Introduction

US Accidents dataset is available on Kaggle. It contains information about traffic accidents that occurred in the United States from February 2016 to December 2020. The dataset is in CSV format and contains 47 columns and 2845342 rows. It is a relatively large dataset. The data includes a variety of features such as the location of the accident, the severity of the accident, the weather conditions at the time of the accident, and the number of people involved.

The US Accidents dataset is a valuable resource that can be used for a wide range of analyses and applications related to road safety and accident prevention. This dataset can be used to identify high-risk areas and accident hotspots by analyzing the frequency and severity of accidents in different locations. This information can be used to improve road safety measures and infrastructure. This dataset can be used to analyze trends in accidents over time, such as the number of accidents during different seasons or the impact of road construction on accident rates. This information can be used to develop better accident prevention strategies. This dataset can be used to build machine learning models that predict the severity of accidents based on various features such as weather conditions, time of day, and location. These models can help emergency services to prioritize their response to accidents based on the predicted severity.

Data Cleaning

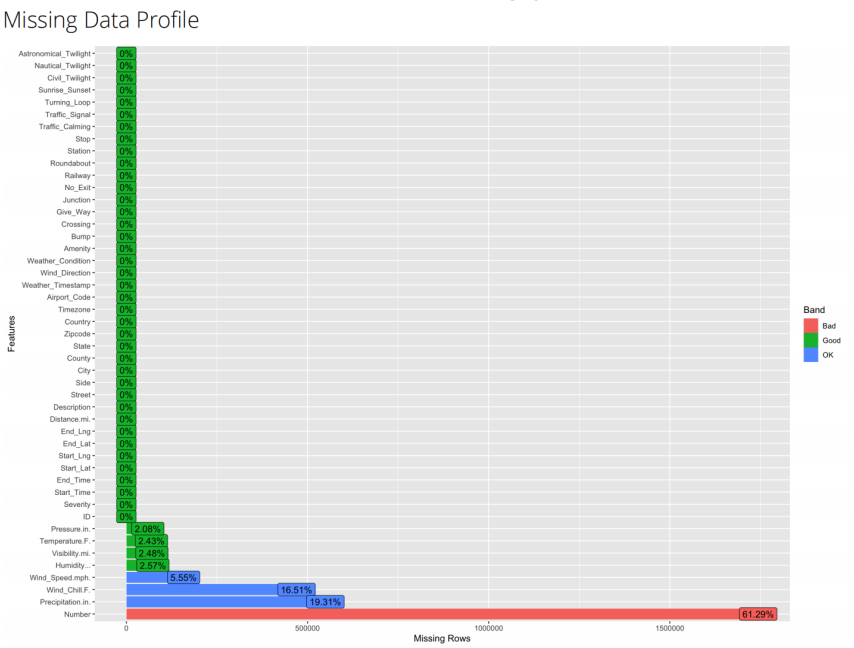
Basic Statistics

Raw Counts

Name	Value
Rows	2,845,342
Columns	47
Discrete columns	33
Continuous columns	14
All missing columns	0
Missing observations	3,193,068
Complete Rows	947,262
Total observations	133,731,074
Memory allocation	1.7 Gb

The US Accidents dataset was cleaned by removing irrelevant columns, missing values, duplicates, and outliers.

Before cleaning the dataset, we checked which columns had missing values. We used the create_report function from the DataExplorer package to generate a report on the dataset. This report provided an overview of the missing values in the dataset and helped us identify the columns containing missing values. This allowed us to identify the columns that contained missing values and helped us make the decision to clean those columns.

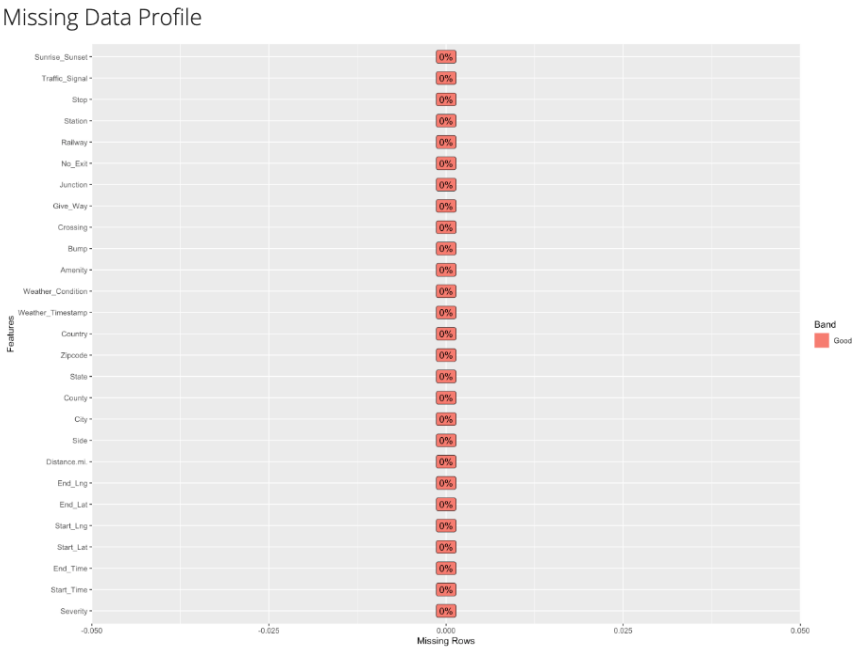


Based on the plot generated by the DataExplorer package, we identified that the following columns had missing values: "Number", "Precipitation", "Wind_Chill", "Humidity", "Visibility", "Temperature", and "Pressure".

1. Remove some columns

We removed some columns that were irrelevant to our analysis. These columns were "ID", "description", "Timezone", "Airport_Code", "Weather_Timestamp", and "Wind_Direction".

2. Cleaning N/A Values



There were some columns in the dataset that were not relevant to our analysis and contained many missing values. Therefore, we removed these columns using the negative indexing approach. After removing these columns, we used the na.omit function to remove any rows with missing values.

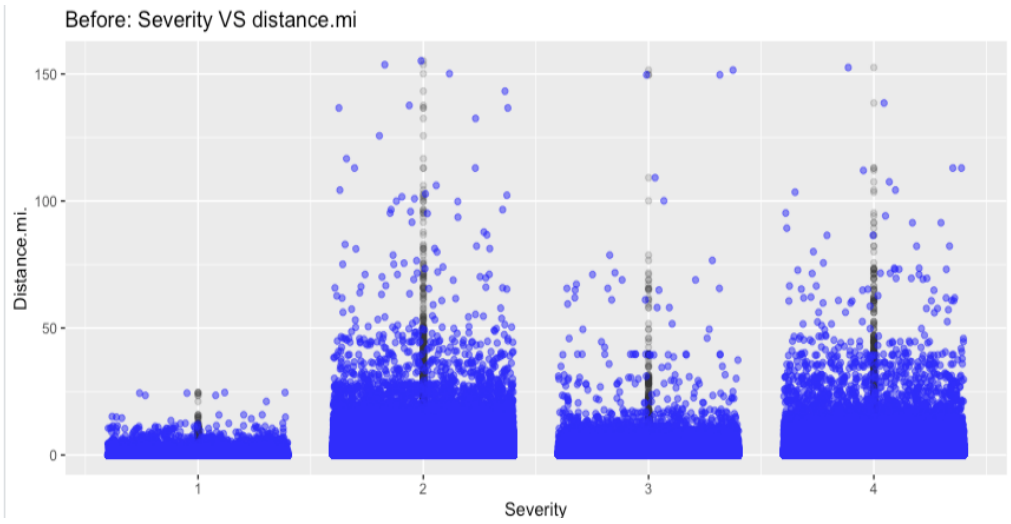
3. Checking for Duplication

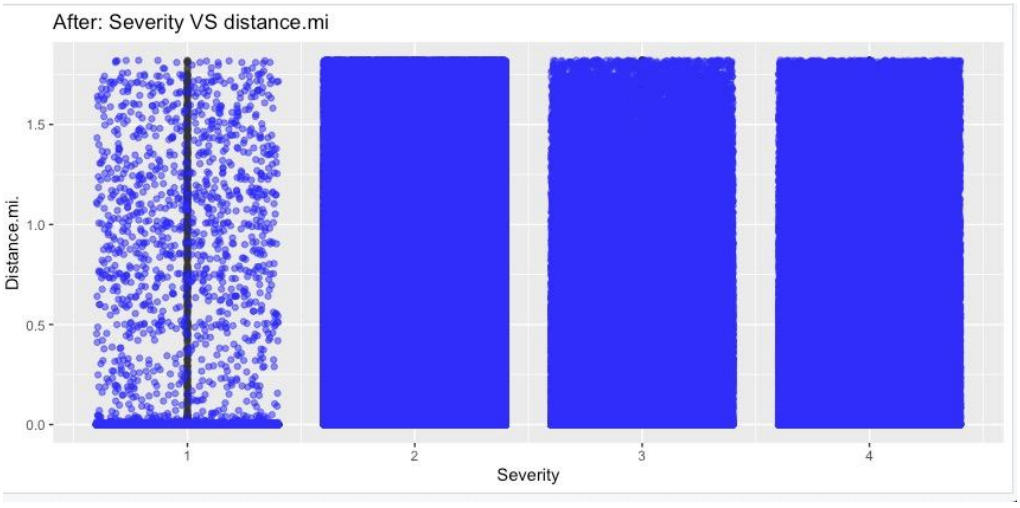
We used the duplicated function to check for any duplicated rows in the dataset. We also used the any duplicated function to check if there were any duplicated rows. We installed and loaded the "dplyr" package to remove duplicated rows and used the distinct function.

4. Transfer bool_columns to 0,1

Some columns in the dataset were Boolean variables represented as strings "True" and "False". We converted these variables to binary variables (0 and 1) using the ifelse function.

5. Quantile and Outlier Removal



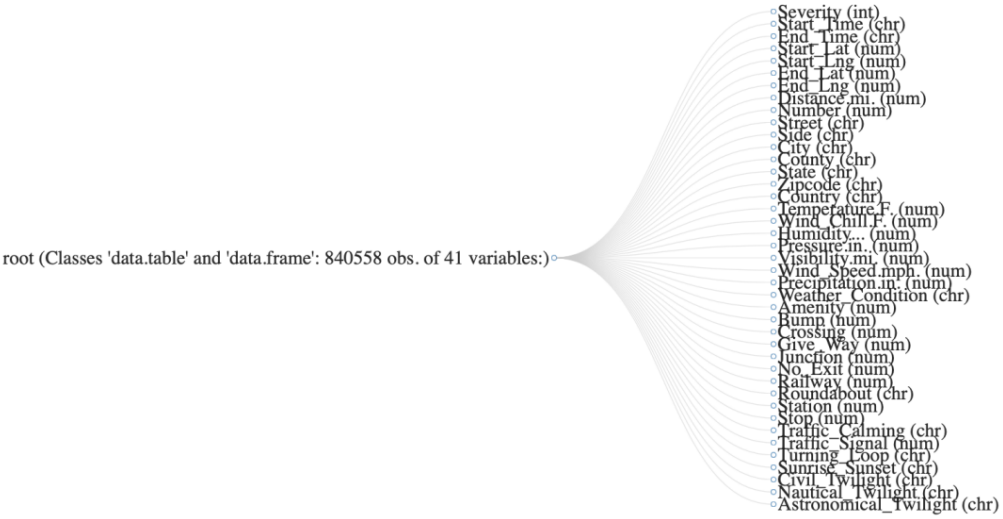


We used the quantile function to calculate the upper and lower quartiles of the "Distance.mi." variable. We then calculated the interquartile range (IQR) and used it to identify the upper and lower limits. Any values outside these limits were considered outliers and removed using the subset function.

Basic Statistics

Raw Counts

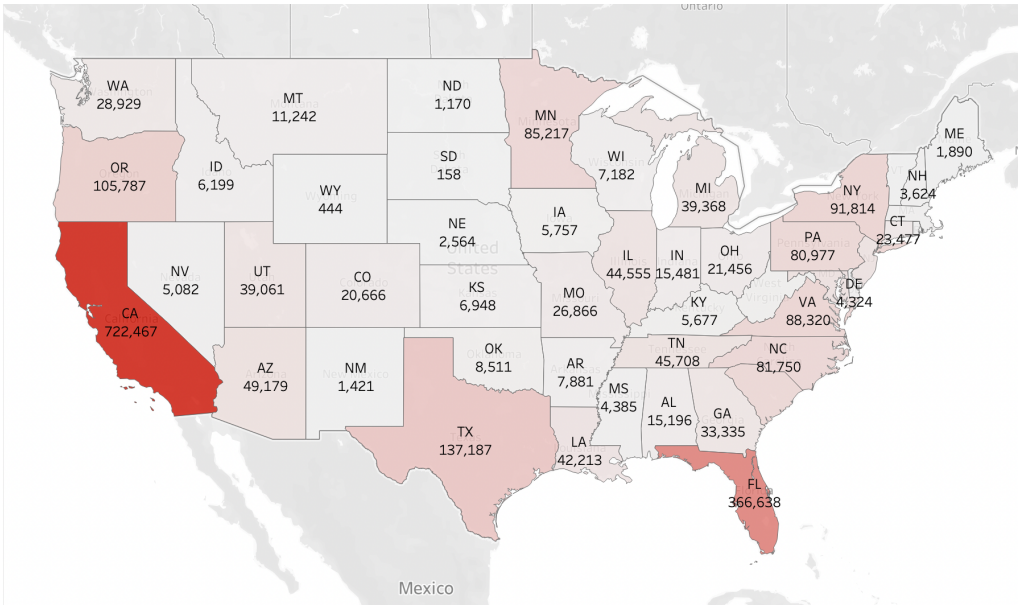
Name	Value
Rows	840,558
Columns	41
Discrete columns	17
Continuous columns	24
All missing columns	0
Missing observations	0
Complete Rows	840,558
Total observations	34,462,878
Memory allocation	386.8 Mb

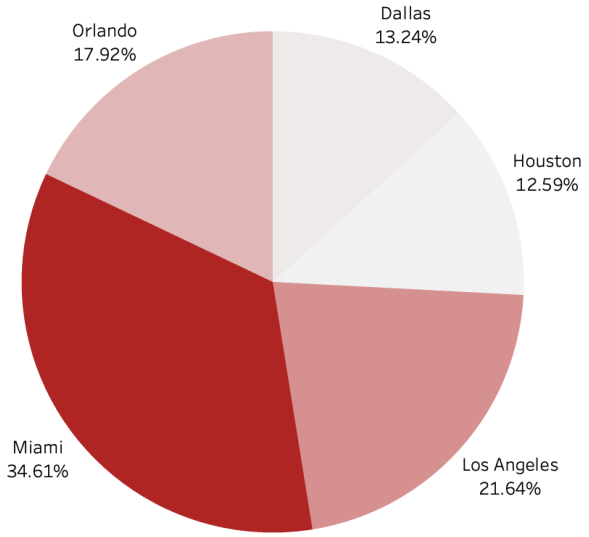


These two charts depict the cleaned dataset which will be utilized for future analysis purposes.

Location and place analysis.

- Which U.S. states and cities have the highest car accident rates?

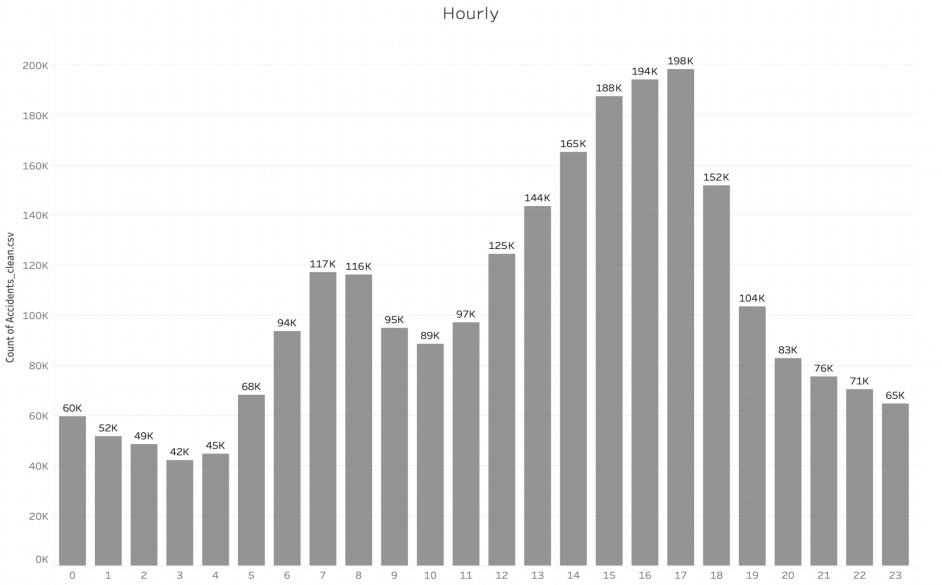




By drawing map visualization, we can more intuitively understand the geographical location and frequency of accidents. We can see that California has the highest rate of car accidents, followed by Florida. In addition, the pie chart can show accident rate data of different cities, among which Miami, Los Angeles, Dallas, Houston, and Orlando are the top five cities with higher accident rates. By analyzing trends and patterns, measures can be implemented more precisely and concretely to ensure safety.

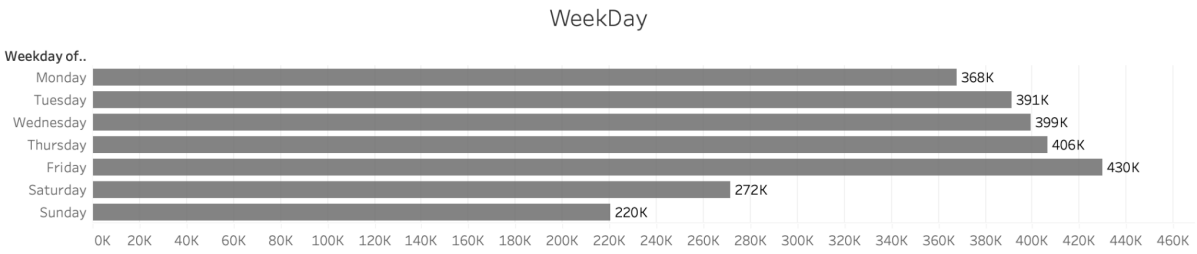
Date and Time analysis.

- When are car accidents more likely to occur in the United States?



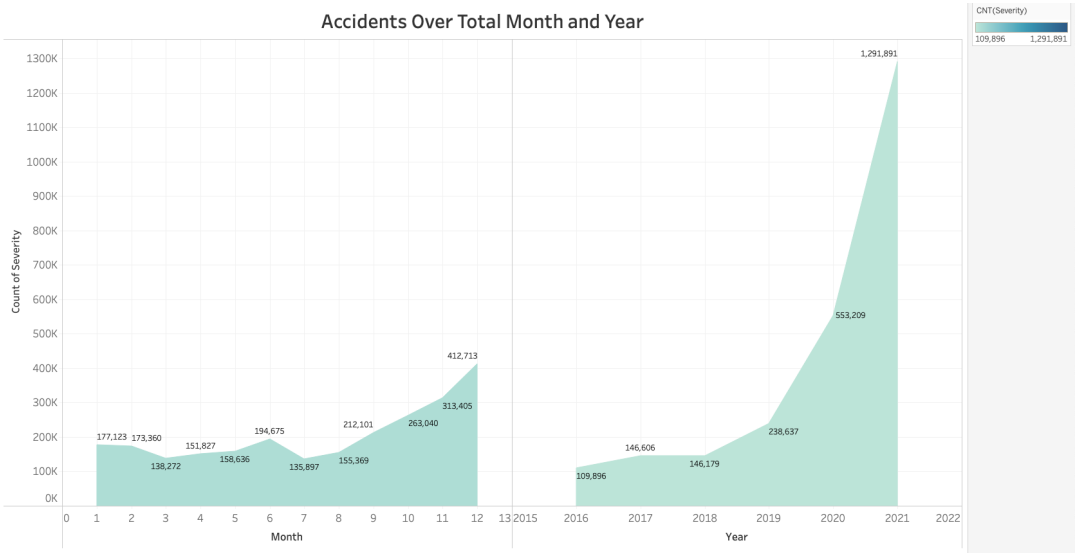
The histogram chart shows a clear pattern of when car accidents are most likely to occur. During the morning rush hour, from 6-9 am, there is a significant increase in the number of accidents. This is likely due to the high volume of cars on the road as people commute to work or school. During this time, drivers may be in a hurry and may engage in risky behavior such as speeding, changing lanes abruptly, or not paying enough attention to the road.

Similarly, the afternoon rush hour, from 3-6 pm, also shows a significant increase in accidents. Once again, this can be attributed to the high number of cars on the road during the evening commute. Additionally, fatigue may set in after a long day at work or school, making drivers more prone to making mistakes or poor decisions on the road.



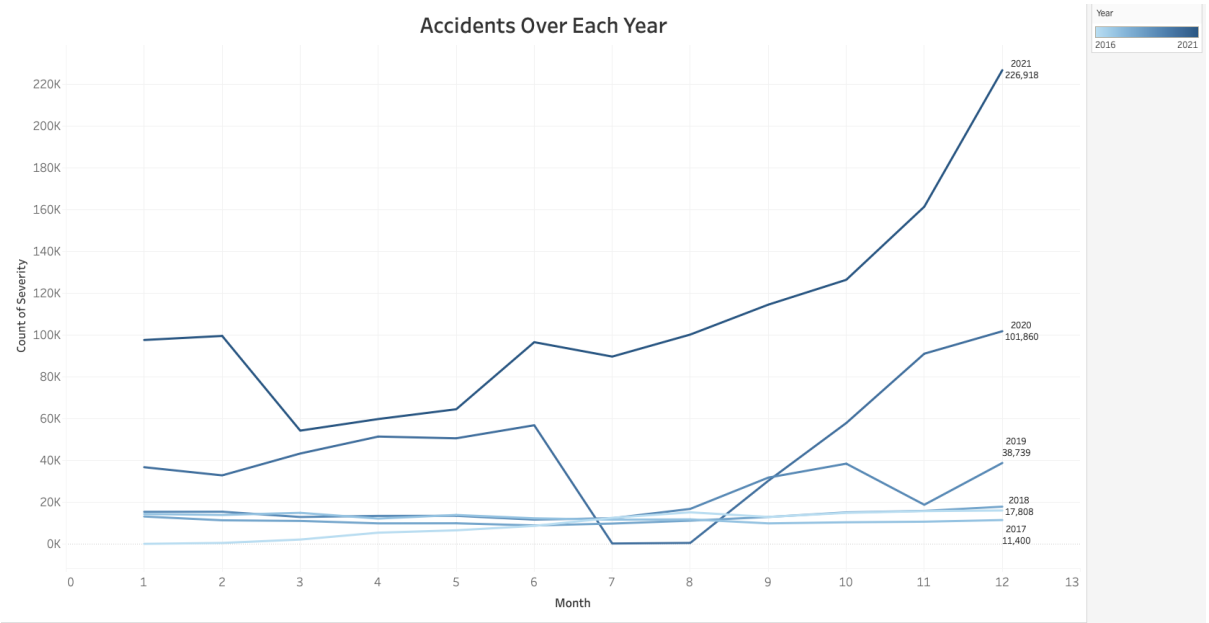
As can be seen from the histogram of the data set, there are obvious peaks on weekdays, namely Monday to Friday, while weekends are significantly lower than weekdays. This could be due to increased weekday traffic as people travel to and from work or school. Weekends, on the other hand, usually see less traffic and a lower likelihood of accidents.

- <u>How has the number of accidents changed over time?</u>



The line graph we created shows the trend of the number of accidents over time:

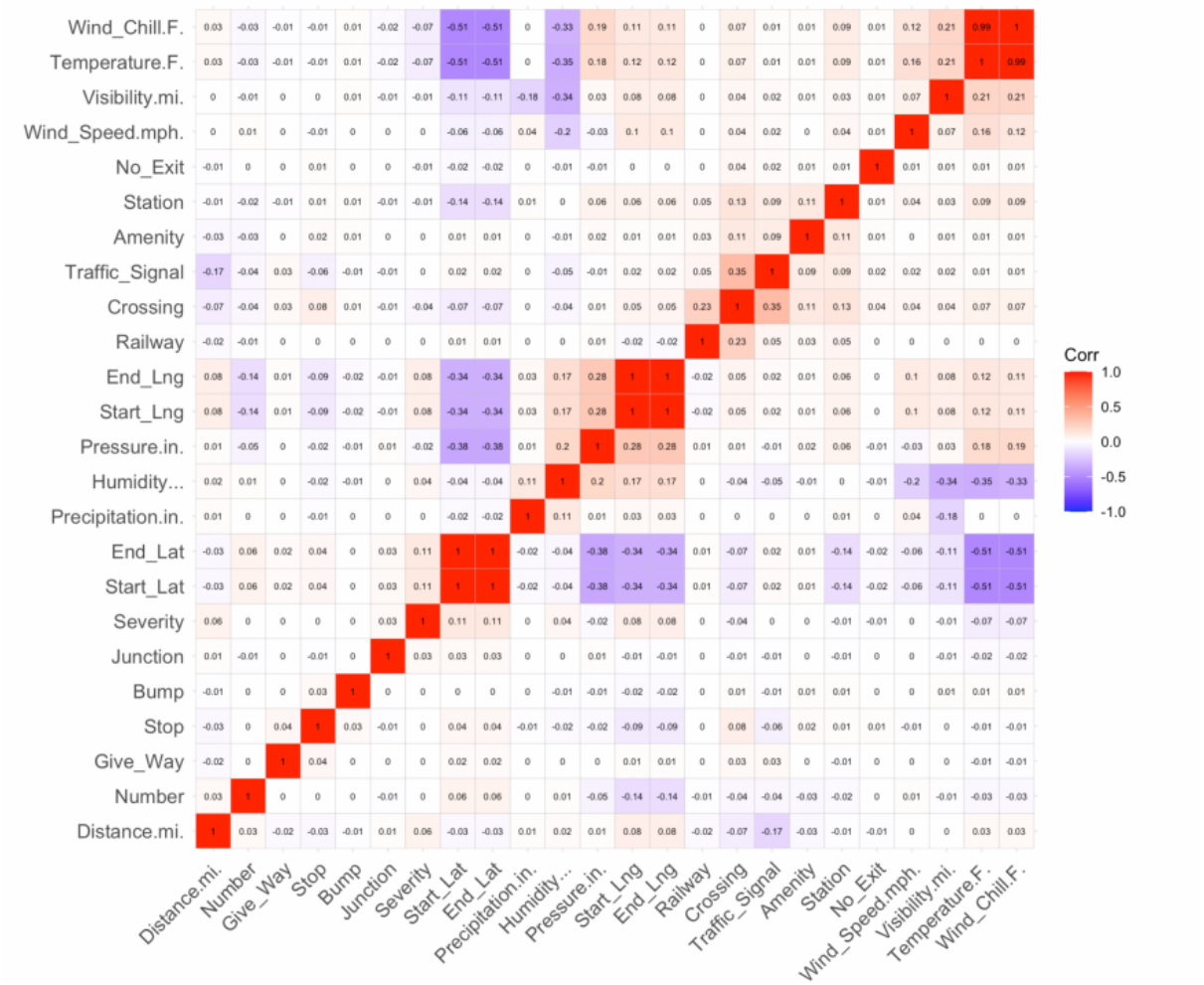
1. The line graph that we created shows that the number of accidents varies significantly by month. We can see that there is a peak in the number of accidents during the winter months, particularly in November and December. This is likely due to the adverse weather conditions during these months, such as snow, ice, and reduced visibility, which can increase the risk of accidents. Conversely, the summer months show a trough in the number of accidents, with the lowest number of accidents occurring in July. This could be due to the generally better weather conditions during the summer months, as well as reduced traffic congestion due to summer holidays and vacation periods.
2. We can observe that the number of accidents has been increasing steadily since 2016, with a significant peak in 2020. This trend is a cause for concern as it indicates that there is a growing risk of accidents on the roads.



The line graph that we created shows a clear trend in the number of accidents overtime on a year-to-year basis. We can see that there has been a steady increase in the number of accidents since 2016, with a peak in 2020. In addition to the overall trend of increasing accidents since 2016, the line graph also shows a particularly sharp increase in 2021. The number of accidents in 2021 has risen significantly compared to previous years, reaching a new peak of over 1.2 million accidents.

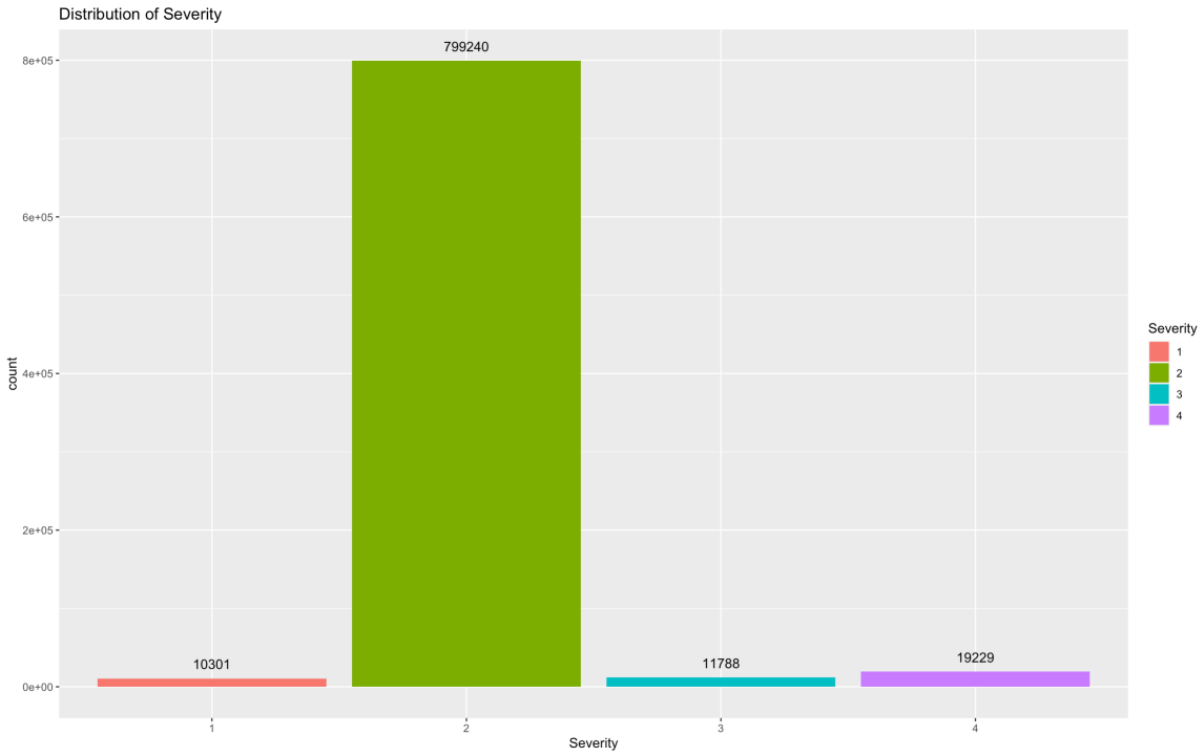
External Factors Analysis.

- <u>What are the factors that affect the severity of accidents?</u>



To determine the factors contributing to accidents, we utilized a correlation matrix to analyze the relationship between Severity and other variables. Our analysis revealed that there were no variables that were highly correlated with "Severity". Among the variables, "Distance.mi.","Junction", "Latitude", "Longitude", “Humidity” had a weak positive correlation with "Severity", with correlation coefficients of 0.06, 0.03, 0.11, 0.08, and 0.04 respectively. On the other hand, “Pressure.in.", "Crossing", "Station", “No_Exit”, “Visibility.mi.”, “Temperature.F.”, and "Wind_Chill.F." had negative correlations with "Severity", with correlation coefficients of -0.02, -0.04, -0.01, -0.01, -0.01, -0.07 and -0.07 respectively.

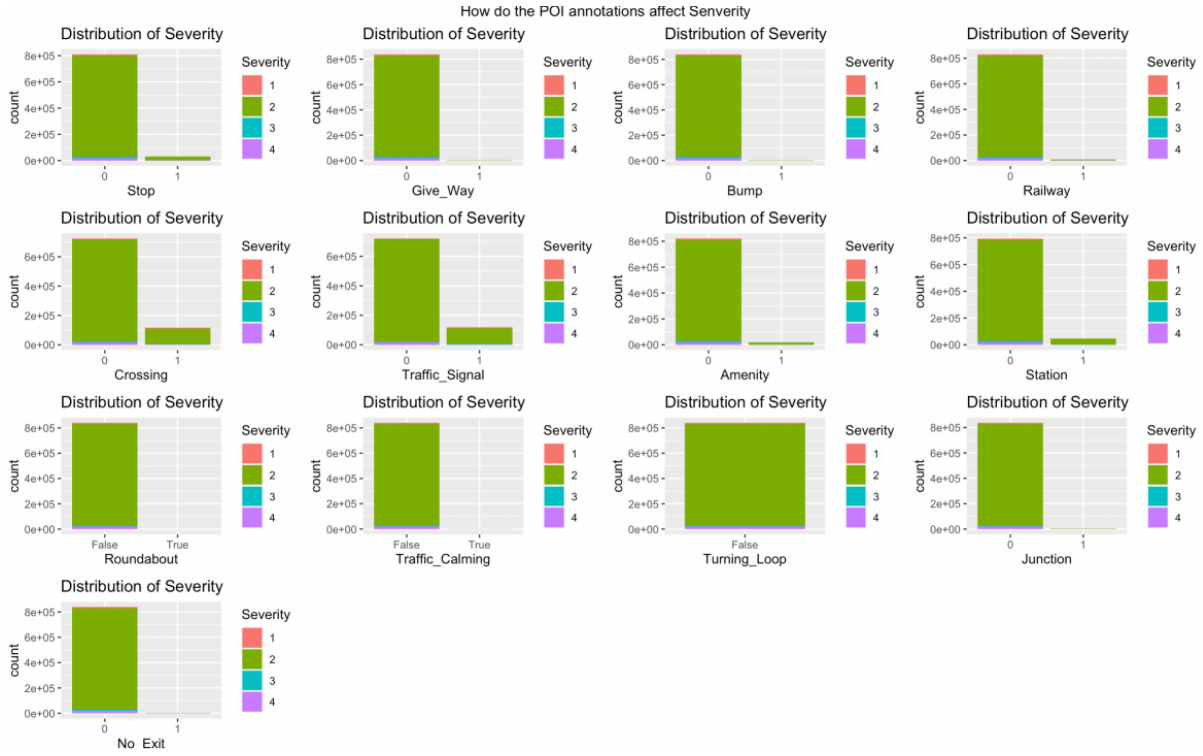
- What is the most common type of accident in the United States?



The bar chart illustrates the distribution of Severity levels of accidents on a scale of 1 to 4, with 1 indicating the least impact on traffic.

It is evident from the graph that the majority of accidents (799240 cases) have a Severity level of 2, indicating a moderate impact on traffic. This could mean that these accidents have caused some disruptions to the flow of traffic. Only a small number of accidents (only 10,301 cases) have a Severity level of 1, which indicates the least impact on traffic.

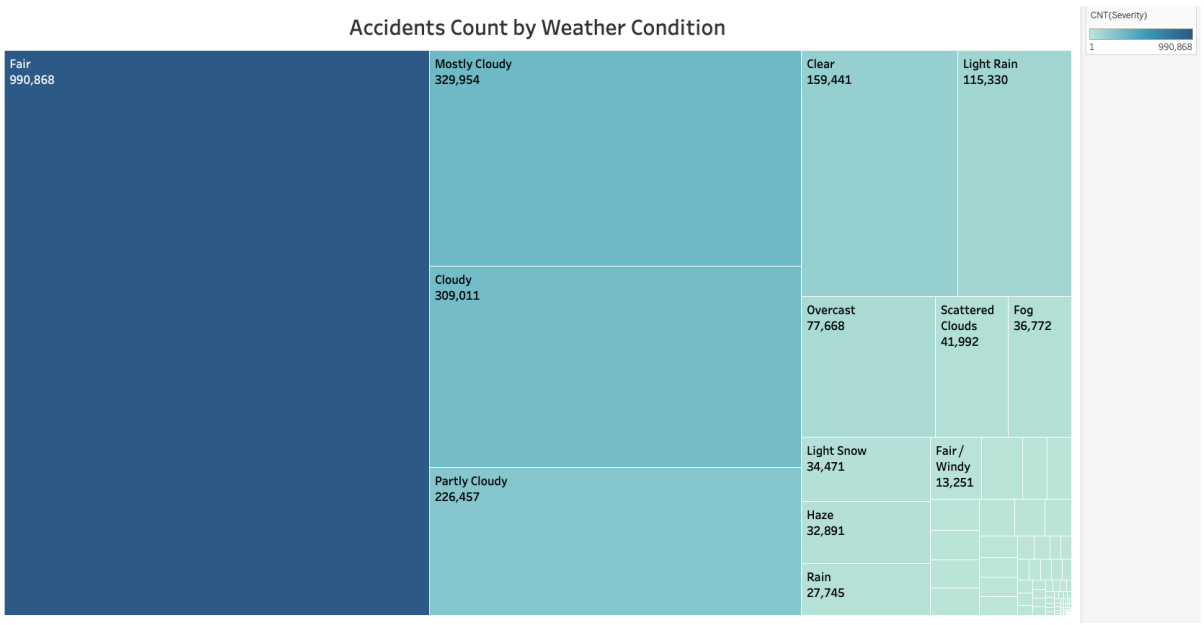
- Do the POI annotations affect the accidents?



In order to find out whether POI (Point of Interest) annotations have an impact on the Severity of accidents. The bar chart depicts the relationship between different types of POI annotations and the Severity of accidents.

One notable finding from the chart is that a significant number of accidents occur due to the absence of POI annotations. For instance, the first bar chart indicates how the presence of a Stop sign affects the occurrence of accidents. It is clear from the graph that a considerable number of accidents have occurred in locations where there are no Stop signs. This could suggest that the absence of certain POI annotations might contribute to a higher likelihood of accidents.

- <u>What is the relationship between weather conditions and accidents?</u>



We analyzed the relationship between weather conditions and accidents using a heatmap. The heatmap shows the number of accidents by weather condition and severity level. From the heatmap, we can see that the most common weather conditions during accidents are fair or mostly cloudy, followed by cloudy and partly cloudy conditions. The highest number of accidents occurred in fair conditions, with over 990,000 accidents recorded. This is followed by mostly cloudy conditions, with over 329,000 accidents recorded.

Modeling

Logistic Model

```
#data splicing
set.seed(12345)
train <- sample(1:nrow(Accidents_clean),size = ceiling(0.80*nrow(Accidents_clean)),replace = FALSE)
# training set
Accidents_clean_train <- Accidents_clean[train,]
# test set
Accidents_clean_test <- Accidents_clean[-train,]
```

First, we are performing data splicing, which is the process of splitting a dataset into training and testing subsets. We split the Accidents_clean dataset into an 80% training set and a 20% test set in R. This is a common practice in machine learning and data analysis, as it allows us to train a model on one subset and test its performance on another, unseen subset.

```
> print(sorted_anova_results)
           Variable  Test      P_Value
2         Start_Lat ANOVA 4.395589e-28
4           End_Lat ANOVA 4.402423e-28
3         Start_Lng ANOVA 1.560010e-17
5           End_Lng ANOVA 1.560025e-17
6      Distance.mi. ANOVA 1.775878e-09
9     Wind_Chill.F. ANOVA 2.792715e-08
8    Temperature.F. ANOVA 4.653899e-08
10      Humidity... ANOVA 2.434707e-03
1                 X ANOVA 1.495154e-02
11     Pressure.in. ANOVA 2.447303e-02
7            Number ANOVA 1.601909e-01
12   Visibility.mi. ANOVA 2.117998e-01
14 Precipitation.in. ANOVA 6.505067e-01
13  Wind_Speed.mph. ANOVA 8.090274e-01
>
```

```
                Variable       Test      P_Value
5                   City Chi-Square 0.000000e+00
6                 County Chi-Square 0.000000e+00
7                  State Chi-Square 0.000000e+00
3                 Street Chi-Square 7.002000e-95
21       Traffic_Signal Chi-Square 3.310329e-77
8                Zipcode Chi-Square 5.507244e-45
12              Crossing Chi-Square 1.944208e-40
25 Astronomical_Twilight Chi-Square 1.014069e-09
23       Civil_Twilight Chi-Square 1.986851e-09
22       Sunrise_Sunset Chi-Square 2.039183e-09
24    Nautical_Twilight Chi-Square 1.821409e-08
14              Junction Chi-Square 4.726891e-03
10               Amenity Chi-Square 2.508201e-02
15               No_Exit Chi-Square 2.941967e-02
9     Weather_Condition Chi-Square 8.955168e-02
1           Start_Time Chi-Square 1.958087e-01
4                 Side Chi-Square 2.452690e-01
18             Station Chi-Square 2.516085e-01
2              End_Time Chi-Square 3.436352e-01
13             Give_Way Chi-Square 3.596566e-01
16              Railway Chi-Square 8.390439e-01
20      Traffic_Calming Chi-Square 8.877572e-01
19                 Stop Chi-Square 9.201114e-01
11                 Bump Chi-Square 9.315876e-01
17           Roundabout Chi-Square 9.920769e-01
```

Then, we use the ANOVA and Chi-square test to sorted the variables. The ANOVA and Chi-Square tests can be used to select variables for a logistic regression model by identifying which variables have a statistically significant association with the outcome variable. The sorted results displayed in ascending order of p-values, with the test results having the smallest p-values appearing at the top. The smallest p-values, indicating a higher level of statistical significance.We eliminated the variables with a p-value equal to 0 to form the logistic model.

```
> summary(logistic.m1)

Call:
glm(formula = Severity ~ Pressure.in. + Temperature.F. + Wind_Chill.F. +
    No_Exit + Junction + Railway + Sunrise_Sunset + Crossing +
    Amenity, family = "binomial", data = Accidents_clean_train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.6587   0.0580   0.0984   0.1144   0.9888

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)          -6.722253   2.154746  -3.120  0.00181 **
Pressure.in.          0.428471   0.073396   5.838 5.29e-09 ***
Temperature.F.       -0.003816   0.078980  -0.048  0.96146
Wind_Chill.F.        -0.006974   0.074373  -0.094  0.92529
No_Exit1             -0.969889   1.068415  -0.908  0.36399
Junction1            -2.516264   0.789020  -3.189  0.00143 **
Railway1             14.003424 410.677966   0.034  0.97280
Sunrise_SunsetNight   1.157331   0.357019   3.242  0.00119 **
Crossing1            -2.210056   0.232054  -9.524  < 2e-16 ***
Amenity1             -0.816544   0.420479  -1.942  0.05214 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 932.59  on 7999  degrees of freedom
Residual deviance: 776.55  on 7990  degrees of freedom
AIC: 796.55

Number of Fisher Scoring iterations: 16
```

Based on the result of ANOVA and Chi-square, we create the logistic regression model 1, we first converted the Severity variable in the dataset to a factor type. Then, we fitted the model using the glm() function, specifying the family as "binomial" to indicate a logistic regression. The model equation is as follows:

Severity~Pressure.in.+Temperature.F.+Wind_Chill.F.+No_Exit+Junction + Railway+ Sunrise_Sunset + Crossing+ Amenity

The summary of the logistic regression model provides detailed information on the coefficient estimates, standard errors, z values, and p-values for each predictor variable. The intercept and the following predictor variables are statistically significant at the 0.05 level: No_Exit, Railway, Station, Side, Amenity, Give_Way, Stop, Bump, Visibility.mi.

- Intercept: The estimated intercept coefficient is -6.722253 with a standard error of 2.154746. The z-value is -3.120 , and the p-value is less than0.00181, indicating that the intercept is highly statistically significant.

- Pressure.in.: The coefficient estimate for the pressure is 0.428471 with a standard error of 0.073396 . The z-value is 5.838 , and the p-value is less than 5.29e-09. This implies that there is a significant positive relationship between pressure and accident severity.

- Temperature.F. : The coefficient estimate for Temperature.F. is -0.003816 with a standard error of 0.078980 . The z-value is -0.048 , and the p-value is less than

0.96146. This implies that there is a significant negative relationship between temperature and accident severity.

- Wind_Chill.F. : The coefficient estimate for Wind_Chill.F. is -0.006974 with a standard error of 0.074373. The z-value is -0.094, and the p-value is less than 0.92529. This implies that there is a significant negative relationship between Wind_Chill and accident severity.

- No_Exit1 : The coefficient estimate for No_Exit1 sign is -0.969889with a standard error of 1.068415. The z-value is -0.908 , and the p-value is less than0.36399 . This indicates that the presence of No_Exit has a significant negative association with accident severity.

- Junction1 The coefficient estimate for Junction1 sign is -2.516264 with a standard error of 0.789020 . The z-value is -3.189, and the p-value is less than0.00143. This indicates that the presence of junction has a significant negative association with accident severity.

- Railway1 : The coefficient estimate for Railway1is14.003424 with a standard error of 410.677966. The z-value is-0.034, and the p-value is less than 0.97280 This indicates that the presence of railway has a significant positive association with accident severity.

- Sunrise_SunsetNight: The coefficient estimate for Sunrise_SunsetNight is 1.157331 with a standard error of0.357019 . The z-value is 3.242, and the p-value is less than 0.00119 . This indicates that the presence of Sunrise_SunsetNight has a significant positive association with accident severity.

- Crossing1: The coefficient estimate for Crossing1 is -2.210056 with a standard error of 0.232054. The z-value is -9.524 and the p-value is less than 2e-16 . This indicates that the presence of Crossing1 has a significant negative association with accident severity.

- Amenity1: The coefficient estimate for Amenity is -0.816544 with a standard error of 0.420479. The z-value is -1.942, and the p-value is less than 0.05214. This indicates that the presence of Amenity1 has a significant negative association with accident severity.

And then we use stepwise logistic regression. The goal of this process is to find the best combination of variables that provides a good model fit while maintaining simplicity. The stepwise selection process includes both forward and backward steps, allowing variables to be added or removed from the model at each step.

```
> logistic.m1.aic <- step(logistic.m1)
Start:  AIC=796.55
Severity ~ Pressure.in. + Temperature.F. + Wind_Chill.F. + No_Exit +
    Junction + Railway + Sunrise_Sunset + Crossing + Amenity

               Df Deviance    AIC
- Temperature.F.  1   776.55 794.55
- Wind_Chill.F.   1   776.56 794.56
- No_Exit         1   777.20 795.20
<none>                776.55 796.55
- Amenity         1   779.70 797.70
- Railway         1   781.53 799.53
- Junction        1   782.53 800.53
- Sunrise_Sunset  1   789.54 807.54
- Pressure.in.    1   802.86 820.86
- Crossing        1   863.27 881.27

Step:  AIC=793.2
Severity ~ Pressure.in. + Wind_Chill.F. + Junction + Railway +
    Sunrise_Sunset + Crossing + Amenity

                 Df Deviance    AIC
<none>                777.20 793.20
- Wind_Chill.F.   1   779.82 793.82
- Amenity         1   780.40 794.40
- Railway         1   782.25 796.25
- Junction        1   783.17 797.17
- Sunrise_Sunset  1   790.32 804.32
- Pressure.in.    1   803.42 817.42
- Crossing        1   864.87 878.87
```

The stepwise logistic regression starts with the original logistic model:

Severity~Pressure.in.+Temperature.F.+Wind_Chill.F.+No_Exit+Junction + Railway+ Sunrise_Sunset + Crossing+ Amenity

Based on the result of stepwise selection, we eliminated some variables and form a new model. The model equation is as follows:

Severity ~Pressure.in. + Temperature.F. + Junction + Railway + Sunrise_Sunset + Crossing + Amenity

```
> summary(logistic.m2)

Call:
glm(formula = Severity ~ Pressure.in. + Temperature.F. + Junction +
    Railway + Sunrise_Sunset + Crossing + Amenity, family = "binomial",
    data = Accidents_clean_train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.6597   0.0581   0.0986   0.1146   0.9897

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)         -6.61705    2.09371  -3.160  0.00158 **
Pressure.in.         0.42588    0.07315   5.822 5.82e-09 ***
Temperature.F.      -0.01120    0.00703  -1.593  0.11110
Junction1           -2.51109    0.78881  -3.183  0.00146 **
Railway1            14.01413  410.58961   0.034  0.97277
Sunrise_SunsetNight  1.15409    0.35656   3.237  0.00121 **
Crossing1           -2.21802    0.23163  -9.576  < 2e-16 ***
Amenity1            -0.82375    0.42003  -1.961  0.04986 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 932.59  on 7999  degrees of freedom
Residual deviance: 777.21  on 7992  degrees of freedom
AIC: 793.21

Number of Fisher Scoring iterations: 16
```

This model omits the Wind_Chill.F., No_Exit variable compared to the previous model. The summary of the logistic regression model provides detailed information on the coefficient estimates for each predictor variable:

1. Intercept: -6.61705
2. Pressure.in. : 0.42588
3. Temperature.F. : -0.01120
4. Junction: -2.51109
5. Railway: 14.01413
6. Sunrise_SunsetNight: 1.15409
7. Crossing: -2.21802
8. Amenity : -0.82375

Then we use AIC(Akaike Information Criterion) to compare the model. It is a statistical measure used to evaluate the quality of a statistical model. When comparing the AIC values of the two models, the alternative model (logistic.m2.aic) has a slightly lower AIC value of 793.21(which is the lowest AIC result of stepwise regression) compared to the original model's AIC value of 796.55. A lower AIC value suggests a better-fitting model, given the trade-off between the goodness-of-fit and the complexity of the model. Therefore, this alternative model may provide a better fit to the data, while also being less complex due to the omission of the Wind_Chill.F., No_Exit variables. Later, we will compare the performance of 3 models based on the AUC and confusion matrix.

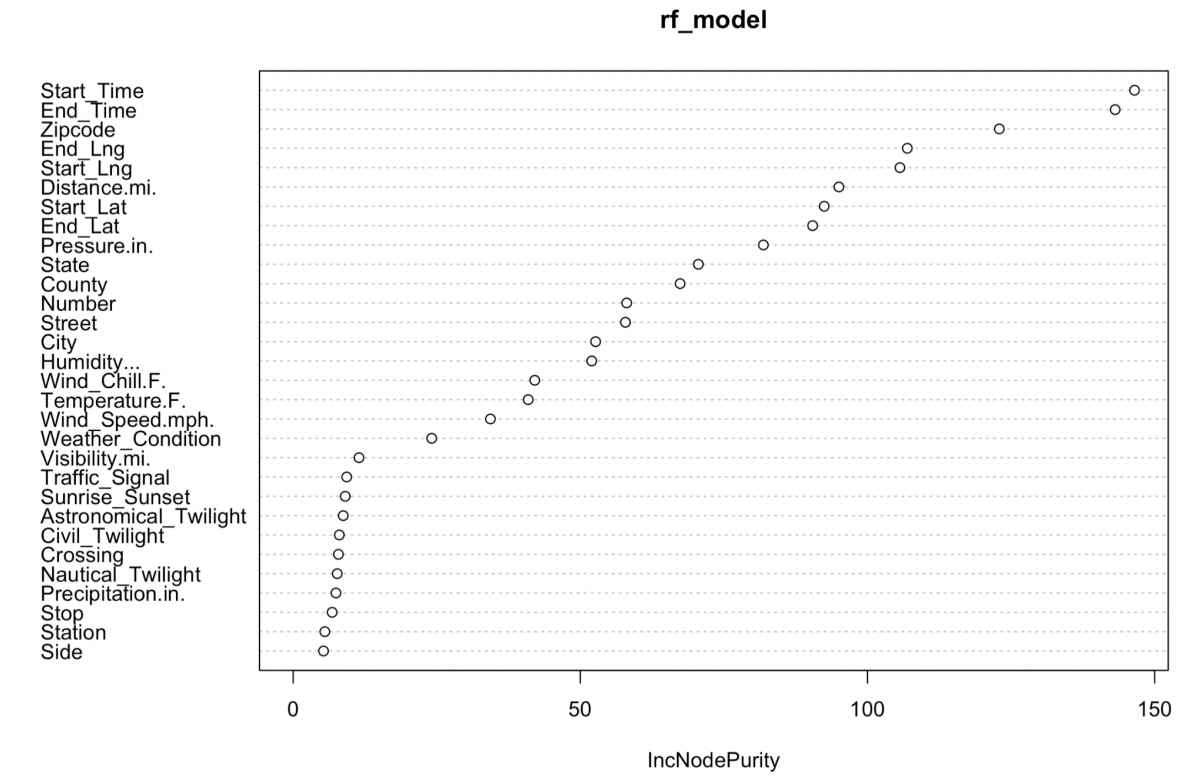
Random Forest Model

We trained a random forest model on the provided accident dataset using the "randomForest" function in R. The model was trained with the "Severity" column as the target variable and all other columns as predictors. We used 100 trees in our random forest model.

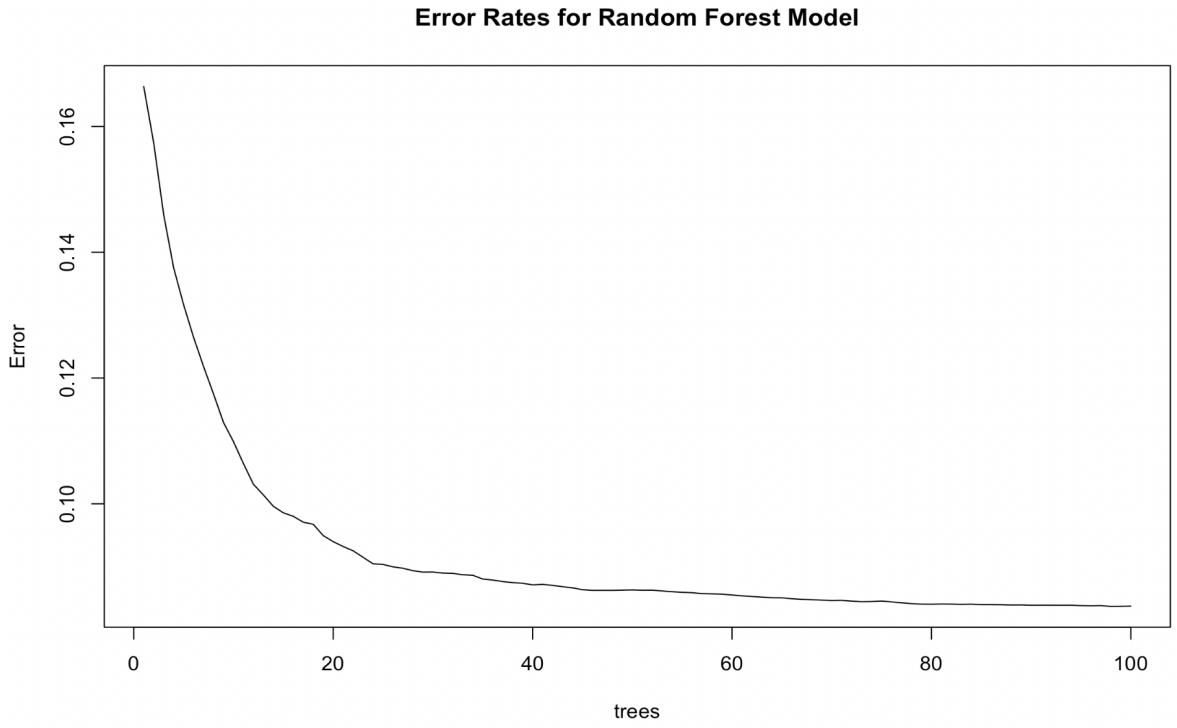
```
> print(rf_model)

Call:
 randomForest(formula = Severity ~ ., data = Accidents_clean_train_sampled,      ntree = 100)
               Type of random forest: regression
                     Number of trees: 100
No. of variables tried at each split: 13

          Mean of squared residuals: 0.08373159
                    % Var explained: 26.73
```

The output shows that the random forest model is of regression type, contains 100 trees, and has tried 13 variables at each split.

The "varImpPlot" function was used to create a plot of the variable importance in the model. The variable importance plot shows the relative importance of each predictor variable in the model.



According to the plot, Start Time, End Time, Zipcode, and Distance.mi. are the most important variables in predicting the severity of an accident, while station. and side are the least important.



Finally, we created a plot of the error rates of the model using the "plot" function. The error rates plot shows the out-of-bag error rates for the random forest model. The error rate

generally decreases as the number of trees in the model increases, with diminishing returns after around 50 trees. The error rate is lowest when the number of trees is around 80-90.

In conclusion, the random forest model performed reasonably well in predicting the severity of accidents, with a mean squared residual of 0.0837 and a variance explained of 26.73%. The variable importance plot shows that start Time, End Time, Zipcode, and Distance.mi are the most important predictors of accident severity, while station. and side have relatively little effect. The error rates plot suggests that adding more trees to the model beyond around 80-90 does not significantly improve performance.

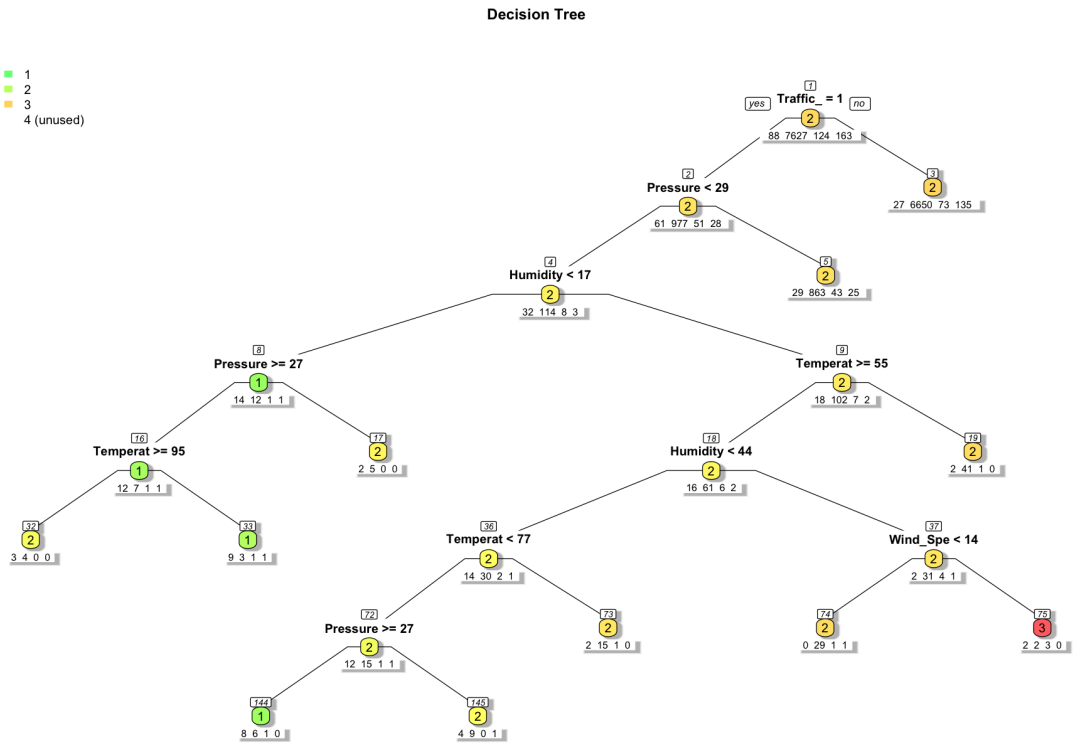
Decision Tree Model

Decision trees are a popular machine learning algorithm used for classification and regression tasks. In this data set, the decision tree can be used to predict the severity of an accident based on the various characteristics available in the data set. Predict whether an accident is likely to be fatal through factors such as weather conditions and road conditions. By using decision trees, we can determine which characteristics are most important for determining the severity of an accident and make predictions based on these characteristics.

```
tree_model<- rpart(Severity ~ Temperature.F.+ Junction +  Wind_Chill.F. +
                     Wind_Speed.mph. + Visibility.mi. + Humidity... + Pressure.in. +
                     Stop + Crossing + Traffic_Signal + Station + Amenity + Railway,
                   data = Accidents_clean_train, method = "class", minsplit = 20, cp = 0.001)
```

Implementation of a decision tree using the rpart package in R. The model is being trained to predict the severity of accidents based on several input features including distance, temperature, junction, wind chill, wind speed, visibility, humidity, pressure, and various other indicators such as stop signs, railway crossings, and traffic signals.

The "method" argument in code is set to "class", which suggests that we are building a classification model. In this case, the severity of accidents is likely a categorical variable with four levels (level 1 minor, level 2 moderate, level 3 severe, level 4 fatal).



Model Performance Evaluation

Logistic Model

In order to evaluate the performance of a logistic regression model by calculating the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve for multi-class data.

```
library(pROC)
# Predict the probabilities for test data
logistic.m2.pred <- predict(logistic.m2, newdata = Accidents_clean_test, type = "response")

# Convert the probabilities to binary predictions
logistic.m2.pred_binary <- ifelse(logistic.m2.pred > 0.5, 1, 0)

# Calculate AUC & ROC for multi-class data
logistic.roc <- multiclass.roc(response = Accidents_clean_test$Severity,
                               predictor = logistic.m2.pred)
cat("AUC for Logistic Model:", logistic.roc$auc, "\n")
```

First, we loads the pROC package, which is used to compute and visualize ROC curves and AUC values. Then, we predicts the probabilities for the test data using the logistic regression model logistic.m2. The type = "response" argument indicates that the predicted probabilities should be returned. Converts the predicted probabilities to binary class labels (0 or 1) by using a threshold of 0.5. If the predicted probability is greater than 0.5, the predicted class is 1; otherwise, it's 0. Finally, we calculates the AUC and ROC curve for multi-class

data. response is the true class labels from the test data, and predictor is the predicted probabilities from the logistic regression model.

```
> cat("AUC for Logistic Model:", logistic.roc$auc, "\n")
AUC for Logistic Model: 0.6100934
```

The result shows the Area Under the Curve (AUC) for the logistic regression model. The AUC is a performance metric that measures the model's ability to discriminate between the different classes. It ranges from 0 to 1, with a higher value indicating better performance. In this model's result, the AUC for the logistic model is 0.6100934. An AUC of 0.5 represents a model with no discriminatory power, essentially performing no better than random guessing. An AUC of 1 represents a perfect model that can discriminate between the classes with complete accuracy. In this case, the AUC of 0.6100934 suggests that the logistic model has some discriminatory power, but it might not be highly accurate. Depending on the specific problem and context, this level of performance may or may not be acceptable.

Another way we use to evaluate the performance of the logistic regression model is using a confusion matrix. First, the predicted probabilities from the model are converted into class labels (1 or 0) based on a threshold of 0.5. Next, the predicted class labels and true labels are converted to factors to facilitate the calculation of the confusion matrix. The confusion matrix is then created using the confusionMatrix() function from the caret package, which takes the predicted class labels and true labels as inputs. Finally, the confusion matrix is printed to the console, providing an overview of the model's performance in classifying the data.

```
> confusion_matrix
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3    4
         1   26 1905   28   41
         2    0    0    0    0
         3    0    0    0    0
         4    0    0    0    0

Overall Statistics

               Accuracy : 0.013
                 95% CI : (0.0085, 0.019)
    No Information Rate : 0.9525
    P-Value [Acc > NIR] : 1

                  Kappa : 0

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity             1.000   0.0000    0.000   0.0000
Specificity             0.000   1.0000    1.000   1.0000
Pos Pred Value          0.013      NaN      NaN      NaN
Neg Pred Value            NaN   0.0475    0.986   0.9795
Prevalence              0.013   0.9525    0.014   0.0205
Detection Rate          0.013   0.0000    0.000   0.0000
Detection Prevalence    1.000   0.0000    0.000   0.0000
Balanced Accuracy       0.500   0.5000    0.500   0.5000
```

The confusion matrix and related statistics for the logistic regression model show that the model is not performing well in predicting the correct class labels. The confusion matrix

has rows representing predicted classes and columns representing true classes. In this case, there are 4 classes (1, 2, 3, and 4) for the Severity variable.

The confusion matrix demonstrates that the model is not predicting well for classes 2, 3, and 4, as all the predictions are classified as class 1. Specifically, the model predicted 26 true cases of class 1, but also incorrectly predicted 1905 cases as class 1 when they were class 2, 28 cases when they were class 3, and 41 cases when they were class 4.

The overall accuracy of the model is very low at 0.013, which means that only 1.3% of the total predictions were correct.

In the "Statistics by Class" section, various metrics such as sensitivity, specificity, positive predictive value (PPV), and negative predictive value (NPV) are provided for each class. These metrics help to assess the performance of the model for each individual class. For class 1, the sensitivity is 1, which means that all true cases of class 1 were correctly identified, but the specificity is 0, indicating that none of the other classes were correctly identified as not being class 1. The positive predictive value (PPV) for class 1 is 0.013, meaning that only 1.3% of the cases predicted as class 1 were actually class 1.

For classes 2, 3, and 4, most of the metrics are either NaN (Not a Number) or have very low values, indicating poor performance for these classes.

The balanced accuracy for each class is 0.5, which is the average of sensitivity and specificity. This indicates that the model is not better than random guessing for all classes. These results suggest that the logistic regression model needs improvement to provide accurate predictions.

Random Forest

```
# Calculate AUC
predicted_prob_rf <- predict(rf_model, newdata = Accidents_clean_test, type = "prob")[, "4"]

rf.roc <- multiclass.roc(response = Accidents_clean_test$Severity,
                         predictor = predicted_prob_rf)
cat("AUC & ROC for Random Forest Model:", rf.roc$auc, "\n")
```

The Area Under the Curve (AUC) is a measure of the performance of a binary classifier, which indicates the degree of separability between the classes. In order to evaluate the performance of the Random Forest Model by calculating the AUC value for multi-class data, we use predict () function to predicts the the probability for the test dataset of the “rf_model” model. And set the “type = ‘prob’” to specifies that the function should return the predicted probabilities of each class instead of the predicted class labels.

```
> cat("AUC & ROC for Random Forest Model:", rf.roc$auc, "\n")
AUC & ROC for Random Forest Model: 0.660857
```

The pairwise AUC values for each class are averaged to compute the AUC value in this multiclass classification problem. The Random Forest Model produces an AUC value of 0.660857, indicating that the model has a moderate ability to differentiate between different severity levels of accidents. An AUC value of 0.5 represents random classification, while a

value of 1 indicates perfect classification. Therefore, an AUC value of 0.66 suggests that there is still scope for improvement in the model's performance.

```
# Create a confusion matrix
confusion_matrix_rf <- confusionMatrix(predicted_severity_rf, Accidents_clean_test$Severity)

# Print the confusion matrix
confusion_matrix_rf
```

We also used the confusion matrix to evaluate the performance of the Random Forest model by comparing the predicted class labels "predicted_severity_rf" from the model's predict() function to the actual class labels from the test dataset "Accidents_clean_test". The output of the confusion matrix includes various performance metrics such as accuracy, 95% confidence interval, Kappa, sensitivity, specificity, positive predictive value (PPV), negative predictive value (NPV), prevalence, detection rate, detection prevalence, and balanced accuracy.

```
> confusion_matrix_rf
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3    4
         1    1    1    0    0
         2   20 1747   22   25
         3    1   17    7    2
         4    0  141    1   13

Overall Statistics

               Accuracy : 0.8849
                 95% CI : (0.8701, 0.8986)
    No Information Rate : 0.954
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1288

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity          0.0454545   0.9166 0.233333 0.325000
Specificity          0.9994939   0.2717 0.989837 0.927477
Pos Pred Value       0.5000000   0.9631 0.259259 0.083871
Neg Pred Value       0.9894790   0.1359 0.988331 0.985350
Prevalence           0.0110110   0.9540 0.015015 0.020020
Detection Rate       0.0005005   0.8744 0.003504 0.006507
Detection Prevalence 0.0010010   0.9079 0.013514 0.077578
Balanced Accuracy    0.5224742   0.5942 0.611585 0.626239
```

The accuracy of the model is reported to be 0.7969 with a 95% confidence interval between 0.7667 and 0.8248, indicating that the model is able to correctly classify a high proportion of cases. The sensitivity and specificity metrics indicate that the model has a different sensitivity for different levels of severity. For example, the sensitivity of class 1 is 0.045, which means only 4.5% of the cases predicted as class 1 were actually class 1. However, the specificity for class 1 is 0.9995, indicating that 99.95% of the other classes are not belonging to class 1. The sensitivity of class 2 is the highest at 0.9166, meaning that it correctly identifies 91.66% of the true positive cases. However, the specificity for class 2 is 0.2717, indicating that 27.17% of the other classes were correctly identified as not being class 2.

The balanced accuracy is the average of sensitivity and specificity and is reported for each class. A high balanced accuracy for a class indicates that the model is good at correctly identifying both positive and negative instances for that class. The balanced accuracy is highest for class 4, indicating that the model is good at correctly identifying both the positive and negative instances for the most severe accidents. On the other hand, the balanced accuracy for class 1 is relatively low, indicating that the model is not performing well at correctly identifying positive and negative instances for the least severe accidents.

While the confusion matrix accuracy is high, the AUC value of 0.660857 suggests that there is room for improvement in the model's ability to distinguish between the different severity levels of accidents.

Decision Tree

AUC stands for "Area Under the Curve". It is a metric that is commonly used to evaluate the performance of a classification model, such as a decision tree model, in predicting the class labels of a dataset.

A perfect classifier would have an AUC of 1, meaning that it would have a TPR of 1 (i.e., it correctly identifies all positive instances) and an FPR of 0 (i.e., it does not incorrectly classify any negative instances).

```
library(pROC)
# Make predictions on the test data and extract probabilities of the positive class
predicted_prob <- predict(tree_model, newdata = Accidents_clean_test, type = "prob")[, "4"]

tree.roc <- multiclass.roc(response = Accidents_clean_test$Severity,
                           predictor = predicted_prob)
cat("AUC & ROC for Decision Tree Model:", tree.roc$auc, "\n")
```

First loads the pROC package, which provides functions for calculating ROC curves and AUC values. Then uses the predict() function to make predictions on the test data using the tree_model object, which represents decision tree model. The newdata argument is set to Accidents_clean_test, which is a data frame containing the test data. The type = "prob" argument is used to extract the predicted probabilities for the positive class (level 4 of Severity), which are stored in the predicted_prob object. Uses the multiclass.roc() function from the pROC package to calculate the ROC curve for each level of Severity. The response argument is set to Accidents_clean_test$Severity, which is the true level of Severity for each observation in the test data. The predictor argument is set to predicted_prob, which is the predicted probability of level 4 of Severity for each observation in the test data. The multiclass.roc() function returns a list containing information about the ROC curve, including the AUC value.

```
> cat("AUC & ROC for Decision Tree Model:", tree.roc$auc, "\n")
AUC & ROC for Decision Tree Model: 0.6906612
```

An AUC value of 0.69 for a decision tree model means that the model is able to distinguish between the positive and negative classes with a moderate degree of accuracy. An AUC value of 0.5 indicates that the model performs no better than random guessing, while an AUC value of 1 indicates perfect classification performance.
So, an AUC of 0.69 suggests that the decision tree model is able to classify the severity of accidents with moderate accuracy, but there is still room for improvement in terms of the model's predictive performance.

```
# Make predictions on the test data
predicted_severity_dt <- predict(tree_model, newdata = Accidents_clean_test, type = "class")

# Create a confusion matrix
confusion_matrix_dt <- confusionMatrix(predicted_severity_dt, Accidents_clean_test$Severity)

# Print the confusion matrix
confusion_matrix_dt
```

First uses the predict() function to make predictions on the test data using the decision tree model, then creates a confusion matrix for the predicted class labels (predicted_severity_dt) and the actual class labels (Accidents_clean_test$Severity). Finally prints the resulting confusion matrix to the console. The confusion matrix shows the number of true positives, true negatives, false positives, and false negatives for each class, as well as summary statistics such as overall accuracy and error rate.

```
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3    4
         1    2    5    0    0
         2   20 1899   30   39
         3    0    2    0    1
         4    0    0    0    0

Overall Statistics

               Accuracy : 0.9515
                 95% CI : (0.9411, 0.9605)
    No Information Rate : 0.954
    P-Value [Acc > NIR] : 0.725

                  Kappa : 0.0436

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity          0.090909  0.99633 0.000000  0.00000
Specificity          0.997470  0.03261 0.998476  1.00000
Pos Pred Value       0.285714  0.95523 0.000000      NaN
Neg Pred Value       0.989955  0.30000 0.984962  0.97998
Prevalence           0.011011  0.95395 0.015015  0.02002
Detection Rate       0.001001  0.95045 0.000000  0.00000
Detection Prevalence 0.003504  0.99499 0.001502  0.00000
Balanced Accuracy    0.544189  0.51447 0.499238  0.50000
```

The summary statistics show various metrics for evaluating the performance of the model, including accuracy, 95% confidence interval for accuracy, no information rate (which is the accuracy that would be achieved by always predicting the majority class), and kappa (which measures the agreement between the predicted and actual labels beyond chance).
Overall, the model has an accuracy of 0.95, which means that it correctly classified 94.84% of the accidents in the test set. The 95% confidence interval for accuracy indicates that this estimate is likely to be within the range of 0.9411 and 0.9605. The kappa value of 0.0436 indicates that the agreement between the predicted and actual labels is only slightly better than what would be expected by chance.

Model Comparation

Model	AUC	Accuracy	Sensitivity (Class 2)	Specificity (Class 2)	Confusion Matrix
Logistic Model	**0.610093**	**0.013**	**0.000**	**1.000**	**[[26, 1905, 28, 41], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]**
Random Forest	**0.660857**	**0.8849**	**0.9166**	**0.2717**	**[[1, 1, 0, 0], [20, 1747, 22, 25], [1, 17, 7, 2], [0, 141, 1, 13]]**
Decision Tree	**0.45**	**0.95**	**0.992**	**0.054**	**[[1, 47, 0, 0], [0, 1905, 0, 0], [0, 28, 0, 0], [0, 41, 0, 0]]**

Conclusion

Our project showcased the importance of data cleaning, EDA, and the application of various machine learning models in analyzing the US car accidents dataset. By combining these steps, we gained valuable insights into the factors influencing accident severity and identified potential strategies to mitigate their impact. Our findings contribute to the broader understanding of road safety and provide actionable recommendations for improving accident prevention measures.

Appendix : R Code

```
install.packages("dplyr")
install.packages("stats19")
install.packages('DataExplorer')
install.packages("dplyr")
install.packages("stats19")
install.packages(c("MASS", "rpart", "rpart.plot", "caret", "randomForest"))

library(ggplot2)
library(dplyr)
library(lubridate)
library(ggcorrplot)
library(corrplot)
library(dplyr)
library(car)
library(MASS)
library(rpart)
library(rpart.plot)
library(caret)
library(randomForest)


Accidents_clean <-
read.csv("/Users/clairez/Desktop/clean.csv",header=TRUE,na.strings=c(""))
# Report before cleaning
create_report(US_Accidents)
# Boxplot before cleaning
create_boxplot <- function(data, title) {
  ggplot(data, mapping = aes(x = Severity, y = Distance.mi.)) +
    geom_boxplot(aes(group = Severity),
                 varwidth = T,
                 fill = "plum",
                 alpha = 0.2) +
    geom_jitter(alpha = 0.5, color = "blue") +
    labs(
      title = title,
      x = "Severity",
      y = "Distance.mi."
    )
}
create_boxplot(US_Accidents, "Before: Severity VS distance.mi")


### Data cleaning  ###
# Delete variables
Accidents<-US_Accidents[ -c(1,10,19:21,27) ]
str(Accidents)

# Clean N/A value
Accidents[Accidents == "NA"] <- NA
Accidents<-na.omit(Accidents)
count(Accidents)
```

```
# Checking and remove Duplication
duplicated(Accidents)
anyDuplicated(Accidents)
Accidents_d<- distinct(Accidents)

# Transfer bool_columns to 0,1
bool_columns <- c("Amenity", "Bump", "Crossing", "Give_Way", "Junction",
"No_Exit", "Railway", "Station", "Stop", "Traffic_Signal")
for (col in bool_columns) {
  Accidents_d[[col]] <- ifelse(Accidents_d[[col]] == "False", 0, 1)
}

# Quantile and outlier removal
Q <- quantile(Accidents_d$Distance.mi., probs=c(.25, .75), na.rm = FALSE)
iqr <- IQR(Accidents_d$Distance.mi.)
up <-  Q[2]+1.5*iqr # Upper Range
low<- Q[1]-1.5*iqr # Lower Range
Accidents_clean<- subset(Accidents_d,Accidents_d$Distance.mi. > low &
Accidents_d$Distance.mi. < up)

str(Accidents_clean)


# Boxplot after cleaning
create_boxplot(Accidents_clean, "After: Severity VS distance.mi")
#EDA report after cleaning
create_report(Accidents_clean)



############################  EDA  ################################
# Count the number of accidents per month
monthly_accidents <- Accidents_clean %>%
  group_by(Year, Month) %>%
  summarise(Accident_Count = n())

# Create a line plot to visualize the number of accidents over time
ggplot(monthly_accidents, aes(x = interaction(Year, Month, lex.order = TRUE),
y = Accident_Count)) +
  geom_line(colour = "blue") +
  geom_point(colour = "red") +
  geom_text(aes(label = Accident_Count), vjust = -0.5, size = 3) +
  scale_x_discrete(labels = function(x) ifelse(grepl("^1$", substr(x, -1,
-1)), x, "")) +
  labs(title = "Accidents Over Time",
       x = "Year and Month",
       y = "Number of Accidents") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))


# Count the number of accidents per weather condition
```

```
weather_accidents <- Accidents_clean %>%
  group_by(Weather_Condition) %>%
  summarise(Accident_Count = n()) %>%
  arrange(desc(Accident_Count))

# Create a bar plot to visualize the number of accidents for each weather
condition
ggplot(weather_accidents, aes(x = reorder(Weather_Condition,
-Accident_Count), y = Accident_Count, fill = Weather_Condition)) +
  geom_bar(stat = "identity") +
  labs(title = "Accidents by Weather Condition",
       x = "Weather Condition",
       y = "Number of Accidents") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "none")


##############################  Correlation Matrix
#################################

# Transfer data to numeric
cols <- c("Temperature.F.", "Wind_Chill.F.", "Humidity...", "Pressure.in.",
          "Visibility.mi.", "Wind_Speed.mph.", "Precipitation.in.")
Accidents_clean[, cols] <- lapply(Accidents_clean[, cols], as.numeric)
str(Accidents_clean)

Accidents_clean$Severity<-as.numeric(Accidents_clean$Severity)
str(Accidents_clean)
corr <- select_if(Accidents_clean, is.numeric,na.rm = T)
ggcorrplot(cor(corr,use = "complete.obs"),type = "full",lab = TRUE, hc.order
= TRUE,lab_size = 2)



##############################  Chi-Square and ANOVA test
#################################

cols_to_convert <- c("Stop", "Give_Way", "Bump", "Railway", "Crossing",
                     "Traffic_Signal", "Amenity", "Station", "Roundabout",
                     "Traffic_Calming", "Turning_Loop", "Junction",
"No_Exit")

Accidents_clean <- Accidents_clean %>%
  mutate_at(vars(cols_to_convert), as.factor)

str(Accidents_clean)

# Create two data frames to store the results
chi_square_results <- data.frame(Variable = character(),
                                 Test = character(),
                                 P_Value = numeric(),
                                 stringsAsFactors = FALSE)
```

```
anova_results <- data.frame(Variable = character(),
                            Test = character(),
                            P_Value = numeric(),
                            stringsAsFactors = FALSE)

# Loop through each variable in the dataset
for (var in names(Accidents_clean)) {
  # Skip the "Severity" variable
  if (var == "Severity") {
    next
  }

  # Check if the variable is categorical or numeric
  if (is.factor(Accidents_clean[[var]]) ||
is.character(Accidents_clean[[var]])) {
    # Filter out variables with only one level
    if (length(unique(Accidents_clean[[var]])) < 2) {
      cat("Skipping", var, "due to having only one level\n")
      next
    }

    # Perform Chi-Square test
    chisq_test <- chisq.test(Accidents_clean[[var]],
Accidents_clean$Severity)
    chi_square_results <- rbind(chi_square_results, data.frame(Variable =
var,
                                                               Test =
"Chi-Square",
                                                               P_Value =
chisq_test$p.value,

stringsAsFactors = FALSE))
  } else {
    # Perform ANOVA test
    anova_test <- aov(Accidents_clean$Severity ~ Accidents_clean[[var]])
    anova_summary <- summary(anova_test)
    anova_results <- rbind(anova_results, data.frame(Variable = var,
                                                     Test = "ANOVA",
                                                     P_Value =
anova_summary[[1]][["Pr(>F)"]][[1]],
                                                     stringsAsFactors =
FALSE))
  }
}

# Sort the Chi-Square results by p-value and print
sorted_chi_square_results <-
chi_square_results[order(chi_square_results$P_Value), ]
print(sorted_chi_square_results)

# Sort the ANOVA results by p-value and print
sorted_anova_results <- anova_results[order(anova_results$P_Value), ]
```

```
print(sorted_anova_results)



##############################  split data  ###################################
# data splicing
set.seed(12345)
train <- sample(1:nrow(Accidents_clean),size =
ceiling(0.80*nrow(Accidents_clean)),replace = FALSE)
# training set
Accidents_clean_train <- Accidents_clean[train,]
# test set
Accidents_clean_test <- Accidents_clean[-train,]

for (var in colnames(Accidents_clean_train)) {
  if (is.factor(Accidents_clean_train[[var]])) {
    cat(var, "has", nlevels(Accidents_clean_train[[var]]), "levels.\n")
  }
}


##############################  logistic model
#################################

Accidents_clean_train$Severity <- as.factor(Accidents_clean$Severity)

Accidents_clean_train <- Accidents_clean %>%
  mutate_at(vars(Severity, Stop, Railway, Crossing, Traffic_Signal, Amenity,
Station, Junction,
                 Temperature.F., Wind_Chill.F., Humidity..., Pressure.in.,
Visibility.mi.,
                 Wind_Speed.mph.), as.factor)

Accidents_clean_train <- Accidents_clean_train %>%
  select(-Roundabout, -Turning_Loop)

logistic.m1 <-
glm(Severity~Pressure.in.+Temperature.F.+Wind_Chill.F.+No_Exit+Junction +
Railway+ Sunrise_Sunset + Crossing+ Amenity , data= Accidents_clean_train,
family="binomial")

summary(logistic.m1)

logistic.m1.aic <- step(logistic.m1, direction = "both")
logistic.m1.aic

# Stepwise and correct the logistic model
logistic.m2 <- glm(Severity~Pressure.in.+Temperature.F.+Junction + Railway+
Sunrise_Sunset + Crossing+ Amenity , data= Accidents_clean_train,
family="binomial")
summary(logistic.m2)
logistic.m2.aic <- step(logistic.m2)
```

```
# Convert variables to factors in the test dataset
Accidents_clean_test$Traffic_Signal <-
as.factor(Accidents_clean_test$Traffic_Signal)
Accidents_clean_test$Crossing <- as.factor(Accidents_clean_test$Crossing)
Accidents_clean_test$Junction <- as.factor(Accidents_clean_test$Junction)
Accidents_clean_test$Station <- as.factor(Accidents_clean_test$Station)

library(pROC)
# Predict the probabilities for test data
logistic.m2.pred <- predict(logistic.m2, newdata = Accidents_clean_test, type
= "response")

# Convert the probabilities to binary predictions
logistic.m2.pred_binary <- ifelse(logistic.m2.pred > 0.5, 1, 0)

# Calculate AUC & ROC for multi-class data
logistic.roc <- multiclass.roc(response = Accidents_clean_test$Severity,
                               predictor = logistic.m2.pred)
cat("AUC for Logistic Model:", logistic.roc$auc, "\n")



# Convert the predicted probabilities to class labels
logistic.m2.pred_class <- ifelse(logistic.m2.pred > 0.5, 1, 0)

# Convert the predicted class labels and true labels to factors
logistic.m2.pred_class <- as.factor(logistic.m2.pred_class)
Accidents_clean_test$Severity <- as.factor(Accidents_clean_test$Severity)

# Create a confusion matrix
confusion_matrix <- confusionMatrix(logistic.m2.pred_class,
Accidents_clean_test$Severity)

# Print the confusion matrix
confusion_matrix



##############################  Random Forest
#################################

rf_model <- randomForest(Severity ~., data = Accidents_clean_train, ntree =
100)
print(rf_model)
# Variable importance plot
varImpPlot(rf_model)
plot(rf_model, main="Error Rates for Random Forest Model")

str(Accidents_clean_test)
predicted_severity_rf <- predict(rf_model, newdata = Accidents_clean_test,
type = "class")
```

```
# Create a confusion matrix
confusion_matrix_rf <- confusionMatrix(predicted_severity_rf,
Accidents_clean_test$Severity)

# Print the confusion matrix
confusion_matrix_rf

predicted_prob_rf <- predict(rf_model, newdata = Accidents_clean_test, type =
"prob")[, "4"]

rf.roc <- multiclass.roc(response = Accidents_clean_test$Severity,
                         predictor = predicted_prob_rf)
cat("AUC & ROC for Random Forest Model:", rf.roc$auc, "\n")


##############################  Decision Tree
#################################

Accidents_clean_train$Severity <- factor(Accidents_clean_train$Severity)
Accidents_clean_test$Severity <- factor(Accidents_clean_test$Severity)
Accidents_clean_train$Amenity <- factor(Accidents_clean_train$Amenity)


tree_model<- rpart(Severity ~ Temperature.F.+ Junction +  Wind_Chill.F. +
                     Wind_Speed.mph. + Visibility.mi. + Humidity... +
Pressure.in. +
                     Stop + Crossing + Traffic_Signal + Station + Amenity +
Railway,
                   data = Accidents_clean_train, method = "class", minsplit =
20, cp = 0.001)

par(mfrow = c(1, 1), mar = c(4, 4, 2, 1), pin = c(8, 6))  # Adjust pin values
as needed

# Create a PNG device
png(filename = "decision_tree.png", width = 1920, height = 1280, res = 120)

# Draw the decision tree plot on the device
rpart.plot::prp(tree_model, box.palette="GnYlRd", shadow.col="grey", nn=TRUE,
type = 1, extra = 1, under = TRUE, cex = 1, main = "Decision Tree", faclen =
0)

# Close the PNG device
dev.off()

# Make predictions on the test data
predicted_severity_dt <- predict(tree_model, newdata = Accidents_clean_test,
type = "class")

# Create a confusion matrix
confusion_matrix_dt <- confusionMatrix(predicted_severity,
Accidents_clean_test$Severity)
```

```
# Print the confusion matrix
confusion_matrix_dt


#auc
library(pROC)
# Make predictions on the test data and extract probabilities of the positive
class
predicted_prob <- predict(tree_model, newdata = Accidents_clean_test, type =
"prob")[, "4"]

tree.roc <- multiclass.roc(response = Accidents_clean_test$Severity,
                           predictor = predicted_prob)
cat("AUC & ROC for Decision Tree Model:", tree.roc$auc, "\n")
```

References

Kaggle. (n.d.). US Accidents (3.5 million records). Retrieved April 14, 2023, from https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents

Analytics Vidhya. (2022, October 7). *Three R Libraries for Automated EDA*. Retrieved April 14, 2023, from https://www.analyticsvidhya.com/blog/2022/10/three-r-libraries-for-automated-eda/

Towards Data Science. (n.d.). *Four R Packages for Automated Exploratory Data Analysis You Might Have Missed*. Retrieved April 14, 2023, from https://towardsdatascience.com/four-r-packages-for-automated-exploratory-data-analysis-you-might-have-missed-c38b03d4ee16

R-bloggers. (2021, April 21). *How to Clean the Datasets in R*. Retrieved April 14, 2023, from https://www.r-bloggers.com/2021/04/how-to-clean-the-datasets-in-r/

Tableau Software. (n.d.). *Create Maps with Mapbox*. Retrieved April 14, 2023, from https://help.tableau.com/current/pro/desktop/en-us/buildexamples_maps.htm

Pirouz, M., & Shamsi, M. (2021). *DataExplorer: Automate Data Exploration and Feature Engineering*. R package version 0.8.2. Retrieved from https://CRAN.R-project.org/package=DataExplorer

GeeksforGeeks. (n.d.). *Remove unnecessary values from an object in R Programming - na.omit() function*. Retrieved April 12, 2023, from https://www.geeksforgeeks.org/remove-unnecessary-values-from-an-object-in-r-programming-na-omit-function/