Machine Learning R Notebook

Code

This is an R Markdown (http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

Hide

```
# check current working directory
getwd()
```

```
[1] "C:/Users/ajaka/OneDrive - Teesside University/CIS4035 - Machine Learning/E4216209_AJAKAY
E_JESUBUKADE"
```

Hide

```
# list the files in the working directory
list.files(getwd())
```

```
 [1] "caret_knn_base.rds"                    "caret_knn_tune.rds"
 [3] "caret_lr_base.rds"                     "caret_lr_tune.rds"
 [5] "caret_nb_base.rds"                     "caret_nb_tune.rds"
 [7] "caret_rf_base.rds"                     "caret_rf_tune.rds"
 [9] "E4216209_AJAKAYE_JESUBUKADE.nb.html"   "E4216209_AJAKAYE_JESUBUKADE.Rmd"
[11] "Machine Learning R Notebook.pdf"       "US_Accidents_March23_sampled_500k.csv"
```

Importing Required Libraries and Reading Dataset

Hide

```
# Loading library
library(tidyverse) # for data processing
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✓ dplyr     1.1.4     ✓ readr     2.1.5
✓ forcats   1.0.0     ✓ stringr   1.5.1
✓ ggplot2   3.5.1     ✓ tibble    3.2.1
✓ lubridate 1.9.4     ✓ tidyr     1.3.1
✓ purrr     1.0.2     ── Conflicts ────────────────────────────────────────── tidyvers
e_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts t
o become errors
```

Hide

```
library(gridExtra) # merging visuals
```

```
Attaching package: ‘gridExtra’

The following object is masked from ‘package:dplyr’:

    combine
```

Hide

```
library(ROSE) # Data Balancing
```

```
Loaded ROSE 0.0-4
```

Hide

```
library(caret) # Model Building (createDataPartition)
```

```
Loading required package: lattice

Attaching package: ‘caret’

The following object is masked from ‘package:purrr’:

    lift
```

Hide

```
library(pROC) # ROC and AUC
```

```
Type 'citation("pROC")' for a citation.

Attaching package: ‘pROC’

The following objects are masked from ‘package:stats’:

    cov, smooth, var
```

Hide

```
library(MLmetrics) # F1 score
```

```
Warning: package ‘MLmetrics’ was built under R version 4.4.3
Attaching package: ‘MLmetrics’

The following objects are masked from ‘package:caret’:

    MAE, RMSE

The following object is masked from ‘package:base’:

    Recall
```

Hide

```
# Read the csv file
US_data <- read.csv("US_Accidents_March23_sampled_500k.csv", na.strings=c("","NA"))
head(US_data)
```

	ID <chr>	Source <chr>	Severity <int>	Start_Time <chr>	End_Time <chr>
1	A-2047758	Source2	2	2019-06-12 10:10:56	2019-06-12 10:55:58
2	A-4694324	Source1	2	2022-12-03 23:37:14.000000000	2022-12-04 01:56:53.0000000
3	A-5006183	Source1	2	2022-08-20 13:13:00.000000000	2022-08-20 15:22:45.0000000
4	A-4237356	Source1	2	2022-02-21 17:43:04	2022-02-21 19:43:23
5	A-6690583	Source1	2	2020-12-04 01:46:00	2020-12-04 04:13:09
6	A-1101469	Source2	2	2021-03-29 07:03:58	2021-03-29 08:51:01

6 rows | 1-6 of 46 columns

Data Preparation and Data Cleaning

Hide

```
# Get the column names of data
colnames(US_data)
```

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

Hide

```
# Checking Data attributes
str(US_data)
```

```
'data.frame':   500000 obs. of  46 variables:
 $ ID                  : chr  "A-2047758" "A-4694324" "A-5006183" "A-4237356" ...
 $ Source              : chr  "Source2" "Source1" "Source1" "Source1" ...
 $ Severity            : int  2 2 2 2 2 2 2 2 2 2 ...
 $ Start_Time          : chr  "2019-06-12 10:10:56" "2022-12-03 23:37:14.000000000" "2022-08
-20 13:13:00.000000000" "2022-02-21 17:43:04" ...
 $ End_Time            : chr  "2019-06-12 10:55:58" "2022-12-04 01:56:53.000000000" "2022-08
-20 15:22:45.000000000" "2022-02-21 19:43:23" ...
 $ Start_Lat           : num  30.6 39 34.7 43.7 35.4 ...
 $ Start_Lng           : num  -91.2 -77.4 -120.5 -93 -119 ...
 $ End_Lat             : num  NA 39 34.7 43.7 35.4 ...
 $ End_Lng             : num  NA -77.4 -120.5 -93 -119 ...
 $ Distance.mi.        : num  0 0.056 0.022 1.054 0.046 ...
 $ Description         : chr  "Accident on LA-19 Baker-Zachary Hwy at Lower Zachary Rd." "In
cident on FOREST RIDGE DR near PEPPERIDGE PL Drive with caution." "Accident on W Central Ave
from Floradale Ave to Western Ave." "Incident on I-90 EB near REST AREA Drive with caution."
...
 $ Street              : chr  "Highway 19" " Forest Ridge Dr" "Floradale Ave" "14th St NW"
...
 $ City                : chr  "Zachary" "Sterling" "Lompoc" "Austin" ...
 $ County              : chr  "East Baton Rouge" "Loudoun" "Santa Barbara" "Mower" ...
 $ State               : chr  "LA" "VA" "CA" "MN" ...
 $ Zipcode             : chr  "70791-4610" "20164-2813" "93436" "55912" ...
 $ Country             : chr  "US" "US" "US" "US" ...
 $ Timezone            : chr  "US/Central" "US/Eastern" "US/Pacific" "US/Central" ...
 $ Airport_Code        : chr  "KBTR" "KIAD" "KLPC" "KAUM" ...
 $ Weather_Timestamp   : chr  "2019-06-12 09:53:00" "2022-12-03 23:52:00" "2022-08-20 12:56:
00" "2022-02-21 17:35:00" ...
 $ Temperature.F.      : num  77 45 68 27 42 42 35 90 91 63 ...
 $ Wind_Chill.F.       : num  77 43 68 15 42 35 35 90 91 63 ...
 $ Humidity...         : num  62 48 73 86 34 58 89 55 39 78 ...
 $ Pressure.in.        : num  29.9 29.9 29.8 28.5 29.8 ...
 $ Visibility.mi.      : num  10 10 10 10 10 10 10 10 10 10 ...
 $ Wind_Direction      : chr  "NW" "W" "W" "ENE" ...
 $ Wind_Speed.mph.     : num  5 5 13 15 0 13 0 12 7 10 ...
 $ Precipitation.in.   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Weather_Condition   : chr  "Fair" "Fair" "Fair" "Wintry Mix" ...
 $ Amenity             : chr  "False" "False" "False" "False" ...
 $ Bump                : chr  "False" "False" "False" "False" ...
 $ Crossing            : chr  "False" "False" "False" "False" ...
 $ Give_Way            : chr  "False" "False" "False" "False" ...
 $ Junction            : chr  "False" "False" "False" "False" ...
 $ No_Exit             : chr  "False" "False" "False" "False" ...
 $ Railway             : chr  "False" "False" "False" "False" ...
 $ Roundabout          : chr  "False" "False" "False" "False" ...
 $ Station             : chr  "False" "False" "False" "False" ...
 $ Stop                : chr  "False" "False" "False" "False" ...
 $ Traffic_Calming     : chr  "False" "False" "False" "False" ...
 $ Traffic_Signal      : chr  "True" "False" "True" "False" ...
 $ Turning_Loop        : chr  "False" "False" "False" "False" ...
 $ Sunrise_Sunset      : chr  "Day" "Night" "Day" "Day" ...
 $ Civil_Twilight      : chr  "Day" "Night" "Day" "Day" ...
 $ Nautical_Twilight   : chr  "Day" "Night" "Day" "Day" ...
 $ Astronomical_Twilight: chr  "Day" "Night" "Day" "Day" ...
```

Hide

```
# Checking for missing values
sum(is.na(US_data))
```

```
[1] 829873
```

Hide

```
# Checking for proportion of missing data
US_pctmiss <- colSums(is.na(US_data))/nrow(US_data)
round(US_pctmiss, 2)
```

```
                 ID               Source             Severity 
               0.00                 0.00                 0.00 
         Start_Time             End_Time            Start_Lat 
               0.00                 0.00                 0.00 
          Start_Lng              End_Lat              End_Lng 
               0.00                 0.44                 0.44 
        Distance.mi.          Description               Street 
               0.00                 0.00                 0.00 
               City               County                State 
               0.00                 0.00                 0.00 
            Zipcode              Country             Timezone 
               0.00                 0.00                 0.00 
       Airport_Code    Weather_Timestamp       Temperature.F. 
               0.00                 0.02                 0.02 
      Wind_Chill.F.          Humidity...         Pressure.in. 
               0.26                 0.02                 0.02 
      Visibility.mi.       Wind_Direction      Wind_Speed.mph. 
               0.02                 0.02                 0.07 
   Precipitation.in.    Weather_Condition              Amenity 
               0.29                 0.02                 0.00 
               Bump             Crossing             Give_Way 
               0.00                 0.00                 0.00 
           Junction              No_Exit              Railway 
               0.00                 0.00                 0.00 
         Roundabout              Station                 Stop 
               0.00                 0.00                 0.00 
     Traffic_Calming       Traffic_Signal         Turning_Loop 
               0.00                 0.00                 0.00 
      Sunrise_Sunset       Civil_Twilight    Nautical_Twilight 
               0.00                 0.00                 0.00 
Astronomical_Twilight 
               0.00 
```

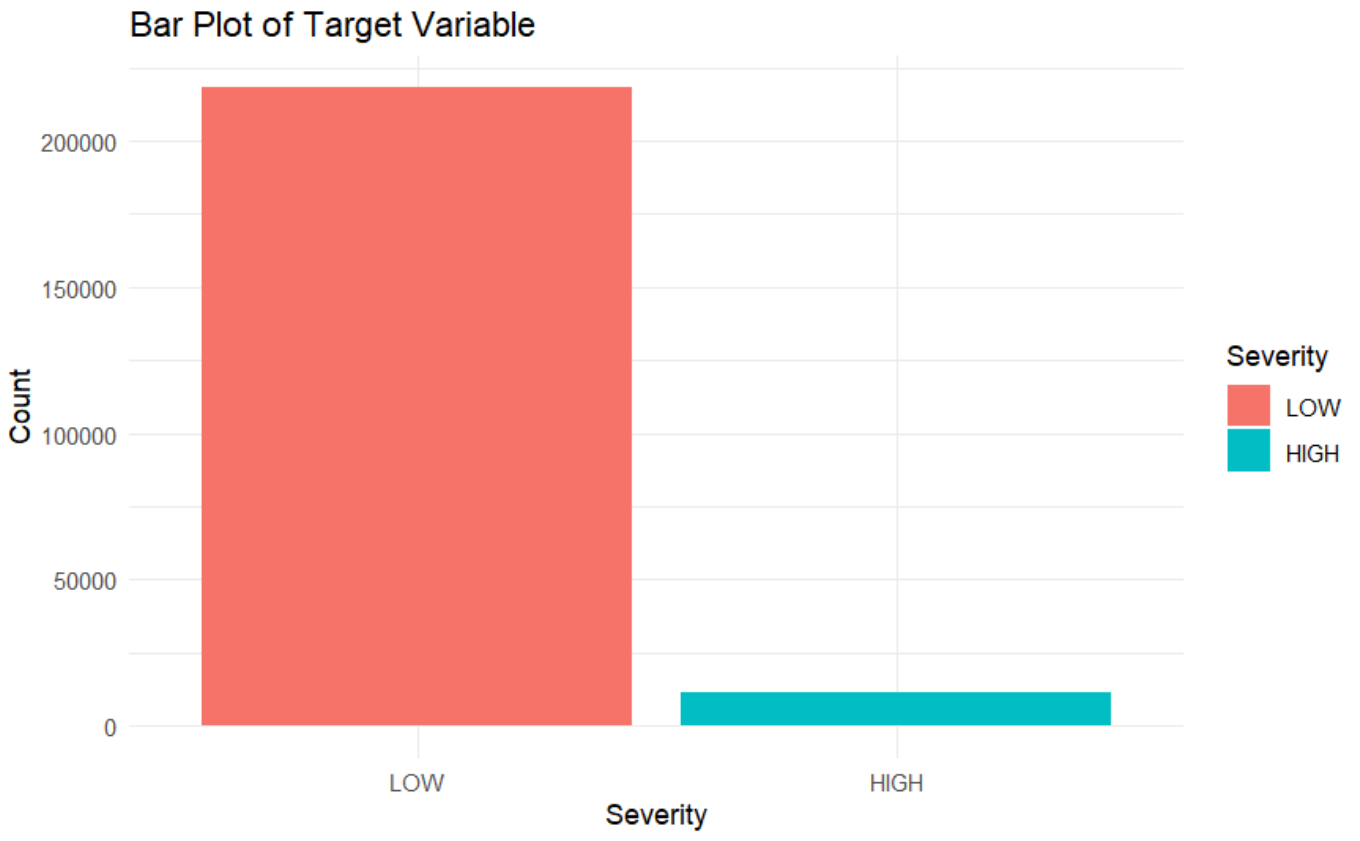
Hide

```
# Dropping All Rows with Missing Data
US_newdata <- na.omit(US_data)

# Re-coding Target variable
US_newdata <- mutate(US_newdata, Severity = ifelse(Severity < 3, 0, 1))

table(US_newdata$Severity)
```

```
     0      1 
218279  11648 
```

Hide

```
# Add a plot of target here
target <- ggplot(US_newdata, aes(x = factor(Severity),
                                 fill = factor(Severity))) +
  geom_bar() +
  labs(title = "Bar Plot of Target Variable",
       x = "Severity",
       y = "Count",
       fill = "Severity") +
  scale_x_discrete(labels  = c("0" = "LOW", "1" = "HIGH")) +
  scale_fill_discrete(labels  = c("0" = "LOW", "1" = "HIGH")) +
  theme_minimal()

target
```



Dealing with imbalance

Hide

```
# Undersampling
set.seed(199)
US_newdata_Sample = ovun.sample(Severity ~ .,
                                data = US_newdata,
                                method = "under",
                                N = 25000)$data

# Checking for missing values
anyNA(US_newdata_Sample)
```

```
[1] FALSE
```

Hide

```
table(US_newdata_Sample$Roundabout)
```

```


False
25000
```

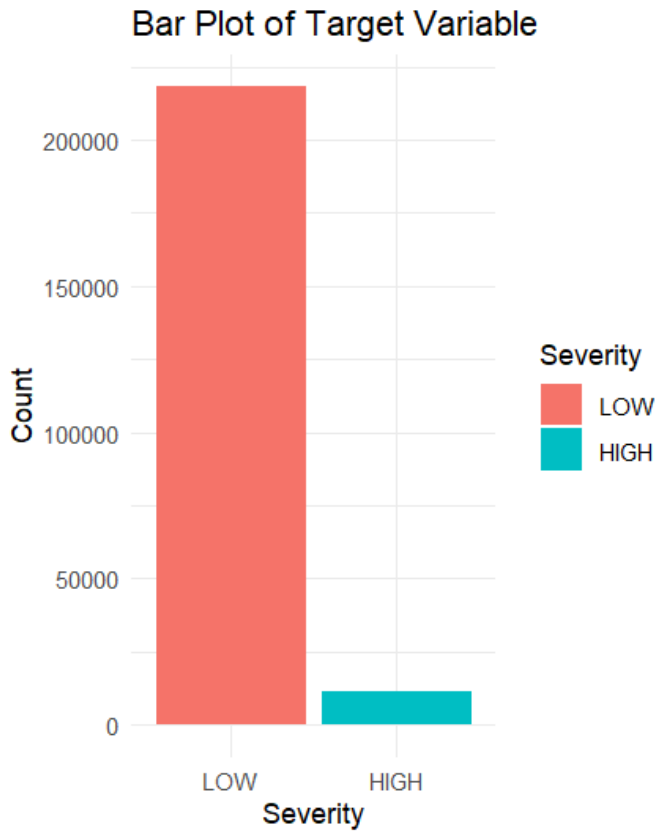
Hide

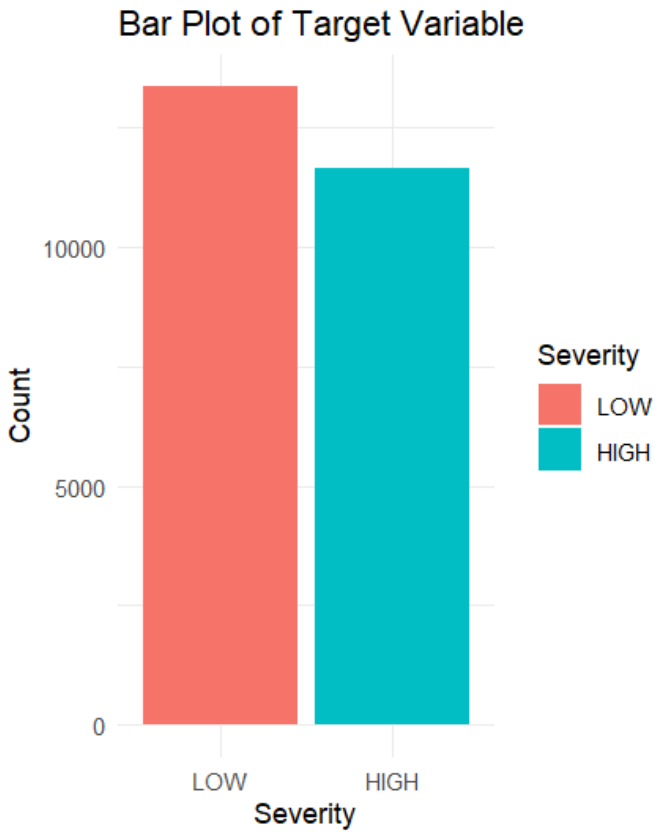
```
table(US_newdata_Sample$Turning_Loop)
```

```


False
25000
```

Hide

```
# Add a plot of new target here
# Add a plot of target here
balances_target <- ggplot(US_newdata_Sample, aes(x = factor(Severity),
                                  fill = factor(Severity))) +
  geom_bar() +
  labs(title = "Bar Plot of Target Variable",
       x = "Severity",
       y = "Count",
       fill = "Severity") +
  scale_x_discrete(labels  = c("0" = "LOW", "1" = "HIGH")) +
  scale_fill_discrete(labels  = c("0" = "LOW", "1" = "HIGH")) +
  theme_minimal()

# Plot Grid of Target Variable
grid.arrange(
  target,
  balances_target,
  ncol = 2
)
```





Data Transformation

Hide

```
# Converting Date Columns to date data
US_newdata_Sample$Start_Time <- parse_date_time(US_newdata_Sample$Start_Time, orders = "ymd_H
MS")

US_newdata_Sample$End_Time <- parse_date_time(US_newdata_Sample$End_Time, orders = "ymd_HMS")

US_newdata_Sample$Weather_Timestamp <- parse_date_time(US_newdata_Sample$Weather_Timestamp, o
rders = "ymd_HMS")

# Creating New Calculated Columns
US_newdata_Sample$Time <- (US_newdata_Sample$End_Time - US_newdata_Sample$Start_Time)

# Converting day difference to numeric
US_newdata_Sample$Time <- as.numeric(US_newdata_Sample$Time)
```

Feature Selection 1

The following features are dropped for various reasons as outline bellow:

1. ID, Source (Do not contain information pertain to target variable)
2. Start_Time, End_Time (Transformed to new column “Time” showing total accident time)
3. Start_Lat, Start_Lng, End_Lat, End_Lng (Dropped as “Distance.mi” is in the dataset)
4. Country, Turning_Loop, Roundabout (Only has one value)
5. Description (Unique values for each entries)

Hide

```
# Drop Columns not needed based on factors such as in already calculated columns
US_newdata_Sample <- select(US_newdata_Sample, -ID, -Source, -Start_Time, -End_Time, -Descrip
tion, -Start_Lat, -Start_Lng, -End_Lat, -End_Lng, -Country, -Turning_Loop, -Roundabout)
```

Data Transformation and Summaries

Hide

```
# Creating list of Columns
numeric_columns = names((which(sapply(select(US_newdata_Sample, -"Severity"),
                                      is.numeric))))
numeric_columns
```

```
[1] "Distance.mi."      "Temperature.F."    "Wind_Chill.F."     "Humidity..."
[5] "Pressure.in."      "Visibility.mi."    "Wind_Speed.mph."   "Precipitation.in."
[9] "Time"
```

Hide

```
categorical_columns <- names((which(sapply(US_newdata_Sample, is.character))))

categorical_columns
```

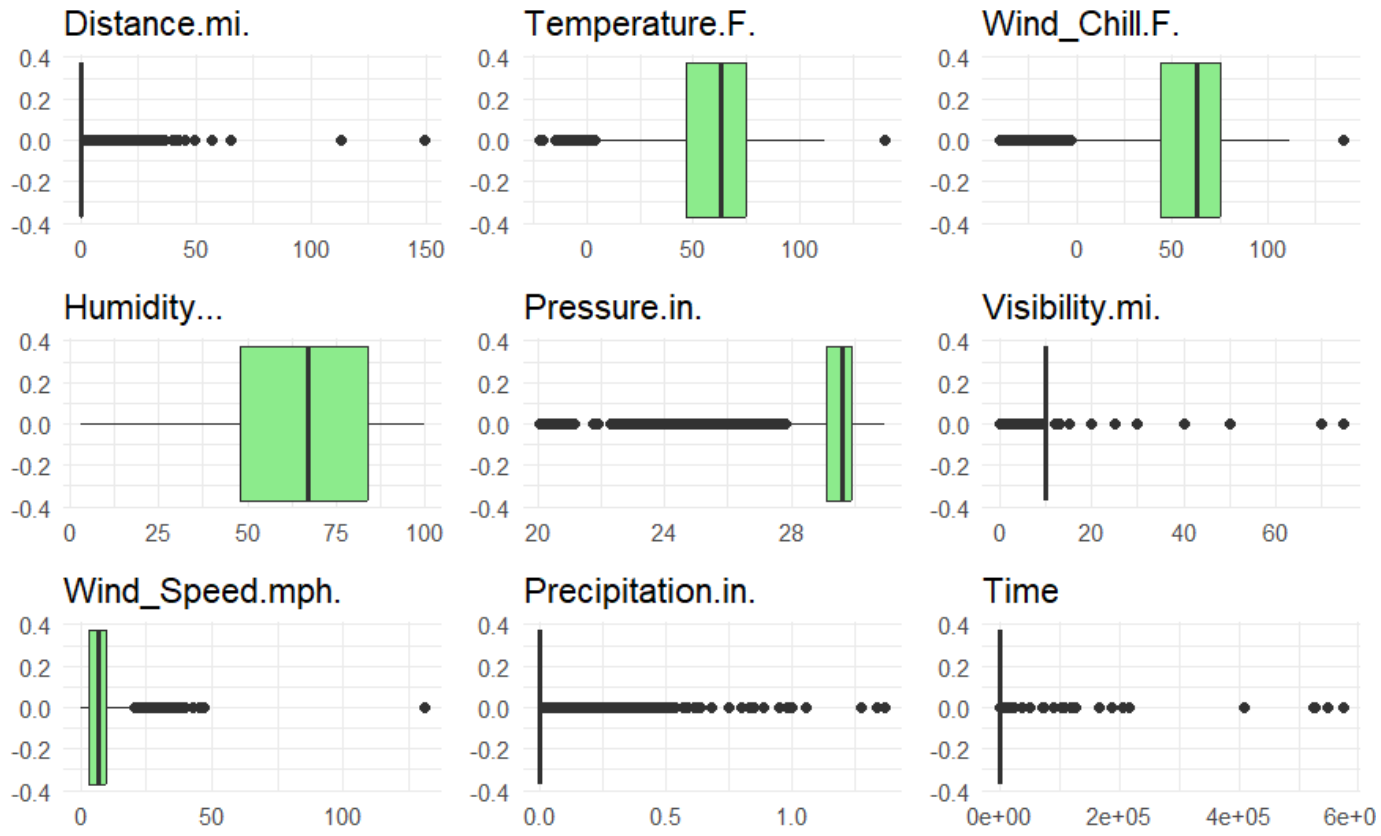
```
 [1] "Street"               "City"                 "County"
 [4] "State"                "Zipcode"              "Timezone"
 [7] "Airport_Code"         "Wind_Direction"       "Weather_Condition"
[10] "Amenity"              "Bump"                 "Crossing"
[13] "Give_Way"             "Junction"             "No_Exit"
[16] "Railway"              "Station"              "Stop"
[19] "Traffic_Calming"      "Traffic_Signal"       "Sunrise_Sunset"
[22] "Civil_Twilight"       "Nautical_Twilight"    "Astronomical_Twilight"
```

Hide

```
# check_categorical_columns <- names((which(sapply(US_newdata_Sample, is.character))))

# check_categorical_columns
```

Hide

```
# create box plot for numeric variables
plots <- lapply(numeric_columns, function(colname) {
  ggplot(US_newdata_Sample, aes(y = .data[[colname]])) +
    geom_boxplot(fill = "lightgreen") +
    theme_minimal() +
    labs(title = colname, y = NULL) +
    coord_flip()  # ← This flips the plot horizontally
})

# Arrange in 3x3 grid
do.call(grid.arrange, c(plots, ncol = 3))
```


Distance.mi.
Temperature.F.
Wind_Chill.F.
Humidity...
Pressure.in.
Visibility.mi.
Wind_Speed.mph.
Precipitation.in.
Time

Hide

```
# create side by side bar chart for categorical variables
Amenity <- ggplot(US_newdata_Sample, aes(x = Amenity,
                  fill = factor(Severity))) +
  geom_bar(position = "fill") +
  labs(x = "Amenity",
       y = "Count",
       fill = "Severity") +
  scale_fill_discrete(labels  = c("0" = "LOW", "1" = "HIGH")) +
  theme_minimal()

Bump <- ggplot(US_newdata_Sample, aes(x = Bump,
                  fill = factor(Severity))) +
  geom_bar(position = "fill") +
  labs(x = "Bump",
       y = "Count",
       fill = "Severity") +
  scale_fill_discrete(labels  = c("0" = "LOW", "1" = "HIGH")) +
  theme_minimal()

Crossing <- ggplot(US_newdata_Sample, aes(x = Crossing,
                  fill = factor(Severity))) +
  geom_bar(position = "fill") +
  labs(x = "Crossing",
       y = "Count",
       fill = "Severity") +
  scale_fill_discrete(labels  = c("0" = "LOW", "1" = "HIGH")) +
  theme_minimal()

Give_Way <- ggplot(US_newdata_Sample, aes(x = Give_Way,
                  fill = factor(Severity))) +
  geom_bar(position = "fill") +
  labs(x = "Give_Way",
       y = "Count",
       fill = "Severity") +
  scale_fill_discrete(labels  = c("0" = "LOW", "1" = "HIGH")) +
  theme_minimal()

Junction <- ggplot(US_newdata_Sample, aes(x = Junction,
                  fill = factor(Severity))) +
  geom_bar(position = "fill") +
  labs(x = "Junction",
       y = "Count",
       fill = "Severity") +
  scale_fill_discrete(labels  = c("0" = "LOW", "1" = "HIGH")) +
  theme_minimal()


No_Exit <- ggplot(US_newdata_Sample, aes(x = No_Exit,
                  fill = factor(Severity))) +
  geom_bar(position = "fill") +
  labs(x = "No_Exit",
       y = "Count",
       fill = "Severity") +
  scale_fill_discrete(labels  = c("0" = "LOW", "1" = "HIGH")) +
  theme_minimal()
```

```
Railway <- ggplot(US_newdata_Sample, aes(x = Railway,
                  fill = factor(Severity))) +
  geom_bar(position = "fill") +
  labs(x = "Railway",
       y = "Count",
       fill = "Severity") +
  scale_fill_discrete(labels  = c("0" = "LOW", "1" = "HIGH")) +
  theme_minimal()

Station <- ggplot(US_newdata_Sample, aes(x = Station,
                  fill = factor(Severity))) +
  geom_bar(position = "fill") +
  labs(x = "Station",
       y = "Count",
       fill = "Severity") +
  scale_fill_discrete(labels  = c("0" = "LOW", "1" = "HIGH")) +
  theme_minimal()

Stop <- ggplot(US_newdata_Sample, aes(x = Stop,
                  fill = factor(Severity))) +
  geom_bar(position = "fill") +
  labs(x = "Stop",
       y = "Count",
       fill = "Severity") +
  scale_fill_discrete(labels  = c("0" = "LOW", "1" = "HIGH")) +
  theme_minimal()

Traffic_Calming <- ggplot(US_newdata_Sample, aes(x = Traffic_Calming,
                  fill = factor(Severity))) +
  geom_bar(position = "fill") +
  labs(x = "Traffic_Calming",
       y = "Count",
       fill = "Severity") +
  scale_fill_discrete(labels  = c("0" = "LOW", "1" = "HIGH")) +
  theme_minimal()

Traffic_Signal <- ggplot(US_newdata_Sample, aes(x = Traffic_Signal,
                  fill = factor(Severity))) +
  geom_bar(position = "fill") +
  labs(x = "Traffic_Signal",
       y = "Count",
       fill = "Severity") +
  scale_fill_discrete(labels  = c("0" = "LOW", "1" = "HIGH")) +
  theme_minimal()

# Arrange in 3x3 grid
grid.arrange(
  Amenity,
  Bump,
  Crossing,
  Give_Way,
  Junction,
  No_Exit,
  Railway,
  Station,
```

```
  Stop,
  Traffic_Calming,
  Traffic_Signal,
  ncol = 3
)
```

Hide

```
# Encoding categorical Variables
US_newdata_Sample[categorical_columns] <- lapply(
  US_newdata_Sample[categorical_columns],
  function(x) as.integer(as.factor(x)) - 1
)

head(US_newdata_Sample)
```

	Severity <dbl>	Distance.mi. <dbl>	Street <dbl>	City <dbl>	Cou… <dbl>	State <dbl>	Zipcode <dbl>	Timezone <dbl>	Airport_Code <dbl>
1	0	0.019	6889	27	9	4	10395	2	89
2	0	1.007	7282	3614	605	2	10697	2	469
3	0	1.060	8894	3558	450	43	3316	1	1104
4	0	1.027	8960	147	364	9	5155	1	499
5	0	0.039	2067	935	261	41	9639	0	332
6	0	0.101	6521	2532	637	8	6119	1	847

6 rows | 1-10 of 35 columns

Hide

```
str(US_newdata_Sample)
```

```
'data.frame':   25000 obs. of  35 variables:
 $ Severity             : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Distance.mi.         : num  0.019 1.007 1.06 1.027 0.039 ...
 $ Street               : num  6889 7282 8894 8960 2067 ...
 $ City                 : num  27 3614 3558 147 935 ...
 $ County               : num  9 605 450 364 261 637 216 320 724 412 ...
 $ State                : num  4 2 43 9 41 8 9 3 8 38 ...
 $ Zipcode              : num  10395 10697 3316 5155 9639 ...
 $ Timezone             : num  2 2 1 1 0 1 1 3 1 1 ...
 $ Airport_Code         : num  89 469 1104 499 332 ...
 $ Weather_Timestamp    : POSIXct, format: "2023-01-20 22:01:00" "2021-09-16 19:51:00" ...
 $ Temperature.F.       : num  20 101 49 37 52 73 51 57 66 61 ...
 $ Wind_Chill.F.        : num  13 101 45 37 52 73 51 57 66 61 ...
 $ Humidity...          : num  89 15 26 73 69 46 21 63 45 30 ...
 $ Pressure.in.         : num  22.6 28.5 30 29.3 29.4 ...
 $ Visibility.mi.       : num  3 10 10 7 10 10 10 10 10 10 ...
 $ Wind_Direction       : num  3 8 19 0 8 10 10 1 7 11 ...
 $ Wind_Speed.mph.      : num  5 6 9 0 6 6 9 7 13 7 ...
 $ Precipitation.in.    : num  0.01 0 0 0 0 0 0 0 0 0 ...
 $ Weather_Condition    : num  33 7 43 7 7 39 39 7 7 7 ...
 $ Amenity              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Bump                 : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Crossing             : num  1 0 0 0 0 0 0 0 0 1 ...
 $ Give_Way             : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Junction             : num  0 0 0 0 1 0 1 0 0 0 ...
 $ No_Exit              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Railway              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Station              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Stop                 : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Traffic_Calming      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Traffic_Signal       : num  1 0 0 0 0 0 0 0 0 0 ...
 $ Sunrise_Sunset       : num  1 1 0 1 1 0 0 1 0 1 ...
 $ Civil_Twilight       : num  1 1 0 1 1 0 0 1 0 1 ...
 $ Nautical_Twilight    : num  1 1 0 1 1 0 0 1 0 0 ...
 $ Astronomical_Twilight: num  1 1 0 1 1 0 0 1 0 0 ...
 $ Time                 : num  178.2 35.5 76.8 149.9 120 ...
```

Hide

```
# Standardizing Feature Columns
US_newdata_Sample <- US_newdata_Sample %>% mutate(Weather_Timestamp =
                                                    scale(Weather_Timestamp))

# Scale Categorical Columns
US_newdata_Sample[categorical_columns] <-
  lapply(US_newdata_Sample[categorical_columns], scale)


# Scale Numeric Columns
normalise <-function(x) { (x -min(x))/(max(x)-min(x)) }

US_newdata_Sample[numeric_columns] <- as.data.frame(
  lapply(US_newdata_Sample[,numeric_columns], normalise))

head(US_newdata_Sample)
```

	Severity <dbl>	Distance.mi. <dbl>	Street <dbl>	City <dbl>	County <dbl>	State <dbl>	Zipcode <dbl>	T
1	0	0.0001269290	0.02311716	-1.7319073	-1.9327793	-1.0913724	0.8598495	0.
2	0	0.0067272363	0.15227865	1.1072578	0.1318473	-1.2240307	0.9373198	0.
3	0	0.0070813014	0.68207081	1.0629329	-0.4050942	1.4954657	-0.9560843	-0.
4	0	0.0068608457	0.70376205	-1.6369255	-0.7030101	-0.7597265	-0.4843365	-0.
5	0	0.0002605384	-1.56165815	-1.0132115	-1.0598163	1.3628073	0.6659173	-1.
6	0	0.0006747278	-0.09782795	0.2508383	0.2426998	-0.8260557	-0.2370473	-0.

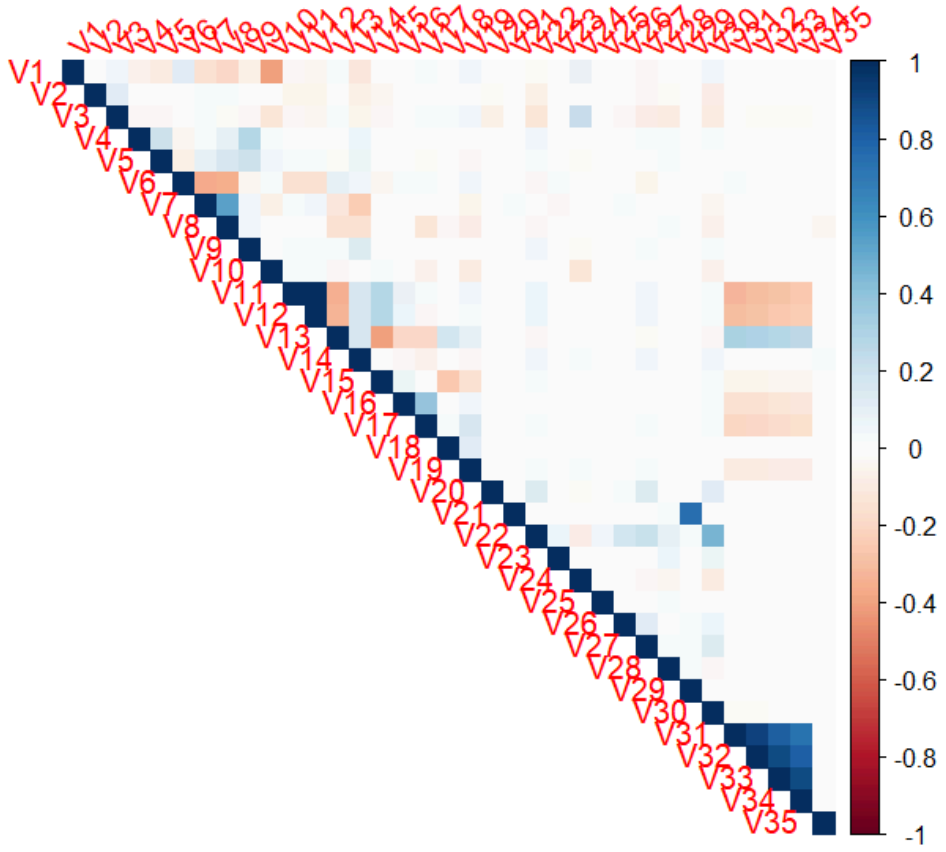
6 rows | 1-9 of 35 columns

Feature Selection 2

Correlation

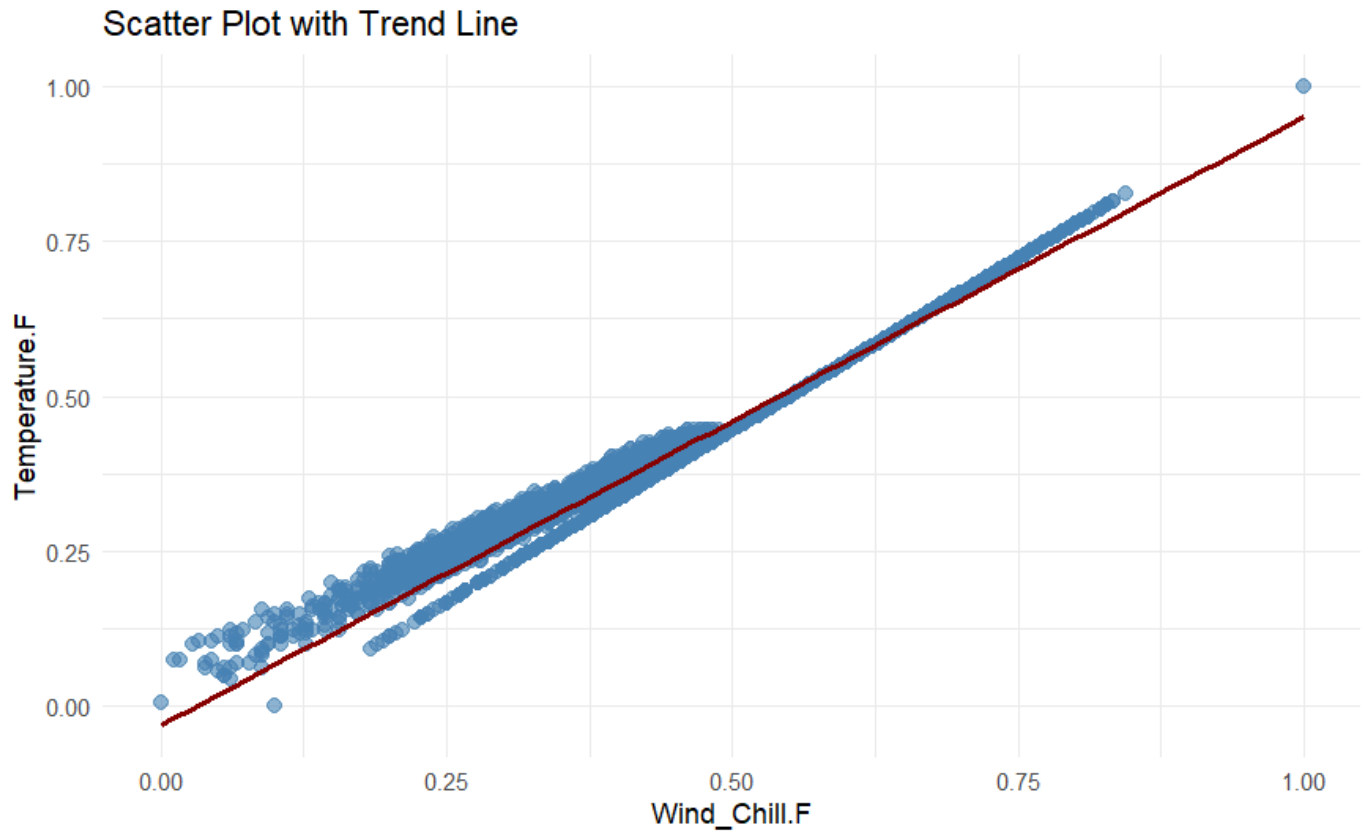
Hide

```
corrmatrix <- cor(US_newdata_Sample)
# Generate new variable names
var_codes <- paste0("V", seq_along(US_newdata_Sample))
names(var_codes) <- US_newdata_Sample  # This will be used for the key

# Rename rows and columns of the correlation matrix
colnames(corrmatrix) <- var_codes
rownames(corrmatrix) <- var_codes

corrplot::corrplot(corrmatrix, method = "color", type = "upper", tl.srt = 45)
```



Hide

```
ggplot(US_newdata_Sample, aes(x = Wind_Chill.F., y = Temperature.F.)) +
  geom_point(color = "steelblue", alpha = 0.6, size = 2.5) +
  geom_smooth(method = "lm", se = FALSE, color = "darkred") +
  labs(title = "Scatter Plot with Trend Line",
       x = "Wind_Chill.F",
       y = "Temperature.F") +
  theme_minimal()
```



Hide

```
# correlation for all variables
correlations = round(cor(US_newdata_Sample), digits = 2)

# Get variable greater than 0.8
weights <- which(abs(correlations) > 0.8 & row(correlations)<col(correlations), arr.ind=TRUE)

## reconstruct names from positions
high_cor <- matrix(colnames(correlations)[weights], ncol=2)

high_cor
```

```
     [,1]               [,2]
[1,] "Temperature.F."   "Wind_Chill.F."
[2,] "Sunrise_Sunset"   "Civil_Twilight"
[3,] "Sunrise_Sunset"   "Nautical_Twilight"
[4,] "Civil_Twilight"   "Nautical_Twilight"
[5,] "Civil_Twilight"   "Astronomical_Twilight"
[6,] "Nautical_Twilight" "Astronomical_Twilight"
```

Hide

```
##### Remove Based on correlation
US_newdata_Sample <- select(US_newdata_Sample, -Wind_Chill.F.,
                            -Nautical_Twilight, -Astronomical_Twilight,
                            -Civil_Twilight)
```

MultiCollinearity using VIF

Hide

```
# Checking for MultiCollinearity using VIF
set.seed(123)
model_data_glm <- glm(Severity ~ ., data = US_newdata_Sample, family = binomial)
model_data_vif <- car::vif(model_data_glm)
model_data_vif
```

```
     Distance.mi.            Street              City            County
         1.042293          1.120201          1.116959          1.081503
            State           Zipcode          Timezone      Airport_Code
         1.212660          1.535404          1.477869          1.124924
Weather_Timestamp    Temperature.F.       Humidity...      Pressure.in.
         1.129427          1.369699          1.557705          1.234861
   Visibility.mi.    Wind_Direction   Wind_Speed.mph. Precipitation.in.
         1.357769          1.216594          1.317629          1.100350
Weather_Condition           Amenity              Bump          Crossing
         1.099975          1.043610          1.997277          1.421154
         Give_Way          Junction           No_Exit           Railway
         1.013457          1.069210          1.005325          1.054065
          Station              Stop   Traffic_Calming    Traffic_Signal
         1.080898          1.036682          1.997021          1.343610
   Sunrise_Sunset              Time
         1.251108          1.007910
```

Building Base Model and Testing

Hide

```
# Checking for missing values
anyNA(US_newdata_Sample)
```

```
[1] FALSE
```

Hide

```
# Checking target distribution
table(US_newdata_Sample$Severity)
```

```

    0     1
13352 11648
```

Hide

```
# factor target
US_newdata_Sample$Severity <- as.factor(US_newdata_Sample$Severity)
```

Hide

```
# Splitting Data into Training and Testing Set
set.seed(42)
train_index <- createDataPartition(US_newdata_Sample$Severity, p = 0.8, list = FALSE)
train_data <- US_newdata_Sample[train_index, ]
test_data <- US_newdata_Sample[-train_index, ]
```

KNN

Hide

```
set.seed(42)
start_time <- Sys.time() # Record start time for model
model_knn <- train(Severity ~ ., data = train_data, method = "knn")
stop_time <- Sys.time() # Record model ending time
training_time <- stop_time - start_time # Calculate training time

cat("Training Time: ", training_time, "\n")
```

```
Training Time:  1.187466
```

Hide

```
print(model_knn$bestTune)
```

```
Error: object 'model_knn' not found
```

Hide

```
# Load KNN base model
model_knn <- readRDS("caret_knn_base.rds")
model_knn
```

```
k-Nearest Neighbors

20001 samples
   30 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 20001, 20001, 20001, 20001, 20001, 20001, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.6923616  0.3784880
  7  0.6994544  0.3922544
  9  0.7029754  0.3989455

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 9.
```

Hide

```
# Base KNN Test Prediction
knn_test_predict <- predict(model_knn, test_data)
confusionMatrix(knn_test_predict, test_data$Severity)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2162  804
         1  508 1525

               Accuracy : 0.7375
                 95% CI : (0.7251, 0.7497)
    No Information Rate : 0.5341
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4683

 Mcnemar's Test P-Value : 3.814e-16

            Sensitivity : 0.8097
            Specificity : 0.6548
         Pos Pred Value : 0.7289
         Neg Pred Value : 0.7501
             Prevalence : 0.5341
         Detection Rate : 0.4325
   Detection Prevalence : 0.5933
      Balanced Accuracy : 0.7323

       'Positive' Class : 0
```

Naive Bayes

Hide

```
# Load NB base model
model_nb <- readRDS("caret_nb_base.rds")
model_nb
```

```
Naive Bayes

20001 samples
   30 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 20001, 20001, 20001, 20001, 20001, 20001, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.6682345  0.3390938
   TRUE      0.6884272  0.3808599

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust'
 was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
```

Hide

```
# Base Naive Bayes Test Prediction
nb_test_predict <- predict(model_nb, test_data)
confusionMatrix(nb_test_predict, test_data$Severity)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2009  761
         1  661 1568

               Accuracy : 0.7155
                 95% CI : (0.7028, 0.728)
    No Information Rate : 0.5341
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4269

 Mcnemar's Test P-Value : 0.008656

            Sensitivity : 0.7524
            Specificity : 0.6733
         Pos Pred Value : 0.7253
         Neg Pred Value : 0.7035
             Prevalence : 0.5341
         Detection Rate : 0.4019
   Detection Prevalence : 0.5541
      Balanced Accuracy : 0.7128

       'Positive' Class : 0
```

Logistic Regression

Hide

```
# Load LR base model
model_lr <- readRDS("caret_lr_base.rds")
model_lr
```

```
Generalized Linear Model

20001 samples
   30 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 20001, 20001, 20001, 20001, 20001, 20001, ...
Resampling results:

  Accuracy   Kappa
  0.7225415  0.4396518
```

Hide

```
# Base Logistic Regression Test Prediction
lr_test_predict <- predict(model_lr, test_data)
confusionMatrix(lr_test_predict, test_data$Severity)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2060  764
         1  610 1565

               Accuracy : 0.7251
                 95% CI : (0.7125, 0.7375)
    No Information Rate : 0.5341
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4454

 Mcnemar's Test P-Value : 3.666e-05

            Sensitivity : 0.7715
            Specificity : 0.6720
         Pos Pred Value : 0.7295
         Neg Pred Value : 0.7195
             Prevalence : 0.5341
         Detection Rate : 0.4121
   Detection Prevalence : 0.5649
      Balanced Accuracy : 0.7217

       'Positive' Class : 0
```

Random Forest

Hide

```
# Load RF base model
model_rf <- readRDS("caret_rf_base.rds")
model_rf
```

```
Random Forest

20001 samples
   30 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 20001, 20001, 20001, 20001, 20001, 20001, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.8245709  0.6432299
  16    0.8548752  0.7076566
  30    0.8499837  0.6978825

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 16.
```

Hide

```
# Base Random Forest Test Prediction
rf_test_predict <- predict(model_rf, test_data)
confusionMatrix(rf_test_predict, test_data$Severity)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2348  367
         1  322 1962

               Accuracy : 0.8622
                 95% CI : (0.8523, 0.8716)
    No Information Rate : 0.5341
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.7227

 Mcnemar's Test P-Value : 0.09369

            Sensitivity : 0.8794
            Specificity : 0.8424
         Pos Pred Value : 0.8648
         Neg Pred Value : 0.8590
             Prevalence : 0.5341
         Detection Rate : 0.4697
   Detection Prevalence : 0.5431
      Balanced Accuracy : 0.8609

       'Positive' Class : 0
```

Feature Selection 3

Feature Importance

Hide

```
importance_rf <- varImp(model_rf)
importance_rf
```

```
rf variable importance

  only 20 most important variables shown (out of 30)
```

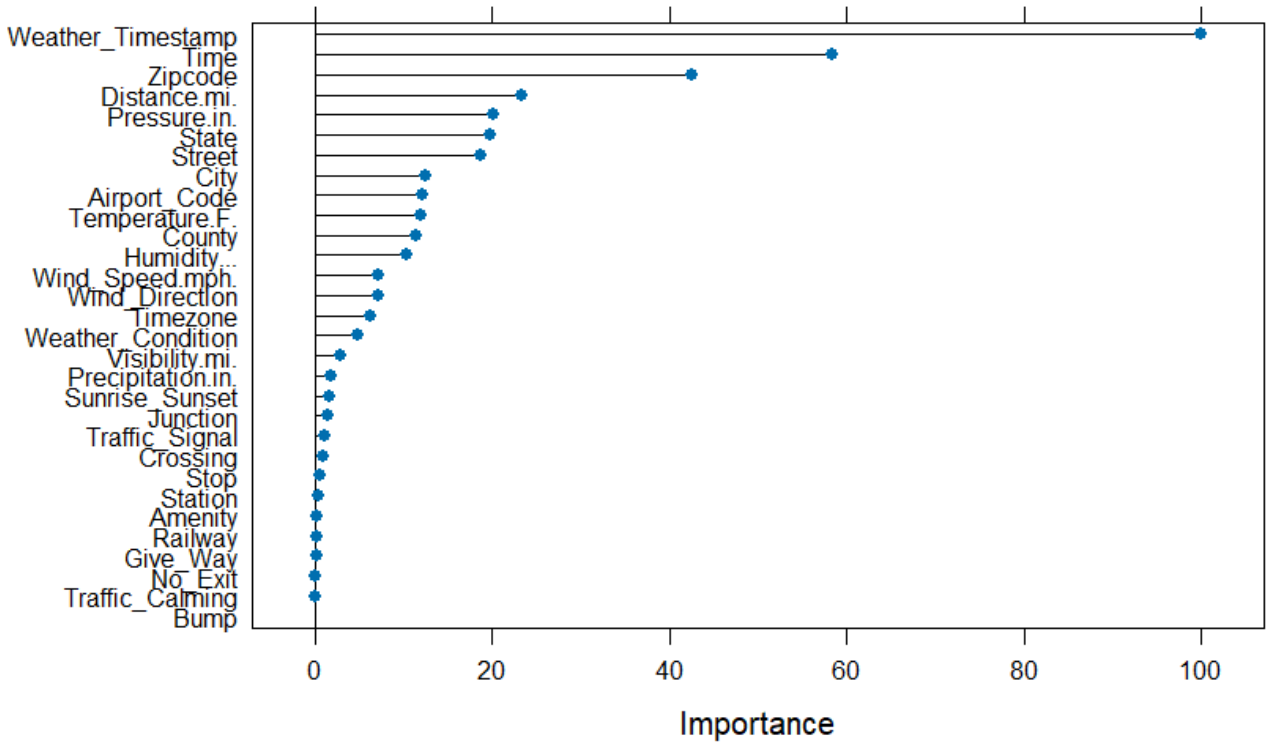
	Overall <dbl>
Weather_Timestamp	100.000000
Time	58.430005

	Overall <dbl>
Zipcode	42.540973
Distance.mi.	23.323113
Pressure.in.	20.056902
State	19.847694
Street	18.687408
City	12.424100
Airport_Code	12.118603
Temperature.F.	11.965721

1-10 of 20 rows Previous **1** 2 Next

Hide

```
plot(varImp(model_rf))
```



Hide

```
# Convert importance object to dataframe
importance_df <- as.data.frame(importance_rf$importance)
importance_df
```

	Overall <dbl>
Distance.mi.	23.3231127
Street	18.6874077
City	12.4241000
County	11.4904068
State	19.8476937
Zipcode	42.5409730
Timezone	6.1838766
Airport_Code	12.1186032
Weather_Timestamp	100.0000000
Temperature.F.	11.9657210

1-10 of 30 rows Previous **1** 2 3 Next

Hide

```
# Sort features by importance (descending order)
sorted_features <- importance_df[order(-rowMeans(importance_df)), , drop = FALSE]

# Extract the top 20 features
top_features <- rownames(sorted_features)[1:20]
top_features
```

```
 [1] "Weather_Timestamp" "Time"              "Zipcode"           "Distance.mi."
 [5] "Pressure.in."      "State"             "Street"            "City"
 [9] "Airport_Code"      "Temperature.F."    "County"            "Humidity..."
[13] "Wind_Speed.mph."   "Wind_Direction"    "Timezone"          "Weather_Condition"
[17] "Visibility.mi."    "Precipitation.in." "Sunrise_Sunset"    "Junction"
```

Hide

```
### Extract New Train and Test Data based of Selected Features
selected_train_data <- train_data %>% select(all_of(top_features), Severity)
selected_test_data <- test_data %>% select(all_of(top_features), Severity)

# Check Any Missing Data in Training Data
anyNA(selected_train_data)
```

```
[1] FALSE
```

Hide

```
# Check Any Missing Data in Testing Data
anyNA(selected_test_data)
```

```
[1] FALSE
```

Hyperparameter Tuning of Model

Hide

```
# Initialize Train Control
trctrl <- trainControl(method = "repeatedcv",
                       summaryFunction = defaultSummary, classProbs = FALSE,
                       number = 5, repeats = 5, allowParallel = TRUE)
```

Tune KNN

Hide

```
# Load KNN Tuned model
model_knn_tune <- readRDS("caret_knn_tune.rds")
print(model_knn_tune$bestTune)
```

	k
	<dbl>
14	27

1 row

Hide

```
model_knn_tune
```

```
k-Nearest Neighbors

20001 samples
   20 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 16001, 16001, 16001, 16001, 16000, 16001, ...
Resampling results across tuning parameters:

  k    Accuracy   Kappa
    1  0.6998050  0.3953949
    3  0.7209139  0.4363610
    5  0.7263935  0.4466940
    7  0.7296733  0.4526942
    9  0.7301133  0.4530209
   11  0.7307433  0.4540134
   13  0.7313833  0.4550004
   15  0.7319833  0.4560331
   17  0.7323333  0.4564734
   19  0.7324233  0.4564727
   21  0.7335434  0.4585754
   23  0.7342632  0.4599027
   25  0.7340132  0.4592443
   27  0.7347432  0.4606935
   29  0.7338332  0.4587973
   31  0.7342430  0.4594965
   33  0.7339531  0.4588029
   35  0.7338133  0.4584729
   37  0.7331932  0.4571018
   39  0.7333432  0.4573352
   41  0.7336832  0.4579058
   43  0.7336431  0.4577917
   45  0.7340231  0.4584679
   47  0.7343930  0.4591780
   49  0.7336532  0.4575247
   51  0.7329931  0.4561681
   53  0.7326532  0.4554206
   55  0.7321432  0.4542716
   57  0.7320532  0.4540102
   59  0.7314832  0.4527768
   61  0.7317032  0.4531957
   63  0.7312333  0.4522394
   65  0.7307733  0.4512452
   67  0.7312033  0.4521354
   69  0.7313032  0.4523053
   71  0.7314832  0.4526439
   73  0.7311333  0.4519292
   75  0.7309533  0.4515448
   77  0.7314733  0.4525893
   79  0.7309634  0.4514704
   81  0.7307533  0.4510415
   83  0.7304535  0.4504278
   85  0.7305235  0.4505504
```

```
   87  0.7300035  0.4494923
   89  0.7301435  0.4497917
   91  0.7296636  0.4487973
   93  0.7303035  0.4500566
   95  0.7305934  0.4506069
   97  0.7299534  0.4493271
   99  0.7295335  0.4484390
  101  0.7294635  0.4482561
  103  0.7295535  0.4484207
  105  0.7292435  0.4478109
  107  0.7295134  0.4483167
  109  0.7295635  0.4484014
  111  0.7298635  0.4490154
  113  0.7294335  0.4480865
  115  0.7293735  0.4479911
  117  0.7288436  0.4469090
  119  0.7286836  0.4465660
  121  0.7285335  0.4462742
  123  0.7286335  0.4464297
  125  0.7284935  0.4461557
  127  0.7280635  0.4452710
  129  0.7281635  0.4454889
  131  0.7277235  0.4445938
  133  0.7272835  0.4437084
  135  0.7270636  0.4432530
  137  0.7269436  0.4429843
  139  0.7270136  0.4431071
  141  0.7271536  0.4433684
  143  0.7269236  0.4428533
  145  0.7266436  0.4422921
  147  0.7266236  0.4422339
  149  0.7260336  0.4410125

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 27.
```

Hide

```
# Tuned KNN Test Prediction
knn_tune_test_predict <- predict(model_knn_tune, selected_test_data)
confusionMatrix(knn_tune_test_predict, selected_test_data$Severity)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2236  817
         1  434 1512

               Accuracy : 0.7497
                 95% CI : (0.7375, 0.7617)
    No Information Rate : 0.5341
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4918

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8375
            Specificity : 0.6492
         Pos Pred Value : 0.7324
         Neg Pred Value : 0.7770
             Prevalence : 0.5341
         Detection Rate : 0.4473
   Detection Prevalence : 0.6107
      Balanced Accuracy : 0.7433

       'Positive' Class : 0
```

Tune Naive Bayes

Hide

```
# Load NB Tuned model
model_nb_tune <- readRDS("caret_nb_tune.rds")
print(model_nb_tune$bestTune)
```

	fL	usekernel	adjust
	<dbl>	<lgl>	<dbl>
4	0	TRUE	0.5

1 row

Hide

```
model_nb_tune
```

```
Naive Bayes

20001 samples
   20 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 16001, 16001, 16001, 16001, 16000, 16001, ...
Resampling results across tuning parameters:

  fL   usekernel  adjust  Accuracy   Kappa
  0.0  FALSE      0.5     0.6656467  0.3133609
  0.0  FALSE      1.0     0.6656467  0.3133609
  0.0  FALSE      2.0     0.6656467  0.3133609
  0.0   TRUE      0.5     0.7245439  0.4373821
  0.0   TRUE      1.0     0.7189240  0.4238247
  0.0   TRUE      2.0     0.7166742  0.4173014
  0.5  FALSE      0.5     0.6656467  0.3133609
  0.5  FALSE      1.0     0.6656467  0.3133609
  0.5  FALSE      2.0     0.6656467  0.3133609
  0.5   TRUE      0.5     0.7245439  0.4373821
  0.5   TRUE      1.0     0.7189240  0.4238247
  0.5   TRUE      2.0     0.7166742  0.4173014
  1.0  FALSE      0.5     0.6656467  0.3133609
  1.0  FALSE      1.0     0.6656467  0.3133609
  1.0  FALSE      2.0     0.6656467  0.3133609
  1.0   TRUE      0.5     0.7245439  0.4373821
  1.0   TRUE      1.0     0.7189240  0.4238247
  1.0   TRUE      2.0     0.7166742  0.4173014

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 0.5.
```

Hide

```
# Tuned Naive Bayes Test Prediction
nb_tune_test_predict <- predict(model_nb_tune, selected_test_data)
```

```
Warning: Numerical 0 probability for all classes with observation 492Warning: Numerical 0 pro
bability for all classes with observation 3008Warning: Numerical 0 probability for all classe
s with observation 4564
```

Hide

```
confusionMatrix(nb_tune_test_predict, selected_test_data$Severity)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2259  946
         1  411 1383

               Accuracy : 0.7285
                 95% CI : (0.716, 0.7408)
    No Information Rate : 0.5341
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4464

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8461
            Specificity : 0.5938
         Pos Pred Value : 0.7048
         Neg Pred Value : 0.7709
             Prevalence : 0.5341
         Detection Rate : 0.4519
   Detection Prevalence : 0.6411
      Balanced Accuracy : 0.7199

       'Positive' Class : 0
```

Tune Logistic Regression

Hide

```
# Load LR Tuned model
model_lr_tune <- readRDS("caret_lr_tune.rds")

# Predict with probability to find best threshold
train_probabilities <- predict(model_lr_tune, selected_train_data, type = "prob")

# set a dataframe to save results
lr_results <- data.frame()

# Set a probability cutoff
cutoffs <- c(0.3, 0.4, 0.5, 0.6, 0.7)

for (cutoff in cutoffs){
  train_predicted_class <- ifelse(train_probabilities[,2] > cutoff, 1, 0)

  # Convert to factor with correct levels
  train_predicted_class <- factor(train_predicted_class,
                                  levels = levels(selected_train_data$Severity))

  # Evaluate accuracy on training data
  cm <- confusionMatrix(train_predicted_class, selected_train_data$Severity)

  Accuracy <- cm$overall["Accuracy"]
  Kappa <- cm$overall["Kappa"]
  df_cm <- data.frame(cutoff, Accuracy, Kappa)
  lr_results <- rbind(lr_results, df_cm)
}

lr_results
```

	cutoff <dbl>	Accuracy <dbl>	Kappa <dbl>
Accuracy	0.3	0.6414679	0.3011257
Accuracy1	0.4	0.6875156	0.3786577
Accuracy2	0.5	0.7237638	0.4420431
Accuracy3	0.6	0.7363632	0.4604895
Accuracy4	0.7	0.7074646	0.3928161

5 rows

Hide

```
# Tuned Logistic Regression Test Prediction

# Predict with probability to find best threshold
test_probabilities <- predict(model_lr_tune, selected_test_data, type = "prob")

# Set best probability cutoff
cutoff <- 0.6

test_predicted_class <- ifelse(test_probabilities[,2] > cutoff, 1, 0)

# Convert to factor with correct levels
test_predicted_class <- factor(test_predicted_class,
                               levels = levels(selected_test_data$Severity))

# Evaluate accuracy on training data
confusionMatrix(test_predicted_class, selected_test_data$Severity)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2344  955
         1  326 1374

               Accuracy : 0.7437
                 95% CI : (0.7314, 0.7558)
    No Information Rate : 0.5341
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4761

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8779
            Specificity : 0.5900
         Pos Pred Value : 0.7105
         Neg Pred Value : 0.8082
             Prevalence : 0.5341
         Detection Rate : 0.4689
   Detection Prevalence : 0.6599
      Balanced Accuracy : 0.7339

       'Positive' Class : 0
```

Tune Random Forest

Hide

```
# Load RF base model
model_rf_tune <- readRDS("caret_rf_tune.rds")

# Print the best hyperparameters
print(model_rf_tune$finalModel)
```

```
Call:
 randomForest(x = x, y = y, ntree = ..1, mtry = param$mtry, nodesize = ..2)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 9

        OOB estimate of  error rate: 13.68%
Confusion matrix:
     0    1 class.error
0 9385 1297   0.1214192
1 1440 7879   0.1545230
```

Hide

```
model_rf_tune
```

```
Random Forest

20001 samples
   20 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 16000, 16001, 16002, 16000, 16001, 16000, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   1    0.8007602  0.5928636
   2    0.8436876  0.6836665
   3    0.8519171  0.7008603
   4    0.8552171  0.7078208
   5    0.8576869  0.7130234
   6    0.8586569  0.7151143
   7    0.8592369  0.7163680
   8    0.8596869  0.7173261
   9    0.8600068  0.7180478
  10    0.8597370  0.7175212
  11    0.8597670  0.7176259
  12    0.8594970  0.7170772
  13    0.8590370  0.7161796
  14    0.8591570  0.7163863
  15    0.8590971  0.7162769
  16    0.8582971  0.7146896
  17    0.8573872  0.7128362
  18    0.8579672  0.7140331
  19    0.8575073  0.7130929
  20    0.8573172  0.7126973

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 9.
```

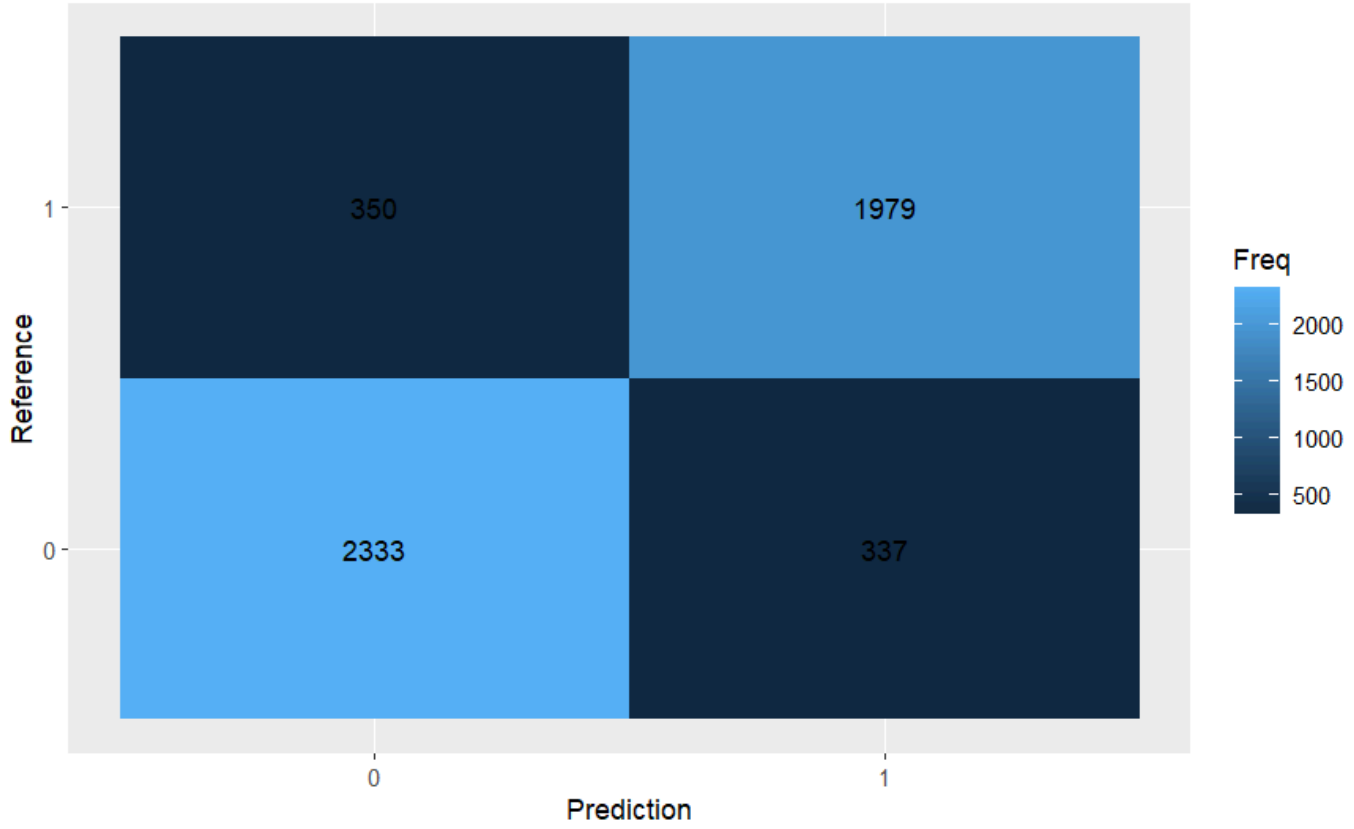
Hide

```
# Tuned Random Forest Test Prediction
rf_tune_test_predict <- predict(model_rf_tune, selected_test_data)
best_model_matrix <- confusionMatrix(rf_tune_test_predict, selected_test_data$Severity)
best_model_matrix
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2333  350
         1  337 1979

               Accuracy : 0.8626
                 95% CI : (0.8527, 0.872)
    No Information Rate : 0.5341
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7238

 Mcnemar's Test P-Value : 0.6471

            Sensitivity : 0.8738
            Specificity : 0.8497
         Pos Pred Value : 0.8695
         Neg Pred Value : 0.8545
             Prevalence : 0.5341
         Detection Rate : 0.4667
   Detection Prevalence : 0.5367
      Balanced Accuracy : 0.8618

       'Positive' Class : 0
```

Hide

```
# Plotting the confusion Matrix
model_conf_mat_data = as.data.frame(best_model_matrix$table)
model_conf_matrix_plot <- model_conf_mat_data %>%
  ggplot(aes(x = Prediction, y = Reference, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq))
model_conf_matrix_plot
```



Model Evaluation

Hide

```
# Plot ROC Curve
test_labels_binary <- as.numeric(selected_test_data$Severity) - 1
knn_model_binary <- as.numeric(knn_tune_test_predict) - 1
nb_model_binary <- as.numeric(nb_tune_test_predict) - 1
lr_model_binary <- as.numeric(test_predicted_class) - 1
rf_model_binary <- as.numeric(rf_tune_test_predict) - 1
```

Hide

```
# Compute the ROC Curve and AUC
roc_knn <- roc(test_labels_binary, knn_model_binary)
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```

Hide

```
roc_nb <- roc(test_labels_binary, nb_model_binary)
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```

Hide

```
roc_lr <- roc(test_labels_binary, lr_model_binary)
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```

Hide

```
roc_rf <- roc(test_labels_binary, rf_model_binary)
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```

Hide

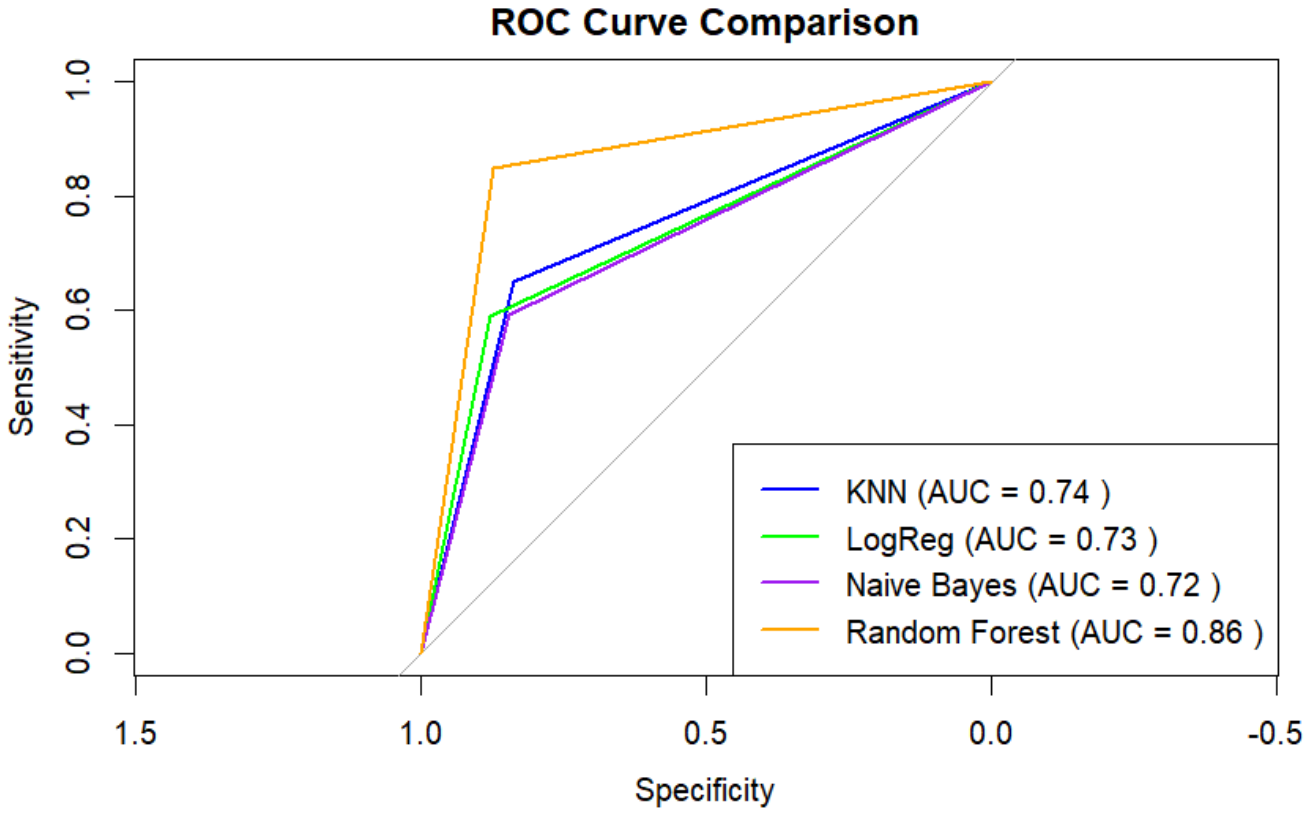
```
# Plot all ROC curves together
plot(roc_knn, col = "blue", lwd = 2, main = "ROC Curve Comparison")
plot(roc_lr, col = "green", lwd = 2, add = TRUE)
```

Hide

```
plot(roc_nb, col = "purple", lwd = 2, add = TRUE)
plot(roc_rf, col = "orange", lwd = 2, add = TRUE)
```

Hide

```

# Add legend
legend("bottomright", legend = c(
    paste("KNN (AUC =", round(auc(roc_knn), 2), ")"),
    paste("LogReg (AUC =", round(auc(roc_lr), 2), ")"),
    paste("Naive Bayes (AUC =", round(auc(roc_nb), 2), ")"),
    paste("Random Forest (AUC =", round(auc(roc_rf), 2), ")")
  ),
  col = c("blue", "green", "purple", "orange"),
  lwd = 2
)
```



Hide

```
F1_knn <- F1_Score(test_labels_binary, knn_model_binary, positive = 0)
F1_nb <- F1_Score(test_labels_binary, nb_model_binary, positive = 0)
F1_lr <- F1_Score(test_labels_binary, lr_model_binary, positive = 0)
F1_rf <- F1_Score(test_labels_binary, rf_model_binary, positive = 0)

f1_scores <- data.frame(
  Model = c("KNN", "Naive Bayes", "Logistic Regression", "Random Forest"),
  F1 = c(F1_knn, F1_nb, F1_lr, F1_rf)
)

f1_scores
```

Model <chr>	F1 <dbl>
KNN	0.7814084
Naive Bayes	0.7690213
Logistic Regression	0.7853912
Random Forest	0.8716608

4 rows