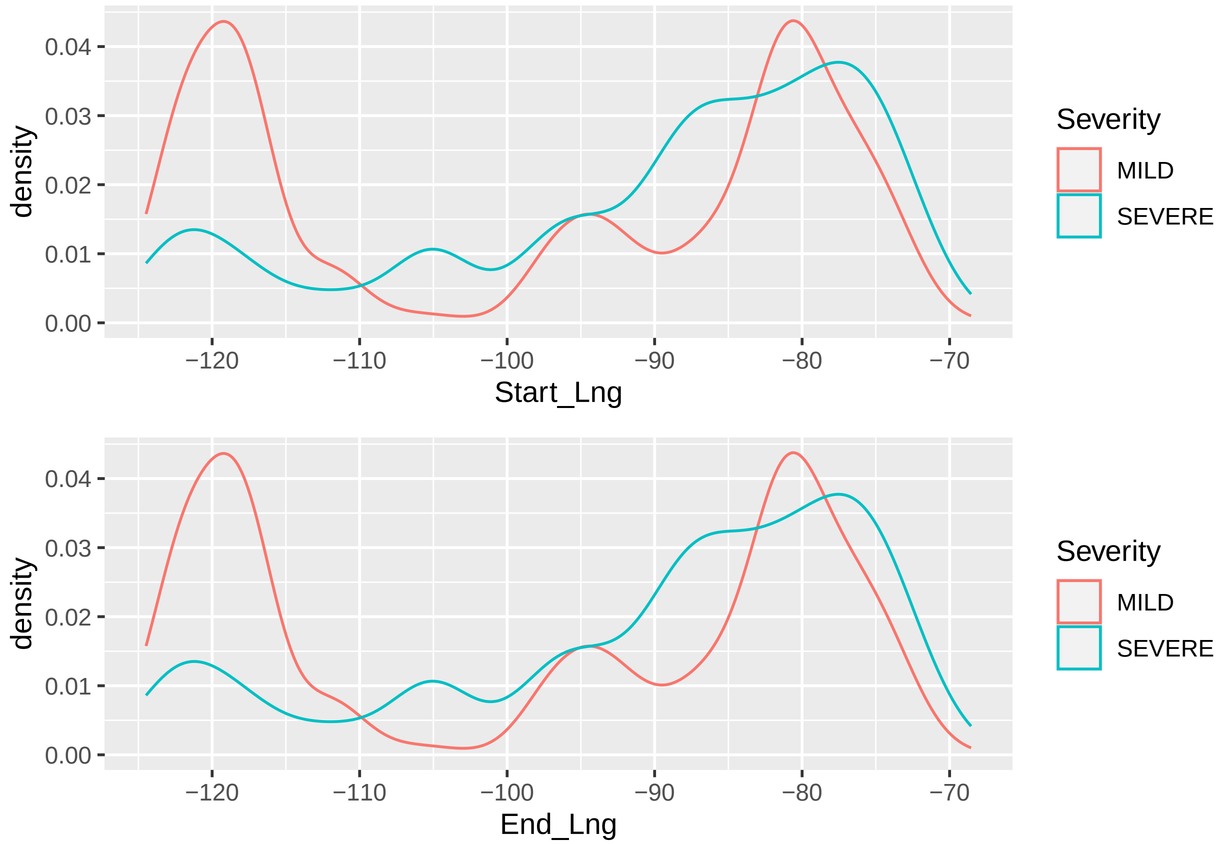
Based on the above graphs, I have that the 6 best numerical variables are Start_Lat, End_Lat, Start_Lng, End_Lng, Wind_Chill.F., Temperature.F.

I will continue to plot these 6 numerical variables

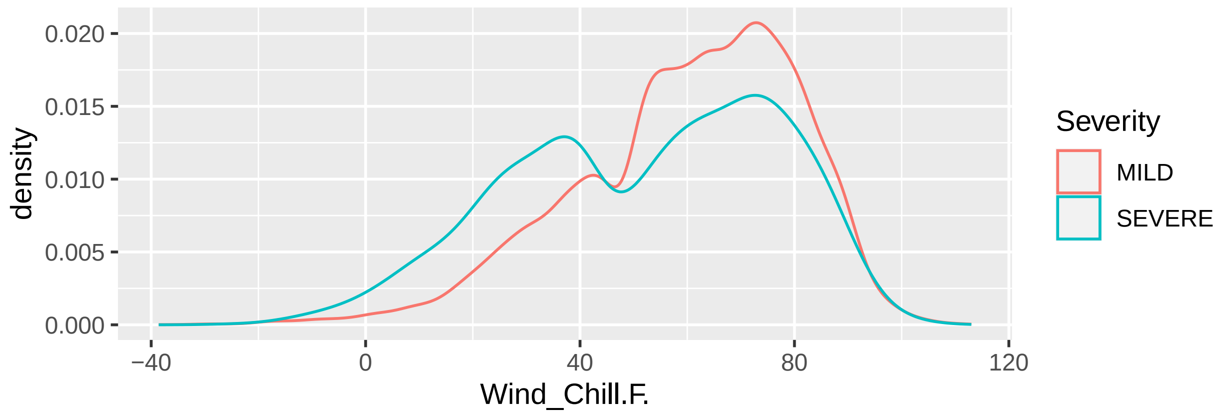
```
grid.arrange(ggstart_lat, ggend_lat)
```

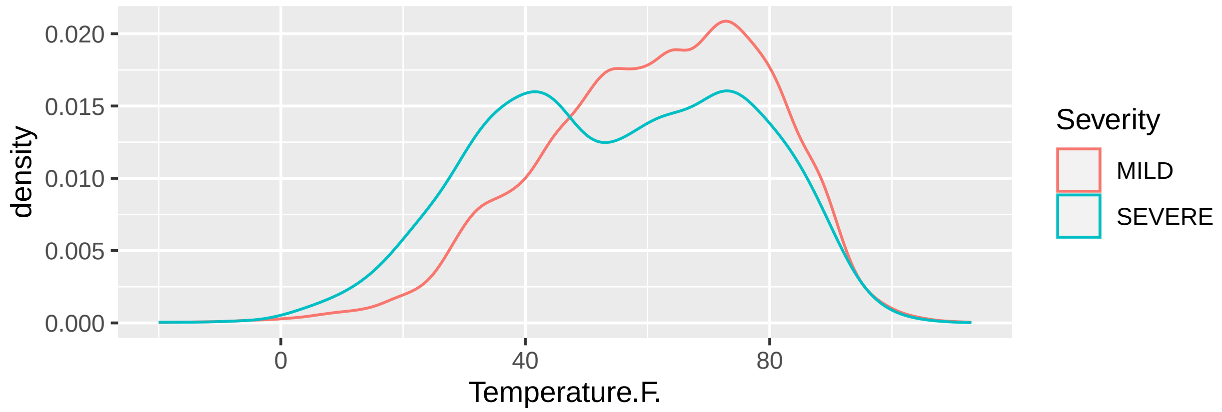



```
grid.arrange(ggstart_lng, ggend_lng)
```




```
grid.arrange(ggwind_chill, ggtemperature)
```

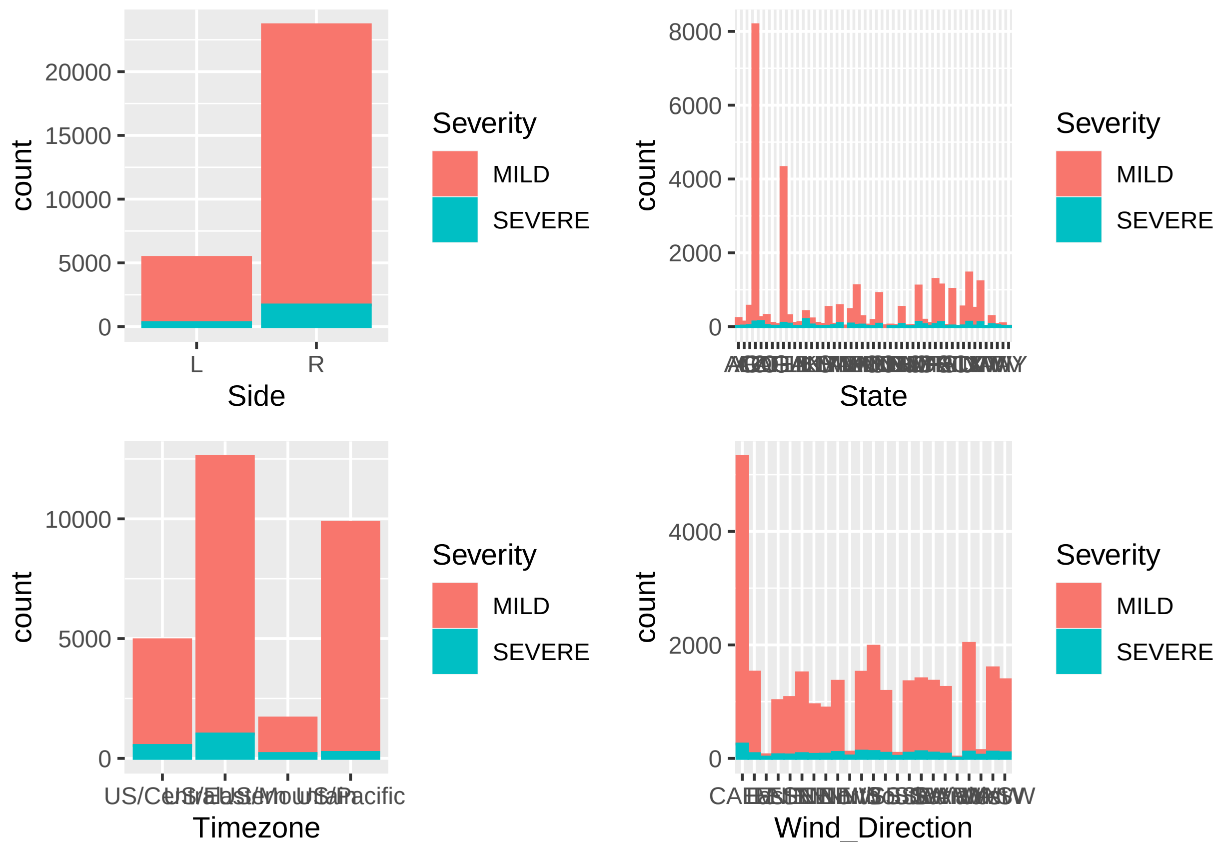




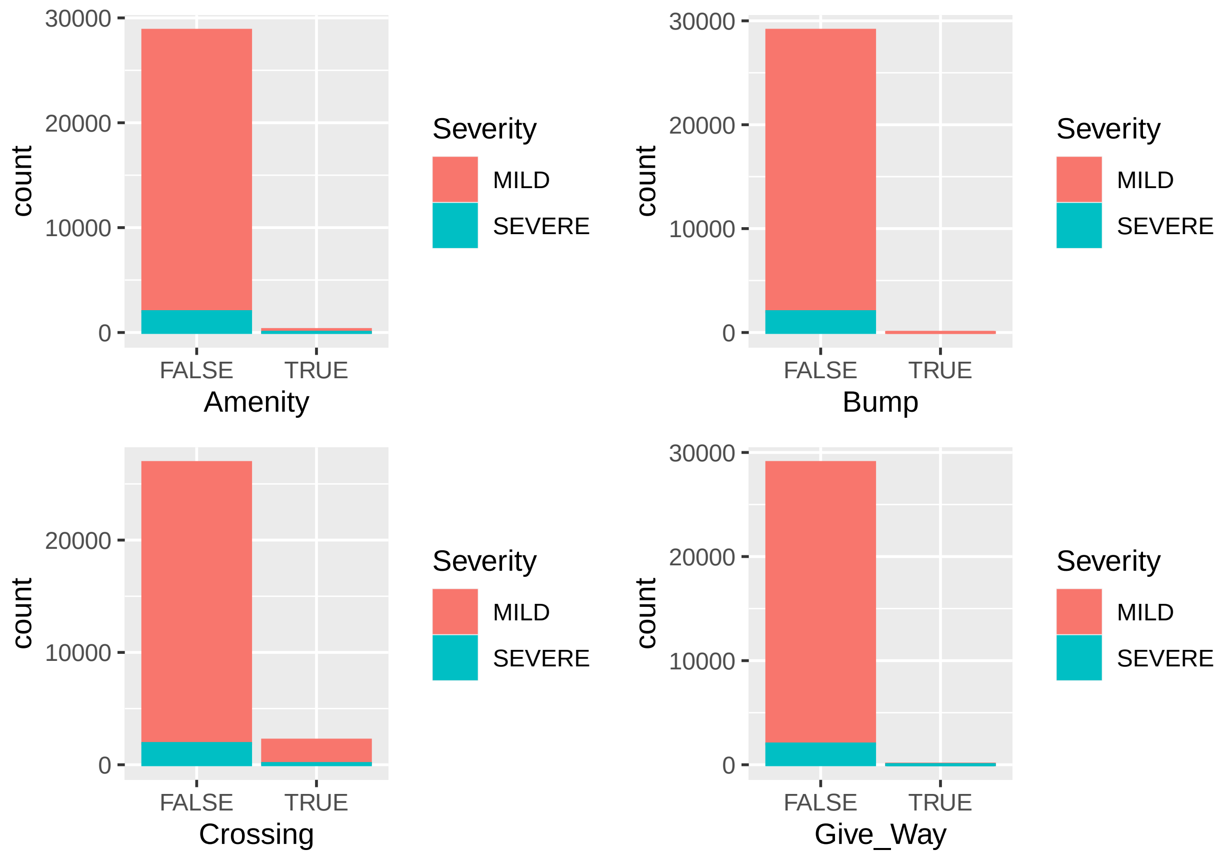
Next, I will construct stacked bar charts for the best three categorical predictors based on the response variables.

```
#
# Street, Side, City, Country, State, Zipcode, Country, Timezone, Airport_Code, Wind_Direction, Weather

# However, I will need to determine which predictors makes sense in the first place in the context of t

ggside <- ggplot(train.1, aes(Side, group = Severity, color = Severity  , fill = Severity)) + geom_bar(
ggstate <- ggplot(train.1, aes(State, group = Severity, color = Severity  , fill = Severity)) + geom_ba
ggtimezone <- ggplot(train.1, aes(Timezone, group = Severity, color = Severity  , fill = Severity)) + g
ggwinddirection <- ggplot(train.1, aes(Wind_Direction, group = Severity, color = Severity  , fill = Sev
ggamenity <- ggplot(train.1, aes(Amenity, group = Severity, color = Severity  , fill = Severity)) + geo
ggbump <- ggplot(train.1, aes(Bump, group = Severity, color = Severity , fill = Severity )) + geom_bar(
ggcrossing <- ggplot(train.1, aes(Crossing, group = Severity, color = Severity , fill = Severity )) + g
gggiveway <- ggplot(train.1, aes(Give_Way, group = Severity, color = Severity , fill = Severity )) + ge
ggjunction <- ggplot(train.1, aes(Junction, group = Severity, color = Severity , fill = Severity )) + g
ggnoexit <- ggplot(train.1, aes(No_Exit, group = Severity, color = Severity , fill = Severity )) + geom
ggrailway <- ggplot(train.1, aes(Railway, group = Severity, color = Severity  , fill = Severity)) + geo
ggroundabout <- ggplot(train.1, aes(Roundabout, group = Severity, color = Severity  , fill = Severity))
ggstation <- ggplot(train.1, aes(Station, group = Severity, color = Severity , fill = Severity )) + geo
ggstop <- ggplot(train.1, aes(Stop, group = Severity, color = Severity , fill = Severity )) + geom_bar(
ggcalm <- ggplot(train.1, aes(Traffic_Calming, group = Severity, color = Severity  , fill = Severity)) 
ggsignal <- ggplot(train.1, aes(Traffic_Signal, group = Severity, color = Severity , fill = Severity ))
ggloop <- ggplot(train.1, aes(Turning_Loop, group = Severity, color = Severity , fill = Severity )) + g
ggsunset<- ggplot(train.1, aes(Sunrise_Sunset, group = Severity, color = Severity  , fill = Severity)) 
```

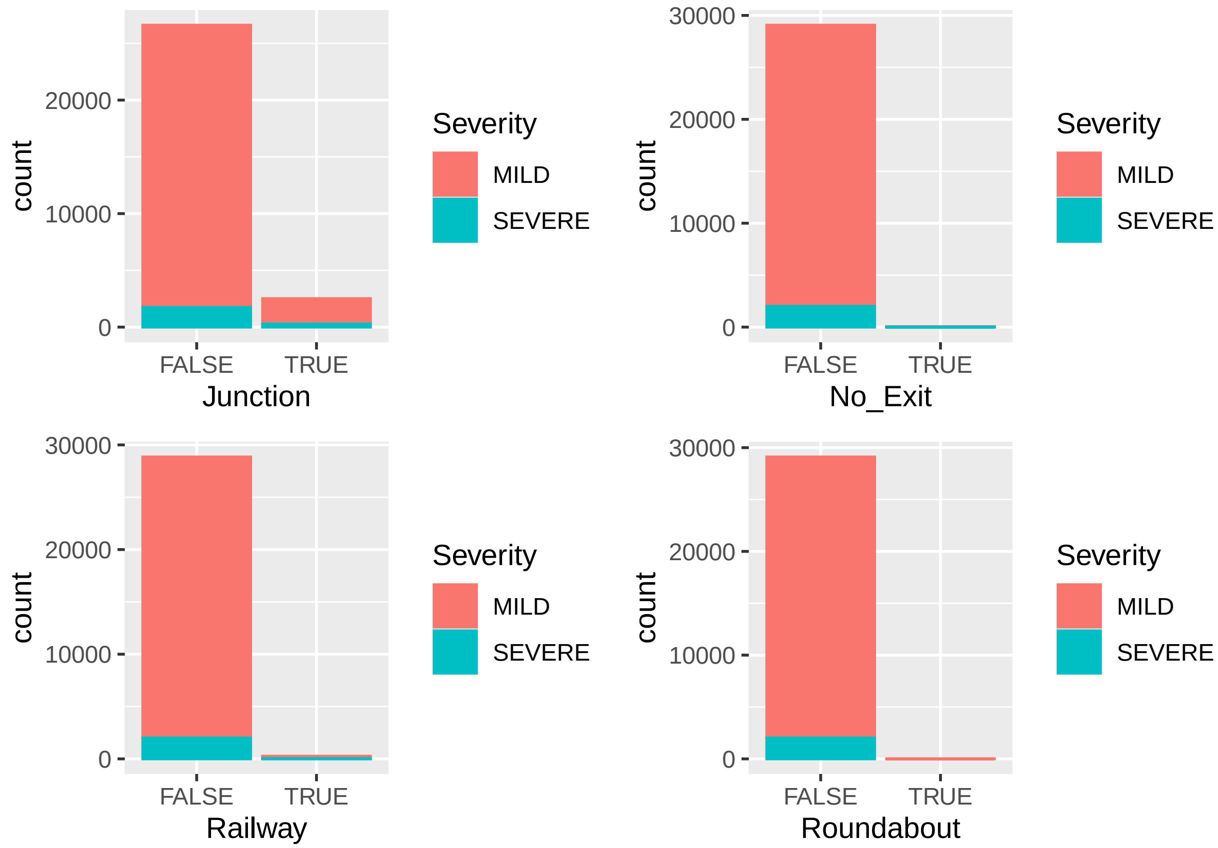
```
ggcivil <- ggplot(train.1, aes(Civil_Twilight, group = Severity, color = Severity , fill = Severity )) +
ggnautical <- ggplot(train.1, aes(Nautical_Twilight, group = Severity, color = Severity , fill = Severit
ggastronomical <- ggplot(train.1, aes(Astronomical_Twilight, group = Severity, color = Severity , fill =


library(gridExtra)
grid.arrange(ggside, ggstate, ggtimezone,ggwinddirection)
```



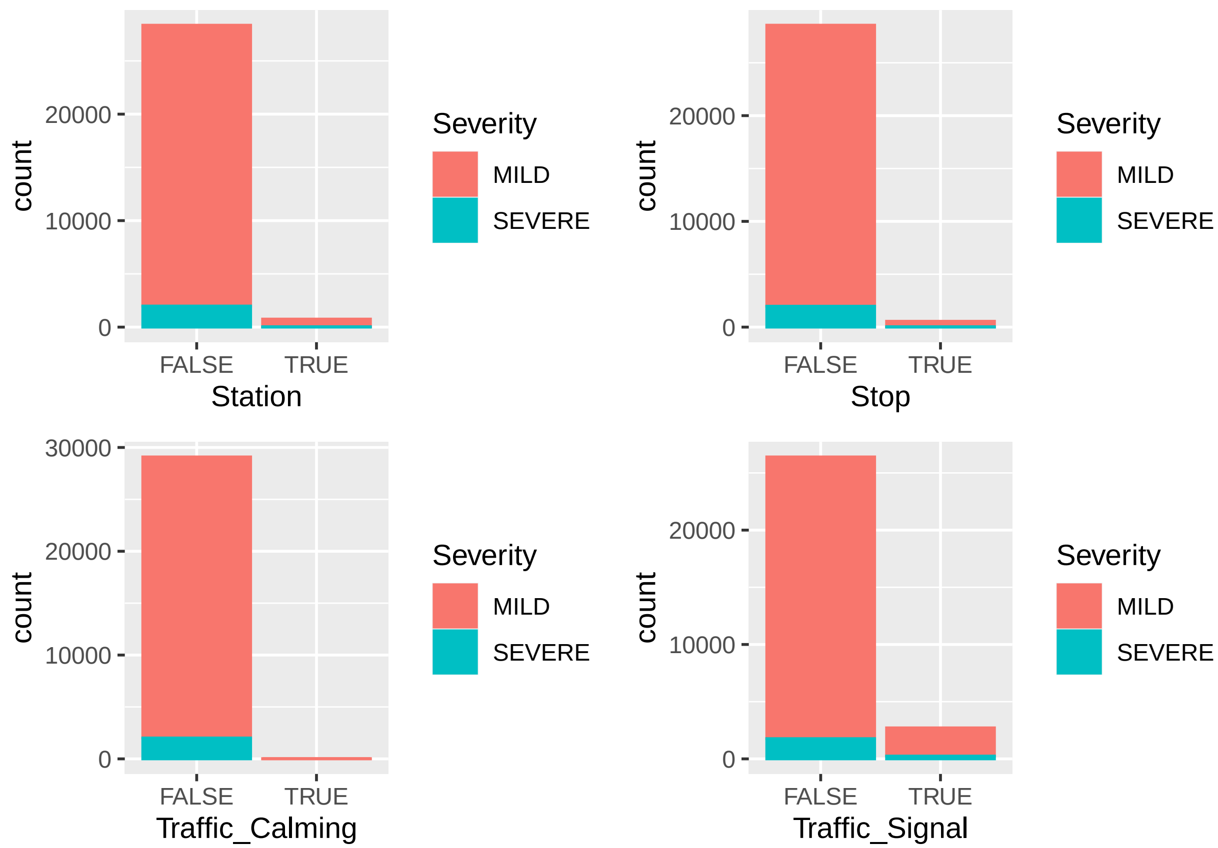

```
grid.arrange(ggamenity, ggbump,ggcrossing, gggiveway)
```



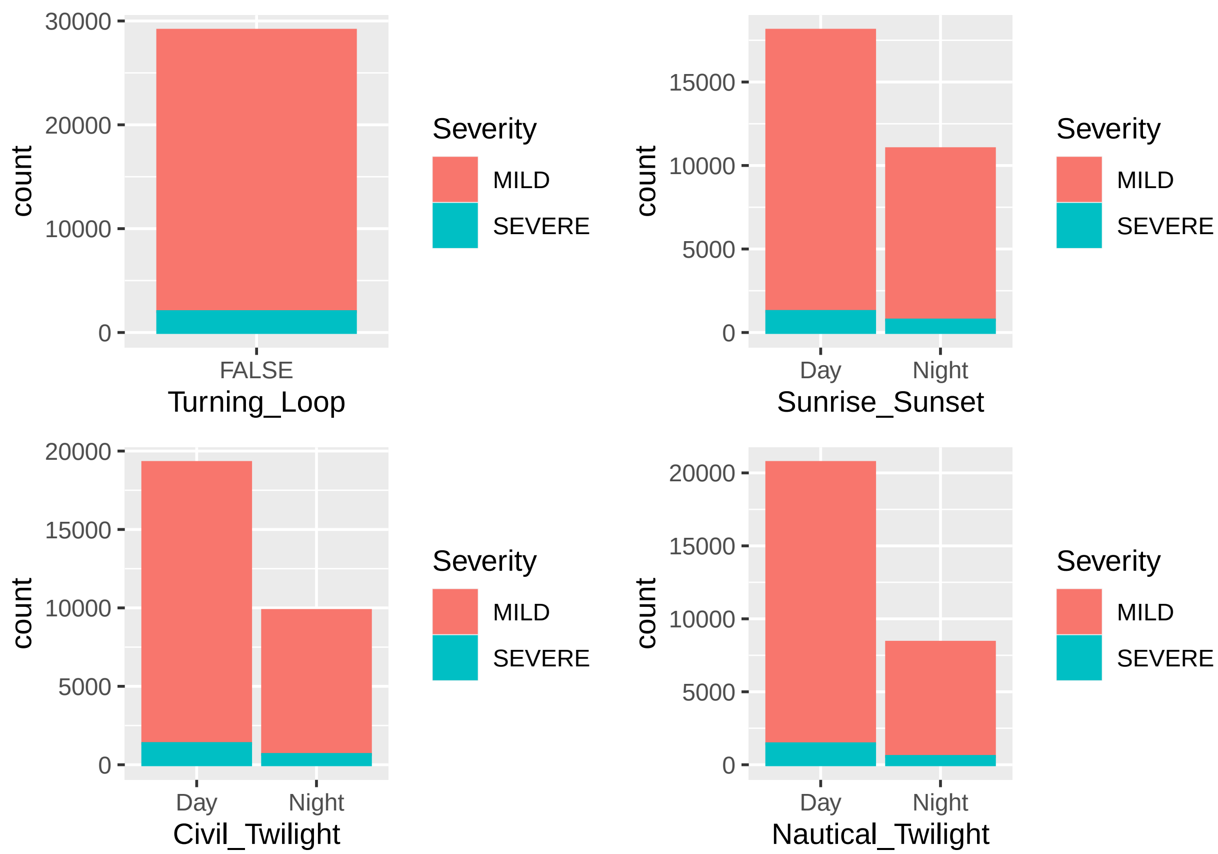

```
grid.arrange(ggjunction, ggnoexit, ggrailway, ggroundabout)
```



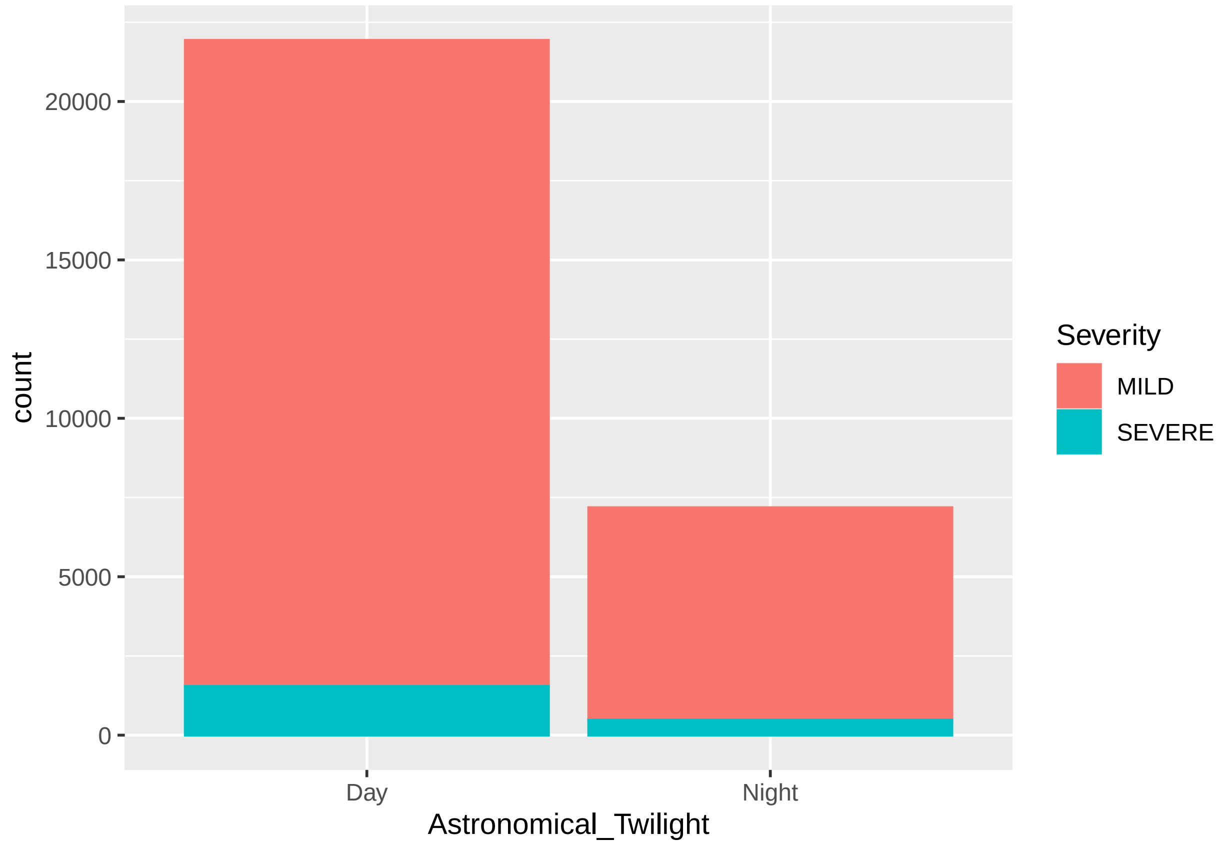

```
grid.arrange(ggstation, ggstop, ggcalm, ggsignal)
```




```
grid.arrange(ggloop, ggsunset, ggcivil, ggnautical)
```




```
grid.arrange(ggastronomical)
```



I aim to look at the proportion of the Severity (either MILD or SEVERE) between the different bars. I want to see the proportion of the MILD and SEVERE to be different between the bars and will look at these proportions to determine the best predictors.

The best categorical predictors based on above are Timezone, Give_Way, and Traffic_Signal.

Creating a testing data set within the training set

```
set.seed(717)
sam <- sample(1:35000, 5000, replace = F)
train.x <- train[,-1]
train.y <- train[,1]

# Fitting values into training and testing set (x and y)
train_sam_x <- train.x[-sam,]
test_sam_x <- train.x[sam,]
train_sam_y <- train.y[-sam]
test_sam_y <- train.y[sam]


head(train_sam_x)
```

```
##             Start_Time             End_Time Start_Lat  Start_Lng  End_Lat
## 1 2021-01-06T13:58:00Z 2021-01-06T16:41:33Z  30.30729  -97.72562 30.30702
```

```
## 2 2021-04-03T11:08:00Z 2021-04-03T12:26:27Z  38.07553 -122.54181 38.07913
## 3 2019-05-09T14:19:17Z 2019-05-09T14:46:43Z  25.81245  -80.21472 25.81242
## 4 2021-11-15T12:22:30Z 2021-11-15T12:46:30Z  34.99765  -82.05707 34.99737
## 5 2021-12-14T10:24:32Z 2021-12-14T12:09:32Z  45.50310 -118.42311 45.50258
## 6 2020-04-11T04:34:29Z 2020-04-11T05:09:27Z  38.61153 -121.51080 38.61153
##      End_Lng Distance.mi.
## 1  -97.72503        0.040
## 2 -122.54512        0.307
## 3  -80.21219        0.157
## 4  -82.05515        0.110
## 5 -118.42264        0.042
## 6 -121.51080        0.000
##                                                                                   Descri
## 1                                                 Incident on E 45TH ST near AVENUE H Drive with cau
## 2                                                    Incident on US-101 NB near CA-37 Drive with cau
## 3                          Ramp closed to I-95 and I-95 Northbound Express Ln - Road closed due to acci
## 4 Stationary traffic on SC-40 from Mitchell Rd (New Cut Rd) to John Dodd Rd (New Cut Rd) due to acci
## 5                                                     Incident on I-84 EB near MP 238 Drive with cau
## 6                                    At Garden Hwy/Exit 521A/Exit 521 - Accident. Hard shoulder blo
##          Street Side       City      County State    Zipcode Country
## 1       Avenue H    R     Austin      Travis    TX 78751-3122      US
## 2  Redwood Hwy S    R     Novato       Marin    CA      94945      US
## 3 Airport Expy E    R      Miami  Miami-Dade    FL      33127      US
## 4     New Cut Rd    R      Inman Spartanburg    SC 29349-4532      US
## 5         I-84 E    R  Pendleton    Umatilla    OR      97801      US
## 6        CA-16 E    R Sacramento  Sacramento    CA      95833      US
##     Timezone Airport_Code    Weather_Timestamp Temperature.F. Wind_Chill.F.
## 1 US/Central         KATT 2021-01-06T13:51:00Z             65            65
## 2 US/Pacific         KDVO 2021-04-03T11:15:00Z             54            54
## 3 US/Eastern         KMIA 2019-05-09T13:53:00Z             87            87
## 4 US/Eastern         KSPA 2021-11-15T12:15:00Z             55            55
## 5 US/Pacific         KPDT 2021-12-14T10:53:00Z             38            30
## 6 US/Pacific         KMCC 2020-04-11T04:50:00Z             46            44
##   Humidity... Pressure.in. Visibility.mi. Wind_Direction Wind_Speed.mph.
## 1          50        29.17             10             NW              10
## 2          67        30.10             10            WNW               9
## 3          61        29.96             10             SE              13
## 4          38        29.29             10              W               3
## 5          60        28.20             10             SW              12
## 6          93        29.88             10            SSE               5
##   Weather_Condition Amenity  Bump Crossing Give_Way Junction No_Exit Railway
## 1     Partly Cloudy   FALSE FALSE     TRUE    FALSE    FALSE   FALSE   FALSE
## 2            Cloudy   FALSE FALSE    FALSE    FALSE     TRUE   FALSE   FALSE
## 3     Mostly Cloudy   FALSE FALSE    FALSE    FALSE    FALSE   FALSE   FALSE
## 4              Fair   FALSE FALSE    FALSE    FALSE    FALSE   FALSE   FALSE
## 5              Fair   FALSE FALSE    FALSE    FALSE    FALSE   FALSE   FALSE
## 6              Fair   FALSE FALSE    FALSE    FALSE    FALSE   FALSE   FALSE
##   Roundabout Station  Stop Traffic_Calming Traffic_Signal Turning_Loop
## 1      FALSE   FALSE FALSE           FALSE          FALSE        FALSE
## 2      FALSE   FALSE FALSE           FALSE          FALSE        FALSE
## 3      FALSE   FALSE FALSE           FALSE          FALSE        FALSE
## 4      FALSE   FALSE FALSE           FALSE          FALSE        FALSE
## 5      FALSE   FALSE FALSE           FALSE          FALSE        FALSE
## 6      FALSE   FALSE FALSE           FALSE          FALSE        FALSE
```

```
##   Sunrise_Sunset Civil_Twilight Nautical_Twilight Astronomical_Twilight
## 1            Day            Day               Day                   Day
## 2            Day            Day               Day                   Day
## 3            Day            Day               Day                   Day
## 4            Day            Day               Day                   Day
## 5            Day            Day               Day                   Day
## 6          Night          Night             Night                 Night
```

```
head(train_sam_y)
```

```
## [1] "MILD"   "MILD"   "SEVERE" "MILD"   "MILD"   "MILD"
```

Inputing missing values

To visualize which predictors has the most missing values, I will be looking at below.

```
library(VIM)
```

```
## Warning: package 'VIM' was built under R version 4.1.2
```

```
## Loading required package: colorspace
```

```
## Warning: package 'colorspace' was built under R version 4.1.2
```

```
## Loading required package: grid
```

```
## VIM is ready to use.
```

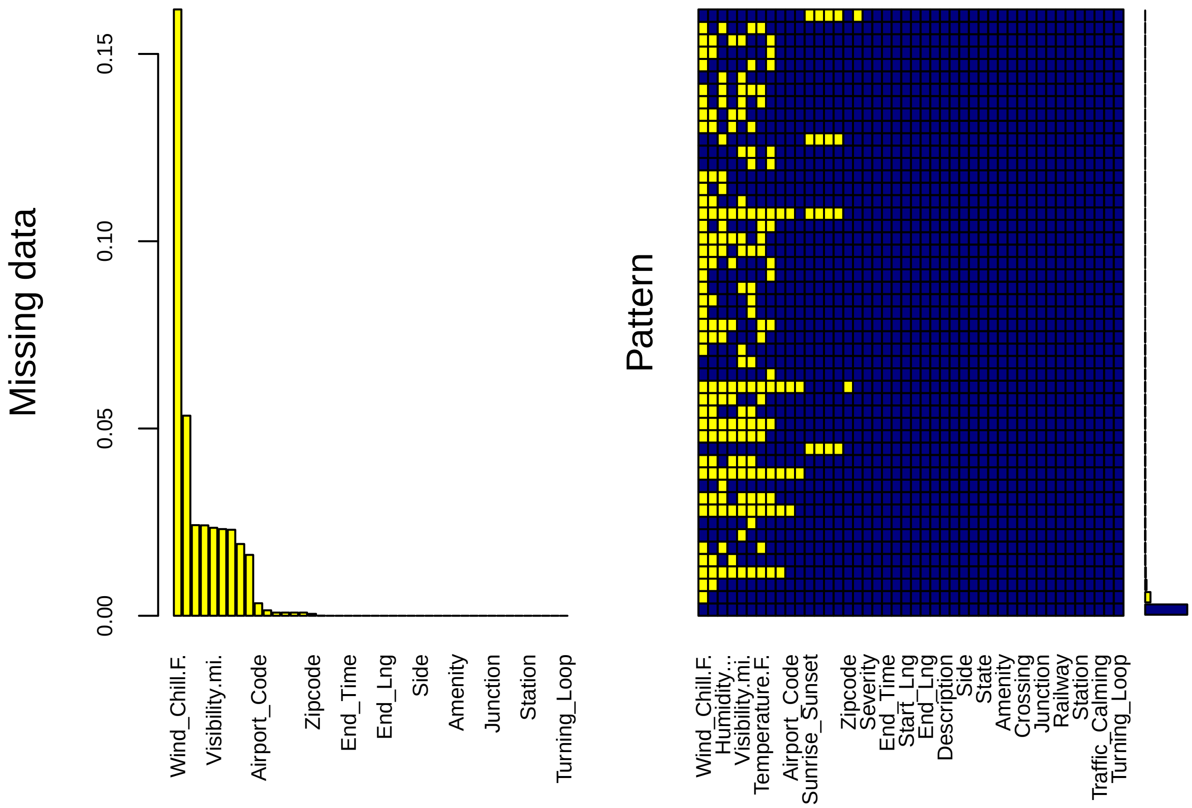
```
## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues
```

```
##
## Attaching package: 'VIM'
```

```
## The following object is masked from 'package:datasets':
##
##     sleep
```

```
acc_plot <- aggr(train, col=c('navyblue','yellow'), numbers=TRUE, sortVars=TRUE, labels=names(train), c
```

```
## Warning in plot.aggr(res, ...): not enough vertical space to display frequencies
## (too many combinations)
```




```
##
##  Variables sorted by number of missings:
##               Variable        Count
##           Wind_Chill.F. 1.618857e-01
##         Wind_Speed.mph. 5.342857e-02
##             Humidity... 2.420000e-02
##          Wind_Direction 2.414286e-02
##          Visibility.mi. 2.348571e-02
##       Weather_Condition 2.314286e-02
##          Temperature.F. 2.297143e-02
##            Pressure.in. 1.917143e-02
##       Weather_Timestamp 1.625714e-02
##            Airport_Code 3.342857e-03
##                Timezone 1.457143e-03
##          Sunrise_Sunset 8.571429e-04
##          Civil_Twilight 8.571429e-04
##       Nautical_Twilight 8.571429e-04
##  Astronomical_Twilight 8.571429e-04
##                 Zipcode 5.142857e-04
##                    City 2.857143e-05
##                Severity 0.000000e+00
##              Start_Time 0.000000e+00
##                End_Time 0.000000e+00
##               Start_Lat 0.000000e+00
##               Start_Lng 0.000000e+00
##                 End_Lat 0.000000e+00
```

```
##                End_Lng 0.000000e+00
##           Distance.mi. 0.000000e+00
##            Description 0.000000e+00
##                 Street 0.000000e+00
##                   Side 0.000000e+00
##                 County 0.000000e+00
##                  State 0.000000e+00
##                Country 0.000000e+00
##                Amenity 0.000000e+00
##                   Bump 0.000000e+00
##               Crossing 0.000000e+00
##               Give_Way 0.000000e+00
##               Junction 0.000000e+00
##                No_Exit 0.000000e+00
##                Railway 0.000000e+00
##             Roundabout 0.000000e+00
##                Station 0.000000e+00
##                   Stop 0.000000e+00
##        Traffic_Calming 0.000000e+00
##         Traffic_Signal 0.000000e+00
##           Turning_Loop 0.000000e+00
```

To better impute the variables, I will temporarily combine the testing and training data.

```
dim(test)
```

```
## [1] 15000    43
```

```
dim(train.x)
```

```
## [1] 35000    43
```

```
comb.df <- rbind(train.x, test)
head(comb.df)
```

```
##             Start_Time             End_Time Start_Lat  Start_Lng  End_Lat
## 1 2021-01-06T13:58:00Z 2021-01-06T16:41:33Z  30.30729  -97.72562 30.30702
## 2 2021-04-03T11:08:00Z 2021-04-03T12:26:27Z  38.07553 -122.54181 38.07913
## 3 2019-05-09T14:19:17Z 2019-05-09T14:46:43Z  25.81245  -80.21472 25.81242
## 4 2021-11-15T12:22:30Z 2021-11-15T12:46:30Z  34.99765  -82.05707 34.99737
## 5 2021-12-14T10:24:32Z 2021-12-14T12:09:32Z  45.50310 -118.42311 45.50258
## 6 2020-04-11T04:34:29Z 2020-04-11T05:09:27Z  38.61153 -121.51080 38.61153
##      End_Lng Distance.mi.
## 1  -97.72503        0.040
## 2 -122.54512        0.307
## 3  -80.21219        0.157
## 4  -82.05515        0.110
## 5 -118.42264        0.042
## 6 -121.51080        0.000
##                                                                         Descri
## 1                                         Incident on E 45TH ST near AVENUE H Drive with cau
## 2                                            Incident on US-101 NB near CA-37 Drive with cau
```

```
## 3                        Ramp closed to I-95 and I-95 Northbound Express Ln - Road closed due to acci
## 4 Stationary traffic on SC-40 from Mitchell Rd (New Cut Rd) to John Dodd Rd (New Cut Rd) due to acci
## 5                                                      Incident on I-84 EB near MP 238 Drive with cau
## 6                                      At Garden Hwy/Exit 521A/Exit 521 - Accident. Hard shoulder blo
##          Street Side       City      County State    Zipcode Country
## 1       Avenue H    R     Austin      Travis    TX 78751-3122      US
## 2  Redwood Hwy S    R     Novato       Marin    CA      94945      US
## 3 Airport Expy E    R      Miami  Miami-Dade    FL      33127      US
## 4     New Cut Rd    R      Inman Spartanburg    SC 29349-4532      US
## 5         I-84 E    R  Pendleton    Umatilla    OR      97801      US
## 6        CA-16 E    R Sacramento  Sacramento    CA      95833      US
##     Timezone Airport_Code    Weather_Timestamp Temperature.F. Wind_Chill.F.
## 1 US/Central         KATT 2021-01-06T13:51:00Z             65            65
## 2 US/Pacific         KDVO 2021-04-03T11:15:00Z             54            54
## 3 US/Eastern         KMIA 2019-05-09T13:53:00Z             87            87
## 4 US/Eastern         KSPA 2021-11-15T12:15:00Z             55            55
## 5 US/Pacific         KPDT 2021-12-14T10:53:00Z             38            30
## 6 US/Pacific         KMCC 2020-04-11T04:50:00Z             46            44
##   Humidity... Pressure.in. Visibility.mi. Wind_Direction Wind_Speed.mph.
## 1          50        29.17             10             NW              10
## 2          67        30.10             10            WNW               9
## 3          61        29.96             10             SE              13
## 4          38        29.29             10              W               3
## 5          60        28.20             10             SW              12
## 6          93        29.88             10            SSE               5
##   Weather_Condition Amenity  Bump Crossing Give_Way Junction No_Exit Railway
## 1     Partly Cloudy   FALSE FALSE     TRUE    FALSE    FALSE   FALSE   FALSE
## 2            Cloudy   FALSE FALSE    FALSE    FALSE     TRUE   FALSE   FALSE
## 3     Mostly Cloudy   FALSE FALSE    FALSE    FALSE    FALSE   FALSE   FALSE
## 4              Fair   FALSE FALSE    FALSE    FALSE    FALSE   FALSE   FALSE
## 5              Fair   FALSE FALSE    FALSE    FALSE    FALSE   FALSE   FALSE
## 6              Fair   FALSE FALSE    FALSE    FALSE    FALSE   FALSE   FALSE
##   Roundabout Station  Stop Traffic_Calming Traffic_Signal Turning_Loop
## 1      FALSE   FALSE FALSE           FALSE          FALSE        FALSE
## 2      FALSE   FALSE FALSE           FALSE          FALSE        FALSE
## 3      FALSE   FALSE FALSE           FALSE          FALSE        FALSE
## 4      FALSE   FALSE FALSE           FALSE          FALSE        FALSE
## 5      FALSE   FALSE FALSE           FALSE          FALSE        FALSE
## 6      FALSE   FALSE FALSE           FALSE          FALSE        FALSE
##   Sunrise_Sunset Civil_Twilight Nautical_Twilight Astronomical_Twilight
## 1            Day            Day               Day                   Day
## 2            Day            Day               Day                   Day
## 3            Day            Day               Day                   Day
## 4            Day            Day               Day                   Day
## 5            Day            Day               Day                   Day
## 6          Night          Night             Night                 Night
```

```
dim(comb.df)
```

```
## [1] 50000    43
```

```
numcomb.df <- comb.df[,c(3,4,5,6,7,19,20,21,22,23,25)]
colSums(is.na(numcomb.df))
```

```
##       Start_Lat       Start_Lng         End_Lat         End_Lng    Distance.mi.
##               0               0               0               0               0
##   Temperature.F.   Wind_Chill.F.     Humidity...    Pressure.in.  Visibility.mi.
##            1161            8151            1220             983            1180
## Wind_Speed.mph.
##            2713
```

```r
median1 <- apply(numcomb.df, 2, median, na.rm = TRUE)

dim(numcomb.df)
```

```
## [1] 50000    11
```

```r
# Imputing the corresponding median values, I have that

# Imputing Mean
# for (i in 1:dim(numericaltest)[2]){
#   for (j in 1:dim(numericaltest)[1]){
#     if(is.na(numericaltest[j,i]) == TRUE){
#       numericaltest[j,i] <- as.numeric(mean(na.omit(numericaltest[,i])))
#     }
#   }
# }


# # Imputing median
for (i in 1:dim(numcomb.df)[2]){
  for (j in 1:dim(numcomb.df)[1]){
    if(is.na(numcomb.df[j,i]) == TRUE){
      numcomb.df[j,i] <- as.numeric(median1[i])
    }
  }
}


# I can see that the amount of NA's in this data frame is now zero
colSums(is.na(numcomb.df))
```

```
##       Start_Lat       Start_Lng         End_Lat         End_Lng    Distance.mi.
##               0               0               0               0               0
##   Temperature.F.   Wind_Chill.F.     Humidity...    Pressure.in.  Visibility.mi.
##               0               0               0               0               0
## Wind_Speed.mph.
##               0
```

```r
# Achieve the numerical predictors from the training and testing data
head(numcomb.df)
```

```
##   Start_Lat  Start_Lng End_Lat    End_Lng Distance.mi. Temperature.F.
## 1  30.30729  -97.72562 30.30702  -97.72503        0.040             65
## 2  38.07553 -122.54181 38.07913 -122.54512        0.307             54
## 3  25.81245  -80.21472 25.81242  -80.21219        0.157             87
```

```
## 4  34.99765  -82.05707 34.99737  -82.05515        0.110             55
## 5  45.50310 -118.42311 45.50258 -118.42264        0.042             38
## 6  38.61153 -121.51080 38.61153 -121.51080        0.000             46
##   Wind_Chill.F. Humidity... Pressure.in. Visibility.mi. Wind_Speed.mph.
## 1            65          50        29.17             10              10
## 2            54          67        30.10             10               9
## 3            87          61        29.96             10              13
## 4            55          38        29.29             10               3
## 5            30          60        28.20             10              12
## 6            44          93        29.88             10               5
```

```r
numericaltrain <- numcomb.df[1:35000,]
numericaltest <- numcomb.df[35001:50000,]
train.x <- train[,1]

library(class)
# take the square root of the number of predictors to find the optimal value of k.
# Perform a knn test with k = 3 and k = 4
knn.model <- knn(numericaltrain, numericaltest,cl = train.x, k = 3)
table(knn.model)
```

```
## knn.model
##   MILD SEVERE
##  14217    783
```

```r
# write.csv(knn.model,"knn.model.3.csv")
knn.model <- knn(numericaltrain, numericaltest,cl = train.x, k = 4)
table(knn.model)
```

```
## knn.model
##   MILD SEVERE
##  14241    759
```

```r
# write.csv(knn.model,"knn.model.3.csv")
```

I have found that the knn model does not product the best results, so I will look into alternative methods.

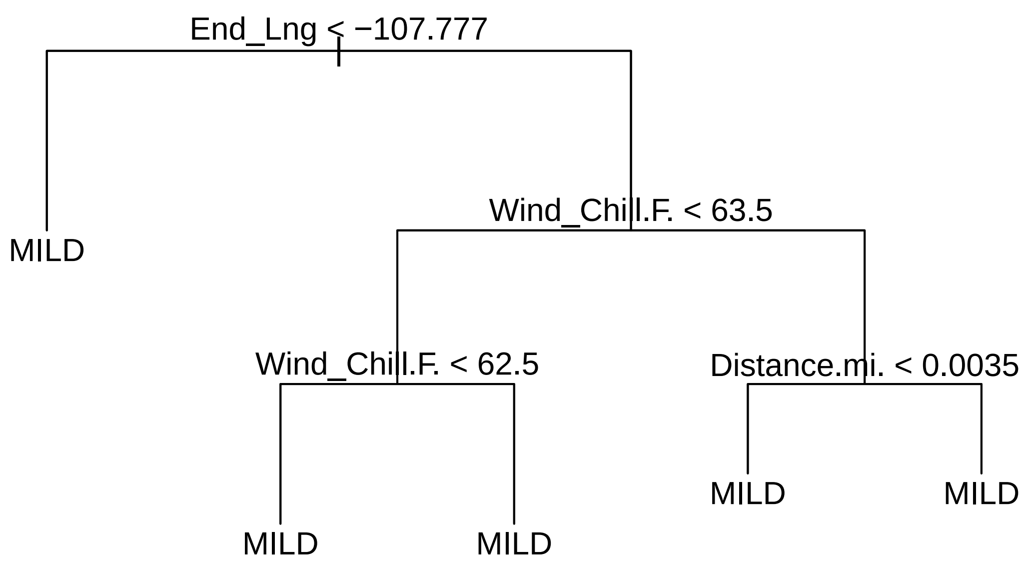
Decision Tree

```r
# I will construct the data frame that will be utilized for the decision tree
Severity <- train.x
traintemp <- cbind(Severity, numericaltrain)

# Construction of the model
train_tree <- tree(factor(Severity) ~ ., data = traintemp)
summary(train_tree)
```

```
##
## Classification tree:
## tree(formula = factor(Severity) ~ ., data = traintemp)
```

```
## Variables actually used in tree construction:
## [1] "End_Lng"       "Wind_Chill.F." "Distance.mi."
## Number of terminal nodes:  5
## Residual mean deviance:  0.5841 = 20440 / 35000
## Misclassification error rate: 0.1005 = 3518 / 35000
```

```r
# Plot and label the tree
plot(train_tree)
text(train_tree, pretty = 0)
```




```r
# Resplit the training data and assess the misclassification rate for training data
Severity <- Severity[sam]
traintemp <- cbind(Severity, numericaltrain[sam,])

train_tree <- tree(factor(Severity) ~ ., data = traintemp)
temp <-  predict(train_tree, newdata = test_sam_x, type = "class")
length(test_sam_y)
```

```
## [1] 5000
```

```r
length(temp)
```

```
## [1] 5000
```

```
table(temp, test_sam_y)
```

```
##          test_sam_y
## temp      MILD SEVERE
##   MILD    4489    511
##   SEVERE     0      0
```

```
mean(temp != test_sam_y)
```

```
## [1] 0.1022
```

Pruning the tree

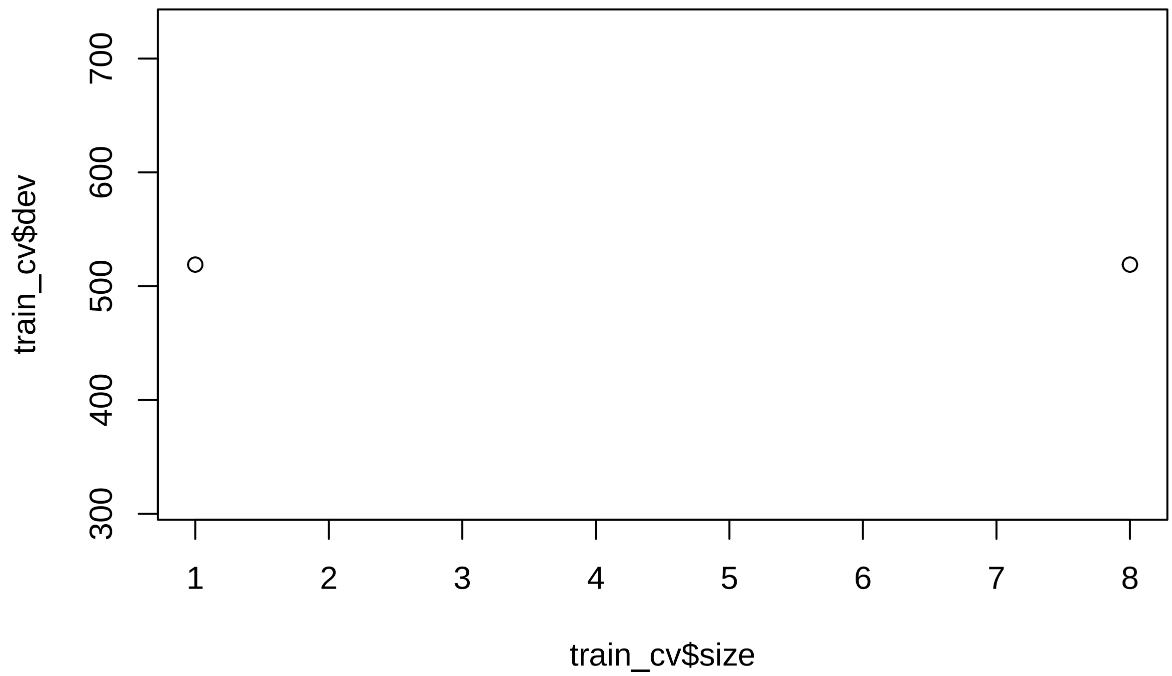
```
# As I see from the decision tree above, everything will be classified as MILD. This is not a very good
train_cv = cv.tree(train_tree, FUN = prune.misclass)
names(train_cv)
```

```
## [1] "size"   "dev"    "k"      "method"
```

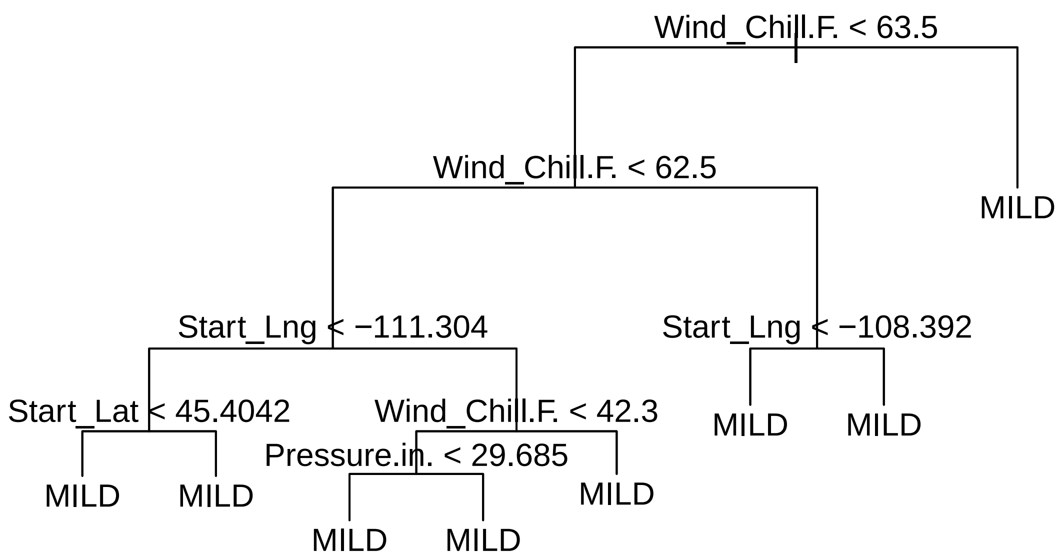
```
summary(train_cv)
```

```
##        Length Class  Mode
## size   2      -none- numeric
## dev    2      -none- numeric
## k      2      -none- numeric
## method 1      -none- character
```

```
plot(train_cv$size, train_cv$dev)
```




```r
train_pruned <- prune.misclass(train_tree, best = 5)
plot(train_pruned)
text(train_pruned, pretty = T)
```



I have pruned the decision tree, but it is still the exactly the same as the initial decision tree. I will look into alternative models.

Split the training data into testing and training to assess accuracy of model prior to submissions

```
set.seed(1128)
sam <- sample(1:35000, 5000, replace = F)
train_sam <- train[-sam,]
test_sam <- train[sam,]
traintemp_sam <- traintemp[-sam,]
testtemp_sam <- traintemp[sam,]
testtemp_sam_x <- testtemp_sam[,-1]

traintest_tree <- tree(formula = factor(Severity)~., data = traintemp_sam)
test_prediction <- predict(traintest_tree, newdata = testtemp_sam, type = "class")

# Building confusion matrix
table((testtemp_sam$Severity), (test_prediction))
```

```
##
##          MILD SEVERE
##   MILD    657      0
##   SEVERE   55      0
```

```
# misclassification rate based on the training set is
mean((testtemp_sam$Severity)!=(test_prediction))
```

```
## [1] NA
```

Uncomment the code chunk below to generate csv file that will be submitted to Kaggle

```
# test_prediction <- predict(train_pruned, data = train_pred, newdata = test_pred, type = "class")
# table(test_prediction)
# write.csv(test_prediction, "tree_model_1.csv")
```

Adding predictors

timediff

I have seen that thus far, I have seen high misclassification rates. I will look into discovering new predictors. Below, I will be looking into generating a time difference predictor. Intuitively, if the duration of the accident is longer, it is more likely that the accident is SEVERE since there could possibly be a greater collision, greater amount of assistance needed, and greater cleaning needed.

```
# To make the time difference predictor

head(train$Start_Time, 5)
```

```
## [1] "2021-01-06T13:58:00Z" "2021-04-03T11:08:00Z" "2019-05-09T14:19:17Z"
## [4] "2021-11-15T12:22:30Z" "2021-12-14T10:24:32Z"
```

```
# Notice that all time stamps are trapped with character T and Z.
start <- strsplit(train$Start_Time, "T|Z")
end <- strsplit(train$End_Time, "T|Z")

for(i in 1:length(end)) {
  end[i] <- paste(end[[i]][1], end[[i]][2], sep = " ")
}
for(i in 1:length(start)) {
  start[i] <- paste(start[[i]][1], start[[i]][2], sep = " ")
}


# Calculate the difference in time through a R function
time_diff <- difftime(unlist(end), unlist(start), unit = "hours")

# Adding to the training model as a new predictor
train$time_diff <- time_diff

# repeat for test data

start <- strsplit(test$Start_Time, "T|Z")
end <- strsplit(test$End_Time, "T|Z")

for(i in 1:length(end)) {
  end[i] <- paste(end[[i]][1], end[[i]][2], sep = " ")
}
for(i in 1:length(start)) {
  start[i] <- paste(start[[i]][1], start[[i]][2], sep = " ")
```

```r
}

time_diff <- difftime(unlist(end), unlist(start), unit = "hours")

# Adding to the testing model as a new predictor
test$time_diff <- time_diff
```

rush_hour

Further, I look into the specific hours that I expect there to be a rush hour. This is mainly when workers leave their work place to go back home. Ultimately, there are more individuals on the road during these hours

```r
# trying to determine which observations are in the rush hours

rush <- c()

# Iterating over each observation, examining the time that the accident occurred
for(i in 1:dim(train)[1]) {
  s <- strsplit(train$Start_Time[i], "T|Z")[[1]]
  seven_pm <- paste(s[1], "17:00:00", sep = " ")
  # four_pm <- paste(s[1], "14:00:00", sep = " ")
  start <- substring(gsub("T|Z", " ", train$Start_Time[i]), 1, nchar(seven_pm))
  before_7 <- difftime(seven_pm, start, unit = "hours")
  # after_4 <- difftime(start, four_pm, unit = "hours")
  if(0 <= before_7 && 3 >= before_7) {
    rush[i] <- TRUE
  } else {
    rush[i] <- FALSE
  }
}

head(rush, 15)
```

```
##  [1] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE  TRUE  TRUE
```

```r
# head(train)

# Add the generated vector as a predictor
train$rush_hour <- rush


# repeat for testing data

rush <- c()
for(i in 1:dim(test)[1]) {
  s <- strsplit(test$Start_Time[i], "T|Z")[[1]]
  seven_pm <- paste(s[1], "17:00:00", sep = " ")
  # four_pm <- paste(s[1], "14:00:00", sep = " ")
  start <- substring(gsub("T|Z", " ", test$Start_Time[i]), 1, nchar(seven_pm))
  before_7 <- difftime(seven_pm, start, unit = "hours")
```

```
  # after_4 <- difftime(start, four_pm, unit = "hours")
  if(0 <= before_7 && 3 >= before_7) {
    rush[i] <- TRUE
  } else {
    rush[i] <- FALSE
  }
}

test$rush_hour <- rush
# head(test)
```

precovid, covid, postcovid

The dataset consists of observations from various years. Mainly in 2019 and 2020, the COVID-19 pandemic has drastically changed the society. More individuals stayed at home to finish their tasks which indicates that less individuals were on the road. I aim to understand when the observation had occurred pre-covid, during covid, or post-covid.

```
# manipulate training data
train$Date <- as.Date(train$Start_Time)
train$precovid <- ifelse(train$Date < as.Date("2020-04-01"), 1, 0)
train$covid <- ifelse(train$Date > as.Date("2020-04-01") & train$Date < as.Date("2021-01-01"), 1, 0)
train$postcovid <- ifelse(train$Date > as.Date("2021-01-01"), 1, 0)

# do same thing to testing

test$Date <- as.Date(test$Start_Time)
test$precovid <- ifelse(test$Date < as.Date("2020-04-01"), 1, 0)
test$covid <- ifelse(test$Date > as.Date("2020-04-01") & test$Date < as.Date("2021-01-01"), 1, 0)
test$postcovid <- ifelse(test$Date > as.Date("2021-01-01"), 1, 0)
```

Description RegEx

In the dataset, there is a column dedicated for the description of the accident. Closely examining the explanations, accidents that are classified as SEVERE tend to have certain vocabularies. Ultimately, I want to assess which observations has these key words.

```
# Key words in SEVERE accidents are Road, alternate, caution, closed, blocked, incident

# Make a column for each of these words
train <- train %>%
  mutate(road = as.numeric(str_detect(Description, "(?i)Road")),
         altrt = as.numeric(str_detect(Description, "(?i)alternate")),
         caution = as.numeric(str_detect(Description, "(?i)caution")),
         closed = as.numeric(str_detect(Description, "(?i)closed")),
         blocked = as.numeric(str_detect(Description, "(?i)blocked")),
         incident = as.numeric(str_detect(Description, "(?i)incident")))
# head(train)

# Perform the same for the testing data
test <- test %>%
  mutate(road = as.numeric(str_detect(Description, "(?i)Road")),
```

```
         altrt = as.numeric(str_detect(Description, "(?i)alternate")),
         caution = as.numeric(str_detect(Description, "(?i)caution")),
         closed = as.numeric(str_detect(Description, "(?i)closed")),
         blocked = as.numeric(str_detect(Description, "(?i)blocked")),
         incident = as.numeric(str_detect(Description, "(?i)incident")))
# head(test)
```

Performing randomForest using RegEx, Covid date, and rush hour that was generated thus far

```
# take all the significant predictors that we generated, exclude time and insignificant predictors
train_mod <- train[, c(1, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56)]
set.seed(1128)
sam <- sample(1:35000, 5000, replace = F)
test_sample <- train_mod[sam,]
train_sample <- train_mod[-sam,]

# take all the significant predictors that we generated, exclude time and insignificant predictors
test_mod <- test[, c(44,45, 47, 48, 49, 50, 51, 52, 53, 54, 55)]


# Performing random forest
training_tree <- randomForest(as.factor(Severity) ~ ., data = train_mod, mtry = 6, importance = T)

training_tree_pred_tr <- predict(training_tree, data = train_sample, newdata = test_sample)
# training_tree_pred_tr

# confusion matrix
table(test_sample$Severity, training_tree_pred_tr)
```

```
##         training_tree_pred_tr
##          MILD SEVERE
##   MILD   4486      9
##   SEVERE  316    189
```

```
# misclassification rate
mean(test_sample$Severity != training_tree_pred_tr)
```

```
## [1] 0.065
```

Uncomment below when generating the Severity for the actual testing data

```
# testing_tree_pred <- predict(training_tree, data = train_mod, newdata = test_mod)
# write.csv(testing_tree_pred, "randForest_rush.csv")
```

Zipcode

Accidents may tend to be more SEVERE in certain regrions and towns. To assess the specific cities, I will look into the Zipcode.

```r
# Which zipcode had the most amount of SEVERE accidents
head(sort(table(train[train$Severity == "SEVERE",]$Zipcode), decreasing = T),15)
```

```
##
## 75243 80229 33168 75228 80216 80401 37210 60607 20020 33127 33150 60621 77092
##    11    10     8     8     8     8     7     7     6     6     6     6     6
## 80104 80118
##     6     6
```

```r
# Below are the most common zipcodes that had severe accidents

top_zip <- c(75243, 80229, 33168, 75228, 80216, 80401, 37210, 60607, 20020, 33127, 33150, 60621, 77092,
train <- train %>%
   mutate(
    newzip = str_detect(Zipcode, "75243|80229|33168|75228|80216|80401|37210|60607|20020|33127|33150|6062
  )

test <- test %>%
   mutate(
    newzip = str_detect(Zipcode, "75243|80229|33168|75228|80216|80401|37210|60607|20020|33127|33150|6062
  )

# generating a new predictor
train_mod$newzip <- train$newzip
test_mod$newzip <- test$newzip
```

Further, from external sources, the zipcodes with the most frequent SEVERE accidents is extracted.

```r
# According to [carinsurance.com](https://www.carinsurance.com/Articles/car-insurance-rate-comparison.a

# 48210|11212|70145|63107|33603|91205|19801|19142|89106|2119

train <- train %>%
  mutate(
    online_zip = str_detect(Zipcode, "48210|11212|70145|63107|33603|91205|19801|19142|89106|2119"),
  )
test <- test %>%
  mutate(
    online_zip = str_detect(Zipcode, "48210|11212|70145|63107|33603|91205|19801|19142|89106|2119"),
  )

# generating new predictors
train_mod$online_zip <- train$online_zip
test_mod$online_zip <- test$online_zip
```

Holidays

Many individuals go outside to have the time of their life during notable holidays. As a result, many do not look into consequences and drink-and-drive. Assess the major holidays and extract accidents that occur during these days.

Add in Holidays (Memorial day, mothers and fathers day, and fourth of July) Our dates are from 2016 to 2021, so we need to find the days of memorial day for each year as the date changes

2016: 2016-05-30 2017: 2017-05-29 2018: 2018-05-28 2019: 2019-05-27 2020: 2020-05-25 2021: 2021-05-31

Fourth of July Just July Fourth of each year

Mothers day 2016: 2016-05-08 2017: 2017-05-14 2018: 2018-05-13 2019: 2019-05-12 2020: 2020-05-10 2021: 2021-05-09

Fathers day 2016: 2016-06-19 2017: 2017-06-18 2018: 2018-06-17 2019: 2019-06-16 2020: 2020-06-21 2021: 2021-06-20

```
train <- train %>%
  mutate(holiday = str_detect(Start_Time,
                              "2016-05-30|2017-05-29|2018-05-28|2019-05-27|2020-05-25|2021-05-31|2016-0
         )
test <- test %>%
  mutate(holiday = str_detect(Start_Time,
                              "2016-05-30|2017-05-29|2018-05-28|2019-05-27|2020-05-25|2021-05-31|2016-0
         )

# generating new predictor
train_mod$holiday <- train$holiday
test_mod$holiday <- test$holiday
```

crash_months

Upon examination, certain months tend to have a higher SEVERE accident rate. This is mainly during the summer and perhaps this nature occurs because of school breaks and as a result, the amount of individuals who travel increases.

```
train$crash_months <- ifelse(str_detect(substring(as.Date(train$Start_Time), 6, 7), "06|07|08|09|10"), 
test$crash_months <- ifelse(str_detect(substring(as.Date(test$Start_Time), 6, 7), "06|07|08|09|10"), 1,

train$online_zip <- ifelse(is.na(train$online_zip), F, train$online_zip)
train$newzip <- ifelse(is.na(train$newzip), F, train$newzip)


# these observations have missing values, imputation of 0 suffices because upon examination, they are n
which(is.na(train_mod$newzip))
```

```
##  [1]  1523  3855  5085 10469 11018 11856 12796 14377 15091 15939 16612 17226
## [13] 20130 20330 21007 22279 25447 32080
```

```
train_mod[c(1523,3855,3855,5085,10469,11018,11856,12796,14377,15091,15939,16612,17226,20130,20330,21007
train_mod[c(1523,3855,3855,5085,10469,11018,11856,12796,14377,15091,15939,16612,17226,20130,20330,21007
```

Upon generating these predictors, as a temporary step, I generated a model again and the accuracy did go up on Kaggle!

```
tree_model <- randomForest(as.factor(Severity) ~ ., data = train_mod, mtry = 7, importance = T)
0.0674 # 6
```

```
## [1] 0.0674
```

```r
0.0668 # 7
```

```
## [1] 0.0668
```

```r
tree_model_predict <- predict(tree_model, newdata = test_mod)
table(tree_model_predict)
```

```
## tree_model_predict
##   MILD SEVERE
##  14390    606
```

```r
# write.csv(tree_model_predict, "randomForest_mtry7.csv")
```

insurance, teendrivers, car crashes fatalities teens

The accuracy thus far is a 0.93235. To increase our testing accuracy, again I look into external sources to see various factors. Three predictors are introducted below. First is insurance which looks into states that have the highest insurance prices. Second is the teen drivers to see which state has the highest proportion of teen drivers. Third is the car crashes fatalities teen which indicates the state with the highest fatalities (SEVERE) amongst teens.

```r
# The following sources were used
# Insurance: [experian.com](https://www.experian.com/blogs/ask-experian/research/most-expensive-states-
# Teen Drivers: [carinsurance.com](https://www.carinsurance.com/Articles/teen-driving-safety-least-and-
# Car Crashes Fatalities Teens: [iihs.org](https://www.iihs.org/topics/fatality-statistics/detail/teena

train <- train %>%
   mutate(
    insurance = str_detect(State, "DE|LO|NY|GA|MD|MI|NJ|FL|RI|SC"),
    teendrivers = str_detect(State, "KY|MS|NC|MO|WV"),
    ccft = str_detect(State, "WY|SD|MS|MO|AL")
  )

# generating the predictors for training
train_mod$insurance <- train$insurance
train_mod$teendrivers <- train$teendrivers
train_mod$ccft <- train$ccft

test <- test %>%
   mutate(
    insurance = str_detect(State, "DE|LO|NY|GA|MD|MI|NJ|FL|RI|SC"),
    teendrivers = str_detect(State, "KY|MS|NC|MO|WV"),
    ccft = str_detect(State, "WY|SD|MS|MO|AL")
  )

# generating the predictors for testing
test_mod$insurance <- test$insurance
test_mod$teendrivers <- test$teendrivers
test_mod$ccft <- test$ccft
```

State Death Rate

Accident mortality rate differs by state. Upon utilizing external sources, a predictor is generated to assess if an accident is inclined to be more SEVERE if it occured at a given state.

```
# How many fatal car crashes per 100,000 people in the state's population?


# Source for State Death Rate: [cdc.gov](https://www.cdc.gov/nchs/pressroom/sosmap/accident_mortality/a

state_death_rate <- c()
for(i in 1:35000) {
  # Let's split the states into three categories of deaths per 100,000 population: high, mid, low
  if(str_detect(train$State[i], "CT|DC|MA|MN|NH|NJ|NY|PA|UT|WA")) {
    state_death_rate[i] <- "Low"
  } else if(str_detect(train$State[i], "AZ|CA|CO|DE|ID|IL|IN|IA|KA|MA|MD|MI|NE|NV|NC|ND|OH|OR|TX|VT|VA|W
    state_death_rate[i] <- "Mid"
  } else {
    state_death_rate[i] <- "High"
  }
}

train$state_death_rate <- state_death_rate
train_mod$sdr <- train$state_death_rate
# repeat for testing data

state_death_rate <- c()
for(i in 1:15000) {
  if(str_detect(test$State[i], "CT|DC|MA|MN|NH|NJ|NY|PA|UT|WA")) {
    state_death_rate[i] <- "Low"
  } else if(str_detect(test$State[i], "AZ|CA|CO|DE|ID|IL|IN|IA|KA|MA|MD|MI|NE|NV|NC|ND|OH|OR|TX|VT|VA|W
    state_death_rate[i] <- "Mid"
  } else {
    state_death_rate[i] <- "High"
  }
}


test$state_death_rate <- state_death_rate
test_mod$sdr <- test$state_death_rate
```

Temperature

At the beginning, I have assessed the significant quantitative predictors. Temperature during the given accident is one of the most significant predictor.

```
# These are the column indeces for temperature and wind chill
temp <- train[,c(20,21)]
temp1 <- test[,c(19,20)]

tempdf <- rbind(temp,temp1)
```

```r
# temporary data cleaning
median1 <- apply(tempdf, 2, median, na.rm = TRUE)
for (i in 1:dim(tempdf)[2]){
  for (j in 1:dim(tempdf)[1]){
    if(is.na(tempdf[j,i]) == TRUE){
      tempdf[j,i] <- as.numeric(median1[i])
    }
  }
}


colSums(is.na(tempdf))
```

```
## Temperature.F.  Wind_Chill.F.
##              0              0
```

```r
temptrain <- tempdf[1:35000,]
temptest <- tempdf[35001:50000,]
```

However, there are many missing or NA values for this predictor.

Below, I perform the missForest imputation to tackle this problem.

```r
# install.packages("missForest")
library(missForest)
```

```
## Warning: package 'missForest' was built under R version 4.1.2
```

```
##
## Attaching package: 'missForest'
```

```
## The following object is masked from 'package:VIM':
##
##     nrmse
```

```r
# changing the qualitative predictors to factor for the missForest imputation
temptemp <- train_mod
# colSums(is.na(temptemp))
temptemp$rush_hour <- as.factor(temptemp$rush_hour)
temptemp$precovid <- as.factor(temptemp$precovid)
temptemp$covid <- as.factor(temptemp$covid)
temptemp$postcovid <- as.factor(temptemp$postcovid)
temptemp$road <- as.factor(temptemp$road)
temptemp$altrt <- as.factor(temptemp$altrt)
temptemp$caution <- as.factor(temptemp$caution)
temptemp$closed <- as.factor(temptemp$closed)
temptemp$blocked <- as.factor(temptemp$blocked)
temptemp$incident <- as.factor(temptemp$incident)
temptemp$newzip <- as.factor(temptemp$newzip)
temptemp$online_zip <- as.factor(temptemp$online_zip)
temptemp$holiday <- as.factor(temptemp$holiday)
```

```r
temptemp$insurance <- as.factor(temptemp$insurance)
temptemp$teendrivers <- as.factor(temptemp$teendrivers)
temptemp$ccft <-  as.factor(temptemp$ccft)
temptemp$sdr <- as.factor(temptemp$sdr)
temptemp$Temperature.F. <- train$Temperature.F.
temptemp1 <- temptemp[,-c(1,2)]
```

Performing the imputation upon changing the class of each predictors

```r
temptemp2 <- missForest(temptemp1)
```

Successful imputation is shown below

```r
temp3 <- temptemp2$ximp
head(temp3)
```

```
##   rush_hour precovid covid postcovid road altrt caution closed blocked incident
## 1     FALSE        0     0         1    0     0       1      0       0        1
## 2     FALSE        0     0         1    0     0       1      0       0        1
## 3      TRUE        1     0         0    1     0       0      1       0        0
## 4     FALSE        0     0         1    0     0       0      0       0        0
## 5     FALSE        0     0         1    0     0       1      0       0        1
## 6     FALSE        0     1         0    0     0       0      0       1        0
##   newzip online_zip holiday insurance teendrivers  ccft  sdr Temperature.F.
## 1      0          0   FALSE     FALSE       FALSE FALSE  Mid             65
## 2      0          0   FALSE     FALSE       FALSE FALSE  Mid             54
## 3      1          0   FALSE      TRUE       FALSE FALSE High             87
## 4      0          0   FALSE      TRUE       FALSE FALSE High             55
## 5      0          0   FALSE     FALSE       FALSE FALSE  Mid             38
## 6      0          0   FALSE     FALSE       FALSE FALSE  Mid             46
```

```r
# generating temperature predictor
train_mod$Temperature.F. <- temp3$Temperature.F.
```

To assess work thus far, the model is implemented

Testing the model on training data which is re-split into training and testing.

```r
set.seed(1128)
sam <- sample(1:35000, 5000, replace = F)
train_sam <- train_mod[-sam,]
test_sam <- train_mod[sam,]

training_tree <- randomForest(as.factor(Severity) ~ ., data = train_sam, mtry = 6, importance = T)
testing_tree_pred <- predict(training_tree, data = train_sam, newdata = test_sam)

# Confusion Matrix
table(testing_tree_pred, test_sam$Severity)
```

```
##
## testing_tree_pred MILD SEVERE
##            MILD   4464    301
##            SEVERE   31    204
```

```r
# Misclassification rate
mean(testing_tree_pred != test_sam$Severity)
```

```
## [1] 0.0664
```

Uncomment below when generating the Severity for the actual testing data

```r
# training_tree <- randomForest(as.factor(Severity) ~ ., data = train_mod, mtry = 6, importance = T)
# summary(training_tree)
# test_mod$Temperature.F. <- temptest$Temperature.F.
# testing_tree_pred <- predict(training_tree, data = train_mod, newdata = test_mod)
```

Writing to our csv file that will be submitted to Kaggle

```r
# write.csv(testing_tree_pred, "randForest_rush8.csv")
```

diff_lat, diff_lng

Similar to the temperature, the latitude and longitude were extremely significant predictors. They will be incorporated as predictors in the form of difference as if both of them are used, multicollinearity can be introduced.

```r
# Start_Lat, End_Lat, Start_Lng, End_Lng
train_mod$diff_lat <- train$Start_Lat - train$End_Lat
train_mod$diff_lng <- train$Start_Lng - train$End_Lng


test_mod$diff_lat <- test$Start_Lat - test$End_Lat
test_mod$diff_lng <- test$Start_Lng - test$End_Lng
```

Final Model Implementation using Logistic Regression

```r
train_mod_sam <- train_mod[-sam,]
test_mod_sam <- train_mod[sam,]

# testing model using samples

lda_train_sam <- glm(as.factor(Severity) ~ ., family = binomial(), data = train_mod_sam)

lda_test_predict_sam <- predict(lda_train_sam, newdata = test_mod_sam, type = "response")
pred_test_sam <- rep("MILD", length(lda_test_predict_sam))
pred_test_sam[lda_test_predict_sam >= 0.5] <- "SEVERE"


table(pred_test_sam, test_mod_sam$Severity)
```

```
##
## pred_test_sam MILD SEVERE
##        MILD   4448    319
##        SEVERE   47    186
```

```r
mean(pred_test_sam != test_mod_sam$Severity)
```

```
## [1] 0.0732
```

Uncomment below when generating the Severity for the actual testing data

```r
# actual testing prediction for submission
# lda_train <- glm(as.factor(Severity) ~ ., family = binomial(), data = train_mod)
# test_mod$Temperature.F. <- numericaltest$Temperature.F.
#
# lda_test <- predict(lda_train, newdata = test_mod, type = "response")
# pred_test <- rep("MILD", length(lda_test))
# pred_test[lda_test >= 0.5] <- "SEVERE"
#
# write.csv(pred_test, "logistic_regression_kaggle.csv")
```

Final Model Implementation using randomForest

Testing the model on training data which is resplit into training and testing.

```r
training_tree <- randomForest(as.factor(Severity) ~ ., data = train_mod_sam, mtry = 6, importance = T)
testing_tree_pred <- predict(training_tree, data = train_mod_sam, newdata = test_mod_sam)

table(testing_tree_pred, test_mod_sam$Severity)
```

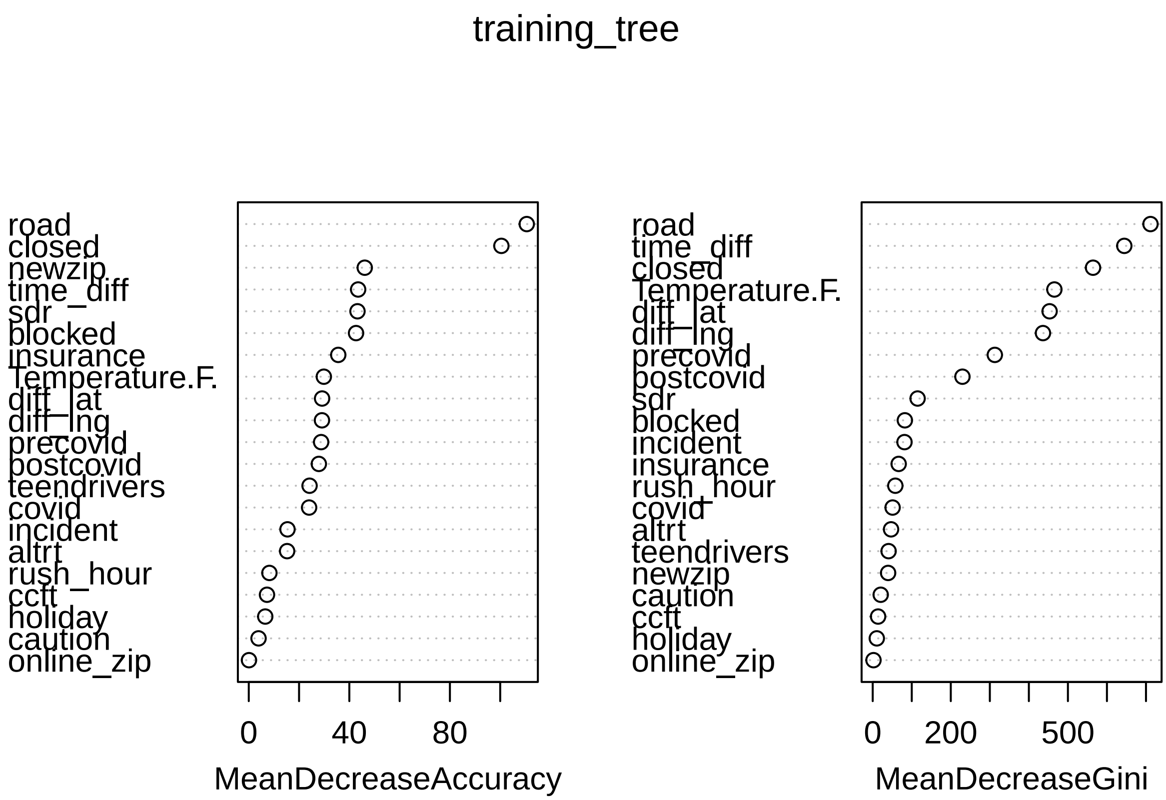
```
##
## testing_tree_pred MILD SEVERE
##            MILD   4463    301
##            SEVERE   32    204
```

```r
mean(testing_tree_pred != test_mod_sam$Severity)
```

```
## [1] 0.0666
```

To look at which predictors are the most significant, we have that

```r
varImpPlot(training_tree)
```



Uncomment below when generating the Severity for the actual testing data

```
# training_tree <- randomForest(as.factor(Severity) ~ ., data = train_mod, mtry = 6, importance = T)
# summary(training_tree)
# testing_tree_pred <- predict(training_tree, data = train_mod, newdata = test_mod)
# write.csv(train_mod, "train_mod.csv")
# write.csv(test_mod, "test_mod.csv")
```

As a result, the constructed model has generated a 0.93688 accuracy! Upon systematic process, the model accuracy has gradually increased.

Below are hidden code chunks that was used during experimentations (failed models and execution, testing)

Attempting to use weather as a predictor - did not generate a good model

```
# test_num <- data.frame(scale(numcomb.df))
# test_pred <- test_num
#
# acc.test$Weather_Condition <- ifelse(is.na(acc.test$Weather_Condition), "Fair", acc.test$Weather_Cond
#
#
# weather <- function(x) {
#   weather_list <- list()
#   for(i in 1:dim(x)[1]) {
```

```
#     if(str_detect(x$Weather_Condition[i], "Clear|Fair|Mostly Cloudy|Drizzle|Light Drizzle|Partly Clou
#       weather_list[i] <- "Light"
#     } else if(str_detect(x$Weather_Condition[i], "Blowing Dust / Windy|Blowing Dust| Cloudy / Windy|D
#       weather_list[i] <- "Mid"
#     } else if(str_detect(x$Weather_Condition[i], "Freezing Rain|Heavy Drizzle|Heavy Rain|Heavy Rain /
#       weather_list[i] <- "Rain"
#     } else {
#       weather_list[i] <- "Heavy"
#     }
#   }
#   return(as.character(weather_list))
# }
#
#
#
# test_pred$weather <- as.factor(weather(comb.df))
```

Failed kNN imputation

```
# Attempting to do kNN impute


# colSums(is.na(test))
# tempcol1 <- as.data.frame(temp[,1])
# tempcol2 <- as.data.frame(temp[,2])
#
# tempcol1
#
# library(caret)
#
# preProcValues <- preProcess(tempcol2,
#                             method = "knnImpute",
#                             k = 20,
#                             knnSummary = mean)
# preProcValues
#
#
# idx <- apply(tempcol2, 1, function(x) sum(is.na(x)))
# length(as.vector(which(idx == ncol(tempcol2))))
#
#
# tempinfo <- predict(preProcValues, tempcol1)
# tempinfo
```

Attempted to use Astronomical_Twilight as a predictor - did not generate best model

```
# table(train_mod$Astronomical_Twilight)
#
# train_mod$Astronomical_Twilight <- ifelse(is.na(train_mod$Astronomical_Twilight), "Day", "Night")
# test_mod$Astronomical_Twilight <- ifelse(is.na(test_mod$Astronomical_Twilight), "Day", "Night")
#
# test_mod$weather <- weather(test)
#
```

```
#
#
# train_mod <- train[, c(1, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53)]
# head(train_mod)
#
#
# test_mod <- test[, c(44, 45, 46, 47, 48, 49, 50, 51, 52, 53)]
#
# tree_train <- tree(as.factor(Severity) ~., data = train_mod)
#
# plot(tree_train)
# text(tree_train, pretty = 0)
#
# tree_train_1 <- predict(tree_train, data = train_mod, newdata = test_mod, type = "class")
#
# table(tree_train_1, test_sam$Severity)
# mean(tree_train_1 != test_sam$Severity)
#
# length(tree_train_1)
#
#
# test_sam <- train_mod[sam, ]
# train_sam <- train_mod[-sam, ]
#
# write.csv(tree_train_1, "tree_model_time_difference.csv")
```

Attempted to detect "Road closed" - predictor already exist, introduces multicollinearity

```
# t <- train_sam[, c(1, 47, 48, 49, 50, 51, 52)]
# t_test <- test_sam[, c(1, 47, 48, 49, 50, 51, 52)]
#
# tree_t <- tree(as.factor(Severity) ~ ., data = t)
#
# head(test)
#
# plot(tree_t)
# text(tree_t, pretty = 0)
#
# t_predict <- predict(tree_t, data = t, newdata = test, type = "class")
# table(t_predict, t_test$Severity)
# mean(t_predict != t_test$Severity)
#
# train[train$Severity == "SEVERE",c(1, 9)]
# train[train$Severity == "MILD",c(1, 9)]
# str_detect(train[16,9], "(?i)Road closed")
#
#
# write.csv(t_predict, "tree_model_road_closed1.csv")
#
#
# length(t_predict)
#
# # incident is an indicator for mild
```

```
# # use closed as an indicator for severe instead of road closed
```

Random Forest experimentation

```
# rf_train <- randomForest(as.factor(Severity) ~ ., data = train_mod, mtry = 3, importance = T)
#
# head(train_mod)
# train_mod$Astronomical_Twilight <- as.factor(train_mod$Astronomical_Twilight)
# train_mod$weather <- as.factor(train_mod$weather)
# train_mod$rdclosed <- as.factor(train_mod$rdclosed)
# train_mod$altrt <- as.factor(train_mod$altrt)
# train_mod$caution <- as.factor(train_mod$caution)
# train_mod$road <- as.factor(train_mod$road)
# train_mod$closed <- as.factor(train_mod$closed)
# train_mod$blocked <- as.factor(train_mod$blocked)
# train_mod$incident <- as.factor(train_mod$incident)
#
# head(test_mod)
# test_mod$Astronomical_Twilight <- as.factor(test_mod$Astronomical_Twilight)
# test_mod$weather <- as.factor(test_mod$weather)
# test_mod$rdclosed <- as.factor(test_mod$rdclosed)
# test_mod$altrt <- as.factor(test_mod$altrt)
# test_mod$caution <- as.factor(test_mod$caution)
# test_mod$road <- as.factor(test_mod$road)
# test_mod$closed <- as.factor(test_mod$closed)
# test_mod$blocked <- as.factor(test_mod$blocked)
# test_mod$incident <- as.factor(test_mod$incident)
#
# test_mod$Astronomical_Twilight <- ifelse(is.na(test_mod$Astronomical_Twilight), "Day", test_mod$Astro
#
# sum(is.na(test_mod))
#
# d <- c()
# for(i in 1:35000) {
#   if(sum(is.na(train_mod[i,] > 0))) {
#     d <- c(d, i)
#   }
# }
# d
# train_mod[d[1:20],]
#
# for(i in 1:35000) {
#   if(is.na(train_mod[i,]$Astronomical_Twilight)) {
#     train_mod[i,]$Astronomical_Twilight <- "Day"
#   }
# }
# summary(rf_train)
# rf_train_prediction <- predict(rf_train, data = train_mod, newdata = test_mod)
# length(rf_train_prediction)
#
#
# rf_train_sam <- predict(rf_train, data = train_sam, newdata = test_sam)
# table(rf_train_sam, test_sam$Severity)
```

```
# mean(rf_train_sam != test_sam$Severity)
#
# write.csv(rf_train_prediction, "rand.forest1.csv")
```

XGBoost - Did not run for this model

```
# library(gbm)
# head(train_mod)
# train_mod$time_diff <- as.numeric(train_mod$time_diff)
# train_mod$Severity <- ifelse(train_mod$Severity == "SEVERE", 0, 1)
# table(train_mod$Severity)
# boost_model <- gbm(as.factor(Severity) ~ ., data = train_mod, distribution = "bernoulli", n.trees = 5
# summary(boost_model)
# boost_train <- predict(boost_model, n.trees = 5000, newdata = test_mod)
# write.csv(boost_train, "rand.forest2.csv")
```

Experiment utilizing time

```
# experiment

# train_severe <- train[train$Severity == "SEVERE",]
# train_mild <- train[train$Severity == "MILD",]
# head(train_severe[,c(1, 10)], 20)
# head(train_mild[,c(1, 10)], 20)
# head(train_severe, 30)
# # indexes are 1, 52, 51, 50, 49, 48, 47, 46, 45, 44
#
#
# start <- strsplit(train_severe$Start_Time[1], "T|Z")[[1]]
# end <- strsplit(train_severe$End_Time[1], "T|Z")[[1]]
# start <- paste(start[1], start[2], sep = " ")
# start
# end <- paste(end[1], end[2], sep = " ")
# end
#
# end <- train_severe$End_Time[1]
# start <- train_severe$Start_Time[1]
#
# time <- strptime(start, format = "%Y-%m-%d %H:%M:%S")
#
# difftime(end, start, unit = "hours")
#
# difftime(train_severe$End_Time[1], train_severe$Start_Time[1], unit = "min")
# start <- strsplit(train_severe$Start_Time, "T|Z")
# end <- strsplit(train_severe$End_Time, "T|Z")
#
# end[[1]][1]
#
# difftime(end, start, unit = "hours")
# for(i in 1:length(end)) {
#   end[i] <- paste(end[[i]][1], end[[i]][2], sep = " ")
# }
# for(i in 1:length(start)) {
```

```
#   start[i] <- paste(start[[i]][1], start[[i]][2], sep = " ")
# }
#
# unlist(end)
#
# difftime(unlist(end), unlist(start), unit = "hours")
# end[[6]]
# start[[6]]
#
# train$time_diff <- difftime(unlist(end), unlist(start), unit = "hours")
#
# length(difftime(unlist(end), unlist(start), unit = "hours"))
#
# head(train_severe, 10)
```