Spring 2020

Predicting Traffic Accident Severity Levels

COM SCI X 450.4 Michael Chang
Ada Guan, Ruohan Dang, Kyle Lee

Table of Contents

Machine Learning Problem Description

The impact of traffic accidents is at large economically, costing hundreds of billions of dollars every year. Reduction of serious accidents can be quite a challenge. Understanding patterns of severe accidents will allow us to take on a proactive approach in implementing actions to alleviate financial burden and improve traffic efficiency. The approach will be a focus on keeping road conditions safe, the effectiveness will rely on accident severity prediction in relation to influencing factors.

Predicting the criticality of traffic accidents will be a classification problem. The dataset will be categorized into different classes based on different parameters and the output variable of the classification algorithms will be a discrete value. Classification algorithm to be implemented: Naïve Bayes, Decision Tree, K-Nearest Neighbor, and Random Forest.

There are over 90 traffic and aviation accident related datasets on Kaggle, which varies by year and location. There are several accident related problems addressed in this domain. One of which is the comparison of the type of cars in relation to the number of accidents. https://www.kaggle.com/nikitagrec/cars-accidents-sales/log From the visuals of this analysis it can be concluded that California had the highest count of accidents and that coupes along with sedans were considered more dangerous than the other types. Traffic accident severity prediction has been done before, however there are various factors that can influence how detrimental an accident turns out to be. Therefore, exploring different factors that impact the criticality of a car collision can provide a more complete examination of what can be done to alleviate it. The results derived from this study can augment previous analyses that have already been done in this space.

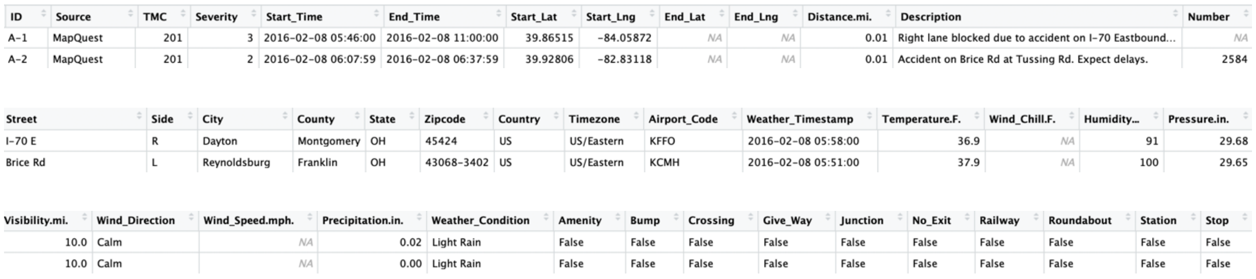
Data Source, Data Preprocessing and Exploratory Data Analysis

A Countrywide Traffic Accident Dataset 2016-2019 from Kaggle will be used in this study (https://www.kaggle.com/sobhanmoosavi/us-accidents). There are about 3 million accident records collected between February 2016 to December 2019 and includes 49 states of the United States. The data was provided by Traffic APIs (MapQuest, Bing, etc.) that were recorded by US and state departments of transportation, law enforcement, traffic cameras and sensors.

Original variables are separated into several different data types; Boolean, categorical, continuous and discrete. Boolean variables consist of; crossing, stop, traffic signal, junction, amenity, bump, give way, no exit, railway, roundabout, station, stop, traffic calming, traffic signal, and turning loop. Categorical variables include; source, TMC, severity, side, sunrise/sunset, weather condition, time zone, airport code, zip code, state, country, county, city, wind direction, civil twilight, nautical twilight, and astronomical twilight. Continuous variables include; start latitude, start longitude, end latitude, end longitude, distance, start time, end time, temperature, wind speed, wind chill, humidity, pressure, visibility, weather timestamp, and precipitation. Lastly, discrete variables involve; ID, description, number, and street.

Sample raw data points

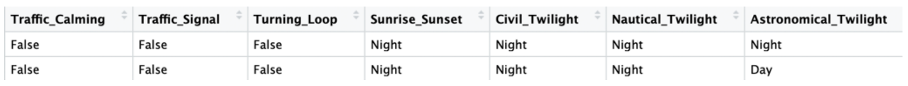
Figure 1.



ID	Source	TMC	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance.mi.	Description	Number
A-1	MapQuest	201	3	2016-02-08 05:46:00	2016-02-08 11:00:00	39.86515	-84.05872	NA	NA	0.01	Right lane blocked due to accident on I-70 Eastbound...	NA
A-2	MapQuest	201	2	2016-02-08 06:07:59	2016-02-08 06:37:59	39.92806	-82.83118	NA	NA	0.01	Accident on Brice Rd at Tussing Rd. Expect delays.	2584

Street	Side	City	County	State	Zipcode	Country	Timezone	Airport_Code	Weather_Timestamp	Temperature.F.	Wind_Chill.F.	Humidity...	Pressure.in.
I-70 E	R	Dayton	Montgomery	OH	45424	US	US/Eastern	KFFO	2016-02-08 05:58:00	36.9	NA	91	29.68
Brice Rd	L	Reynoldsburg	Franklin	OH	43068-3402	US	US/Eastern	KCMH	2016-02-08 05:51:00	37.9	NA	100	29.65

Visibility.mi.	Wind_Direction	Wind_Speed.mph.	Precipitation.in.	Weather_Condition	Amenity	Bump	Crossing	Give_Way	Junction	No_Exit	Railway	Roundabout	Station	Stop
10.0	Calm	NA	0.02	Light Rain	False	False	False	False	False	False	False	False	False	False
10.0	Calm	NA	0.00	Light Rain	False	False	False	False	False	False	False	False	False	False

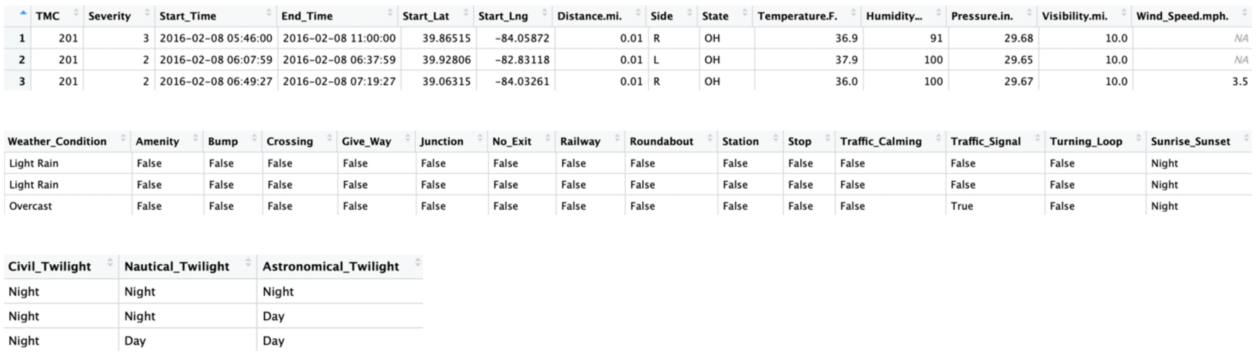


Traffic_Calming	Traffic_Signal	Turning_Loop	Sunrise_Sunset	Civil_Twilight	Nautical_Twilight	Astronomical_Twilight
False	False	False	Night	Night	Night	Night
False	False	False	Night	Night	Night	Day

In dealing with the variables, it was decided that variables with high NA proportion are deemed as not impactful. Variables with NA proportion larger than 50% were eliminated since it lacks adequate information to yield insightful conclusions. End latitude, end longitude, number, wind chill, and precipitation are the five variables that will be left out from our study. In addition, it was concluded that variables ID, source, description, time zone, airport code, weather timestamp, and wind direction show little or no relationship to accident severity level. These variables were removed and not considered in the analysis. There are quite a few variables that indicate the location of an accident occurrence. To reduce the size and clean the dataset further, we turned our focus on nationwide and statewide patterns. In doing so, country, city, county, street, and zip code variables were dropped and not utilized. The original dataset had 49 variables, after the initial filtering there is now 32.

Sample cleaned data

Figure 2.



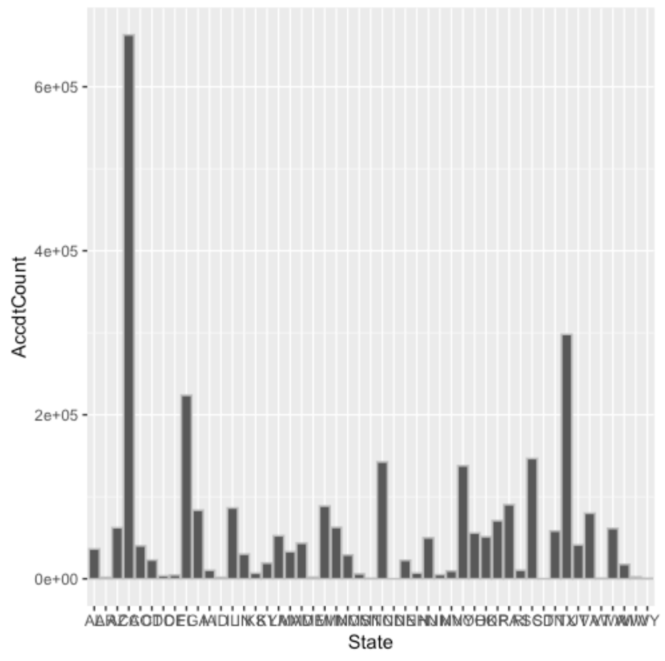
	TMC	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	Distance.mi.	Side	State	Temperature.F.	Humidity...	Pressure.in.	Visibility.mi.	Wind_Speed.mph.
1	201	3	2016-02-08 05:46:00	2016-02-08 11:00:00	39.86515	-84.05872	0.01	R	OH	36.9	91	29.68	10.0	NA
2	201	2	2016-02-08 06:07:59	2016-02-08 06:37:59	39.92806	-82.83118	0.01	L	OH	37.9	100	29.65	10.0	NA
3	201	2	2016-02-08 06:49:27	2016-02-08 07:19:27	39.06315	-84.03261	0.01	R	OH	36.0	100	29.67	10.0	3.5

Weather_Condition	Amenity	Bump	Crossing	Give_Way	Junction	No_Exit	Railway	Roundabout	Station	Stop	Traffic_Calming	Traffic_Signal	Turning_Loop	Sunrise_Sunset
Light Rain	False	False	False	False	False	False	False	False	False	False	False	False	False	Night
Light Rain	False	False	False	False	False	False	False	False	False	False	False	False	False	Night
Overcast	False	False	False	False	False	False	False	False	False	False	False	True	False	Night

Civil_Twilight	Nautical_Twilight	Astronomical_Twilight
Night	Night	Night
Night	Night	Day
Night	Day	Day

Visualization in exploring data

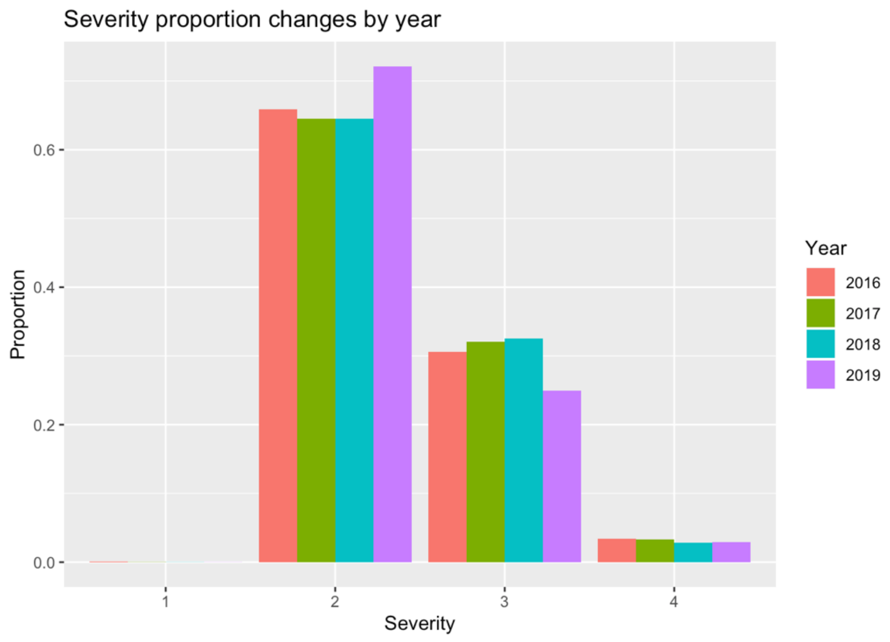
Figure 3. Maximum number of accidents (state level)



It is observed in Figure 3 that California has the highest count of accidents followed by Texas.

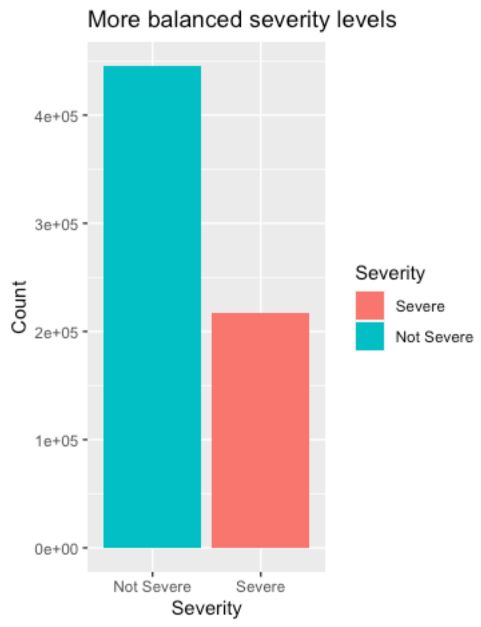
Due to the size of the dataset and limited computing power, California is selected as our Target.

Figure 4. Count of accidents in each severity level by year



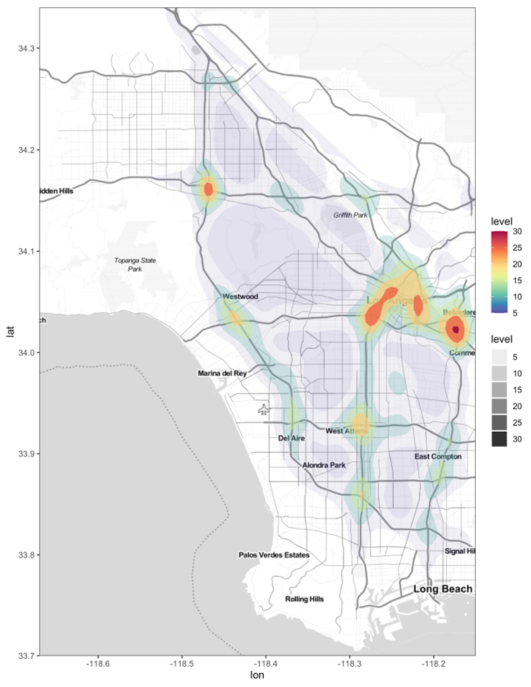
Level of severity was measured on a scale of 1-4, the figure above depicts severity 2 as the most common for all years in the dataset (2016-2019). Severity 1 and 4 were less common in comparison.

Figure 5. Count of not severe vs. severe accidents



The data collected is not well distributed in severity level, since most collisions were classified as having a severity level of 2 or 3 it was regrouped. Severity level of 1 and 2 is classified as "Not Severe" and severity level of 3 and 4 as "Severe". In doing so the balance of the target variable improved tremendously.

Figure 6. Heat map Los Angeles level



As we turn our focus on Los Angeles, the heat map illustrates freeway/highway junctions to have higher density of collision count.

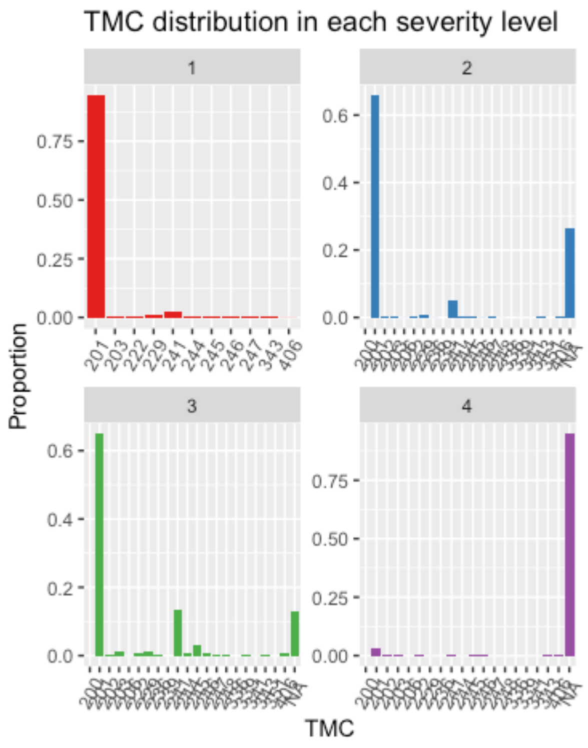
Feature Engineering and Selection

The EDA and data cleaning feature selection process involved separating out variables with majority NA values and values deemed necessary to predict the severity of an accident. Variables deemed not useful in predicting severity levels (ID, Source, Description, Time zone, Weather timestamp, wind direction) were dropped. Due to the general makeup of the dataset, we vouched for a classification prediction for the severity of an accident and not on the probability of an accident occurring. There were no interaction terms used in this project. Variables selected for the prediction model were categorical (close proximity to: Amenity, Bump, Crossing, Give Way, Junction, No Exit, Railway, Roundabout, Station, Stop, Traffic Calming, Traffic

Signal, Weather Condition) and numerical (location coordinates, time, visibility, temperature, wind speed). One feature transformation was made on the date/time variable to split out the hourly, weekly, and monthly data into separate variables.

The purpose of the Traffic Message Channel (TMC) code is to provide a more detailed description of the incident. This particular variable was examined in relation to severity.

Figure 7. TMC code distribution per severity level



It is evident that NA value is a significant feature of level 4 severity, thus NA value is treated as another TMC code. Prior to splitting the dataset into training and test data, number of records fewer than 20 were left out from weather condition and TMC code. Removing levels with few recordings will help prevent imbalance issues that may arise. The balance of independent categorical variables was analyzed, the result tells us that the data has the same values for several predictors. Once the dataset is split for training and testing, the levels in training dataset and test dataset may not match.

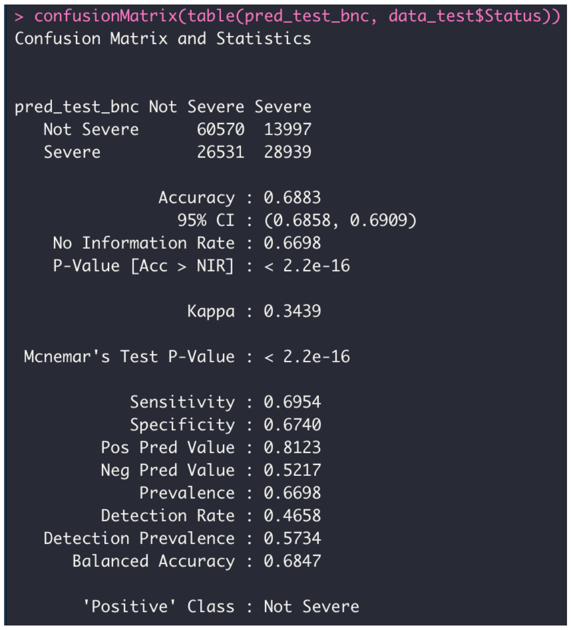
Additional feature engineering during modeling included dropping some feature levels and narrowing down the data set to a usable size. We decided to cut the data set down by focusing on accidents in California, and filtered the data set to only include ‘CA’ records. The dependent variable, Severity, was then transformed from integer factor levels (1-4) to a binary severe/not-severe variable (1-2 Not Severe, 3-4 Severe). Finally, before modeling, the data set was randomly sampled using the over/under sampling function (ovun.sample) to reduce the training data set and cross-validate the model.

Modeling

Naive Bayes

The Naive Bayes algorithm uses the Bayes classifier and makes predictions for the independent variable classification with given predictor values. The Bayes classifier assumes independence of the predictor variables, Gaussian distribution of numeric predictors. For attributes with missing values, the corresponding table entries are omitted for prediction.

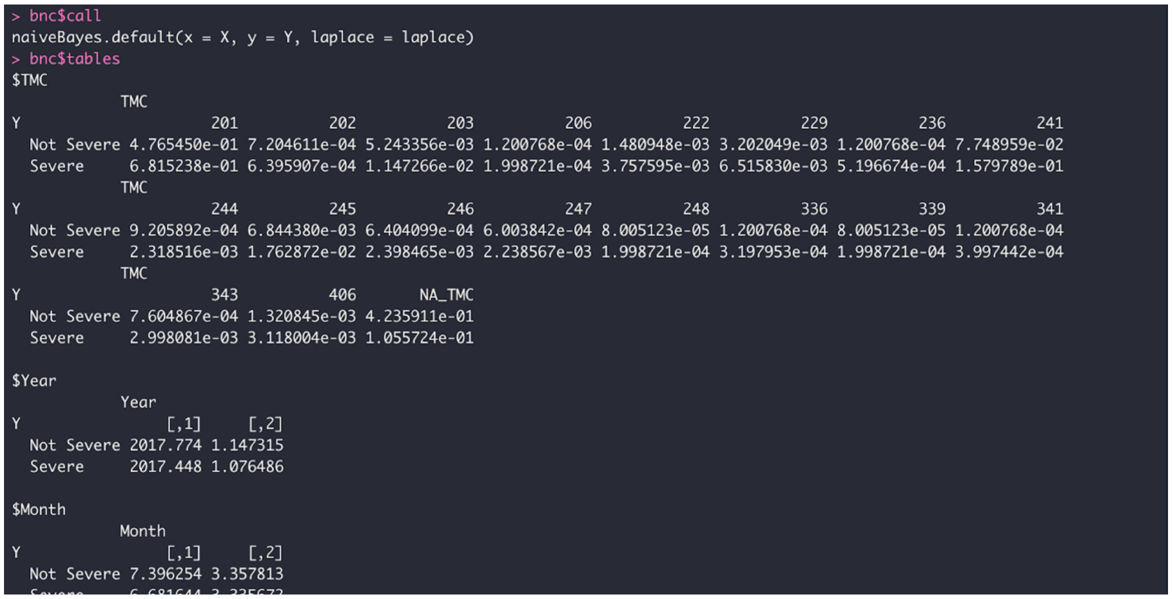
Figure 8. Confusion Matrix for Bayes model



```
> confusionMatrix(table(pred_test_bnc, data_test$Status))
Confusion Matrix and Statistics


pred_test_bnc Not Severe Severe
   Not Severe      60570  13997
   Severe          26531  28939

               Accuracy : 0.6883
                 95% CI : (0.6858, 0.6909)
    No Information Rate : 0.6698
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3439

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

            Sensitivity : 0.6954
            Specificity : 0.6740
         Pos Pred Value : 0.8123
         Neg Pred Value : 0.5217
             Prevalence : 0.6698
         Detection Rate : 0.4658
   Detection Prevalence : 0.5734
      Balanced Accuracy : 0.6847

       'Positive' Class : Not Severe
```

The Bayes algorithm had an accuracy of 68.83% in predicting the severity of the accident. The sensitivity and specificity (accurate Severe/Not-Severe) showed that the accuracy of the model was pretty much the same for both Severe/Not-Severe cases. This model is not quite usable based on the accuracy.

Figure 9. Sample output for Bayesian model

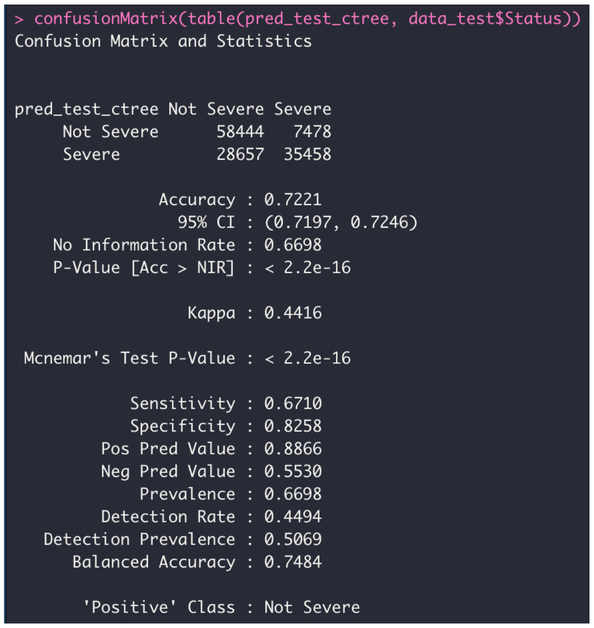


```
> bnc$call
naiveBayes.default(x = X, y = Y, laplace = laplace)
> bnc$tables
$TMC
           TMC
Y                     201          202          203          206          222          229          236          241
  Not Severe 4.765450e-01 7.204611e-04 5.243356e-03 1.200768e-04 1.480948e-03 3.202049e-03 1.200768e-04 7.748959e-02
  Severe     6.815238e-01 6.395907e-04 1.147266e-02 1.998721e-04 3.757595e-03 6.515830e-03 5.196674e-04 1.579789e-01
           TMC
Y                     244          245          246          247          248          336          339          341
  Not Severe 9.205892e-04 6.844380e-03 6.404099e-04 6.003842e-04 8.005123e-05 1.200768e-04 8.005123e-05 1.200768e-04
  Severe     2.318516e-03 1.762872e-02 2.398465e-03 2.238567e-03 1.998721e-04 3.197953e-04 1.998721e-04 3.997442e-04
           TMC
Y                     343          406       NA_TMC
  Not Severe 7.604867e-04 1.320845e-03 4.235911e-01
  Severe     2.998081e-03 3.118004e-03 1.055724e-01

$Year
           Year
Y                [,1]     [,2]
  Not Severe 2017.774 1.147315
  Severe     2017.448 1.076486

$Month
           Month
Y                [,1]     [,2]
  Not Severe 7.396254 3.357813
```

Classification Trees

A Classification Tree was used as the second attempt at predicting accident severity. Classification trees are used for predicting qualitative responses (as opposed to a regression tree which predicts quantitative responses). A Classification tree uses recursive binary splitting to generate the decision tree. Once the trained data is used to grow the tree it is then tested against the test data. This type of model can be useful in our classification problem (Severe and Not-Severe predictions).

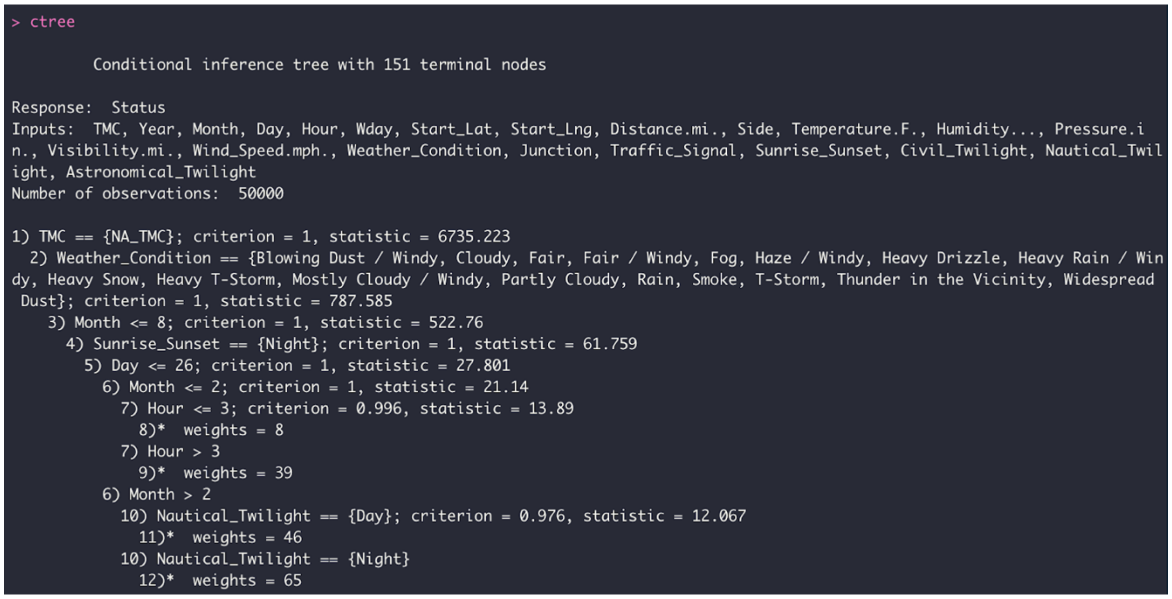
Figure 10. Confusion matrix for Classification Tree



```
> confusionMatrix(table(pred_test_ctree, data_test$Status))
Confusion Matrix and Statistics


pred_test_ctree Not Severe Severe
     Not Severe      58444   7478
     Severe          28657  35458

               Accuracy : 0.7221
                 95% CI : (0.7197, 0.7246)
    No Information Rate : 0.6698
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4416

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

            Sensitivity : 0.6710
            Specificity : 0.8258
         Pos Pred Value : 0.8866
         Neg Pred Value : 0.5530
             Prevalence : 0.6698
         Detection Rate : 0.4494
   Detection Prevalence : 0.5069
      Balanced Accuracy : 0.7484

       'Positive' Class : Not Severe
```

The decision tree model performed slightly better (72.21% accurate) than the Naive Bayes model (68.83% accurate). When looking at the specificity/sensitivity for the model, it looks like the model is pretty good at predicting Severe accidents (82.58% accurate). Overall, the Classification Tree model has superficial results, but it is interesting in its ability to semi-accurately predict severe accidents.

Figure 11. Classification Tree sample output (Nodes)

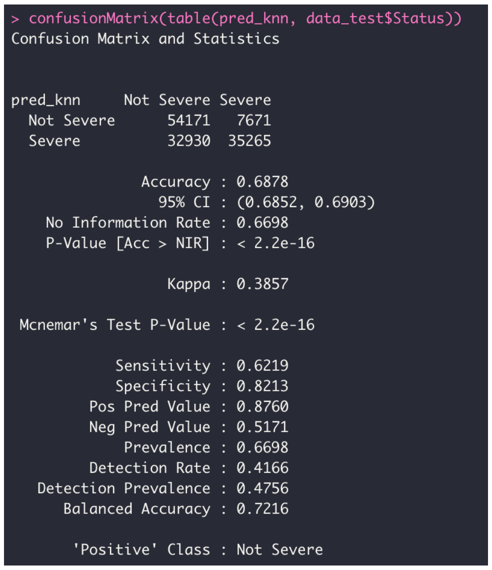


```
> ctree

         Conditional inference tree with 151 terminal nodes

Response:  Status
Inputs:  TMC, Year, Month, Day, Hour, Wday, Start_Lat, Start_Lng, Distance.mi., Side, Temperature.F., Humidity..., Pressure.i
n., Visibility.mi., Wind_Speed.mph., Weather_Condition, Junction, Traffic_Signal, Sunrise_Sunset, Civil_Twilight, Nautical_Twil
ight, Astronomical_Twilight
Number of observations:  50000

1) TMC == {NA_TMC}; criterion = 1, statistic = 6735.223
  2) Weather_Condition == {Blowing Dust / Windy, Cloudy, Fair, Fair / Windy, Fog, Haze / Windy, Heavy Drizzle, Heavy Rain / Win
dy, Heavy Snow, Heavy T-Storm, Mostly Cloudy / Windy, Partly Cloudy, Rain, Smoke, T-Storm, Thunder in the Vicinity, Widespread
 Dust}; criterion = 1, statistic = 787.585
    3) Month <= 8; criterion = 1, statistic = 522.76
      4) Sunrise_Sunset == {Night}; criterion = 1, statistic = 61.759
        5) Day <= 26; criterion = 1, statistic = 27.801
          6) Month <= 2; criterion = 1, statistic = 21.14
            7) Hour <= 3; criterion = 0.996, statistic = 13.89
              8)*  weights = 8
            7) Hour > 3
              9)*  weights = 39
          6) Month > 2
            10) Nautical_Twilight == {Day}; criterion = 0.976, statistic = 12.067
              11)*  weights = 46
            10) Nautical_Twilight == {Night}
              12)*  weights = 65
```

KNN

K-nearest neighbors (KNN) is used to identify K points in training data closest to test observations. As K increases, the technique becomes less flexible and produces a decision limit close to linear. Optimal value for K will be contingent on the bias-variance tradeoff. Small value of K will result in a more flexible fit with low bias but high variance.

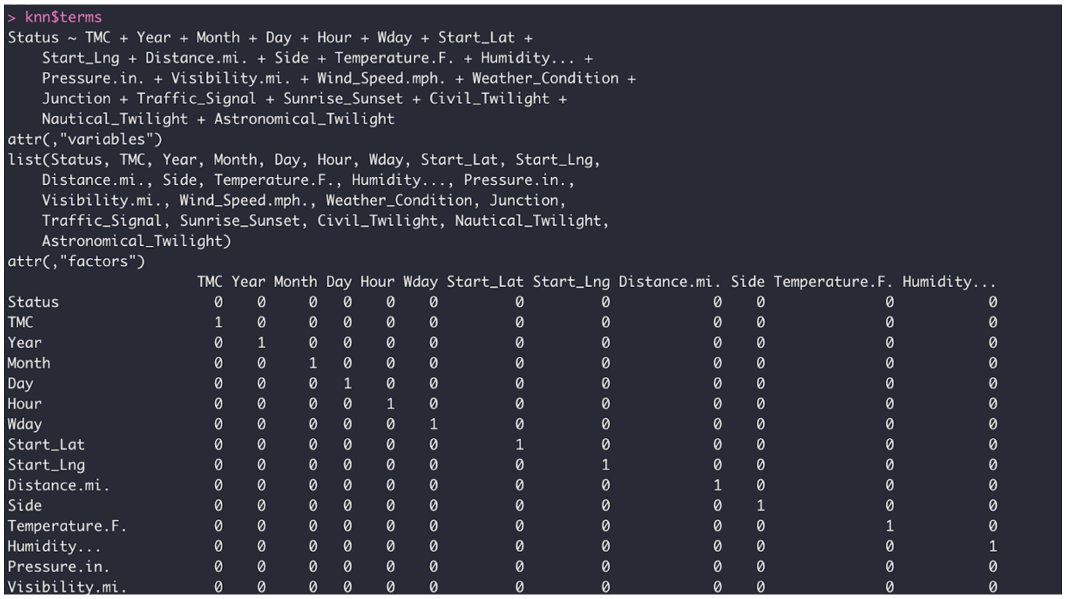
Figure 12. Confusion matrix for KNN



```
> confusionMatrix(table(pred_knn, data_test$Status))
Confusion Matrix and Statistics


pred_knn     Not Severe Severe
  Not Severe      54171   7671
  Severe          32930  35265

               Accuracy : 0.6878
                 95% CI : (0.6852, 0.6903)
    No Information Rate : 0.6698
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3857

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

            Sensitivity : 0.6219
            Specificity : 0.8213
         Pos Pred Value : 0.8760
         Neg Pred Value : 0.5171
             Prevalence : 0.6698
         Detection Rate : 0.4166
   Detection Prevalence : 0.4756
      Balanced Accuracy : 0.7216

       'Positive' Class : Not Severe
```

With 23 features incorporated, an accuracy of 68.78% was returned from the model, a yield in accuracy similar to that of Naïve Bayes. The prediction of Severe accident (82.13% accurate) was much more accurate than Non-Severe accident (62.19% accurate).

Figure 13. Sample output of KNN

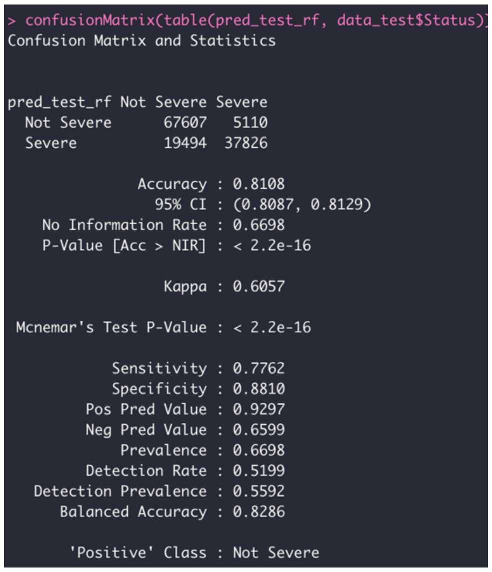


```
> knn$terms
Status ~ TMC + Year + Month + Day + Hour + Wday + Start_Lat +
    Start_Lng + Distance.mi. + Side + Temperature.F. + Humidity... +
    Pressure.in. + Visibility.mi. + Wind_Speed.mph. + Weather_Condition +
    Junction + Traffic_Signal + Sunrise_Sunset + Civil_Twilight +
    Nautical_Twilight + Astronomical_Twilight
attr(,"variables")
list(Status, TMC, Year, Month, Day, Hour, Wday, Start_Lat, Start_Lng,
    Distance.mi., Side, Temperature.F., Humidity..., Pressure.in.,
    Visibility.mi., Wind_Speed.mph., Weather_Condition, Junction,
    Traffic_Signal, Sunrise_Sunset, Civil_Twilight, Nautical_Twilight,
    Astronomical_Twilight)
attr(,"factors")
                      TMC Year Month Day Hour Wday Start_Lat Start_Lng Distance.mi. Side Temperature.F. Humidity...
Status                  0    0     0   0    0    0         0         0            0    0              0           0
TMC                     1    0     0   0    0    0         0         0            0    0              0           0
Year                    0    1     0   0    0    0         0         0            0    0              0           0
Month                   0    0     1   0    0    0         0         0            0    0              0           0
Day                     0    0     0   1    0    0         0         0            0    0              0           0
Hour                    0    0     0   0    1    0         0         0            0    0              0           0
Wday                    0    0     0   0    0    1         0         0            0    0              0           0
Start_Lat               0    0     0   0    0    0         1         0            0    0              0           0
Start_Lng               0    0     0   0    0    0         0         1            0    0              0           0
Distance.mi.            0    0     0   0    0    0         0         0            1    0              0           0
Side                    0    0     0   0    0    0         0         0            0    1              0           0
Temperature.F.          0    0     0   0    0    0         0         0            0    0              1           0
Humidity...             0    0     0   0    0    0         0         0            0    0              0           1
Pressure.in.            0    0     0   0    0    0         0         0            0    0              0           0
Visibility.mi.          0    0     0   0    0    0         0         0            0    0              0           0
```

Random Forest

Random Forest does not overfit and can give an estimate of what variables are significant in the classification. It upholds accuracy when a large percentage of the data is absent, and has an effective way of assessing missing data. For these reasons Random Forest was selected to project traffic accident severity levels in our final model run.

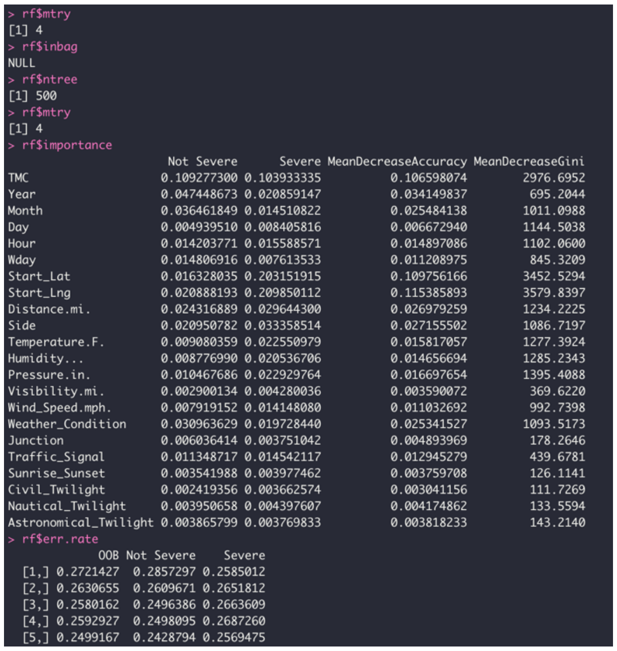
Figure 14. Confusion matrix for Random Forest



```
> confusionMatrix(table(pred_test_rf, data_test$Status)
Confusion Matrix and Statistics


pred_test_rf Not Severe Severe
  Not Severe      67607   5110
  Severe          19494  37826

               Accuracy : 0.8108
                 95% CI : (0.8087, 0.8129)
    No Information Rate : 0.6698
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6057

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

            Sensitivity : 0.7762
            Specificity : 0.8810
         Pos Pred Value : 0.9297
         Neg Pred Value : 0.6599
             Prevalence : 0.6698
         Detection Rate : 0.5199
   Detection Prevalence : 0.5592
      Balanced Accuracy : 0.8286

       'Positive' Class : Not Severe
```

Keeping input features constant, the overall accuracy (81.08%) is the best amongst all model runs in this study. Prediction of Severe accidents had an accuracy of 88.10%, while Non-Severe accidents had an accuracy of 77.62%. Roughly a 10% increase in accuracy compared to the output of Classification Tree. Evidently, it outperformed previous models.

Figure 15. Sample output of Random Forest

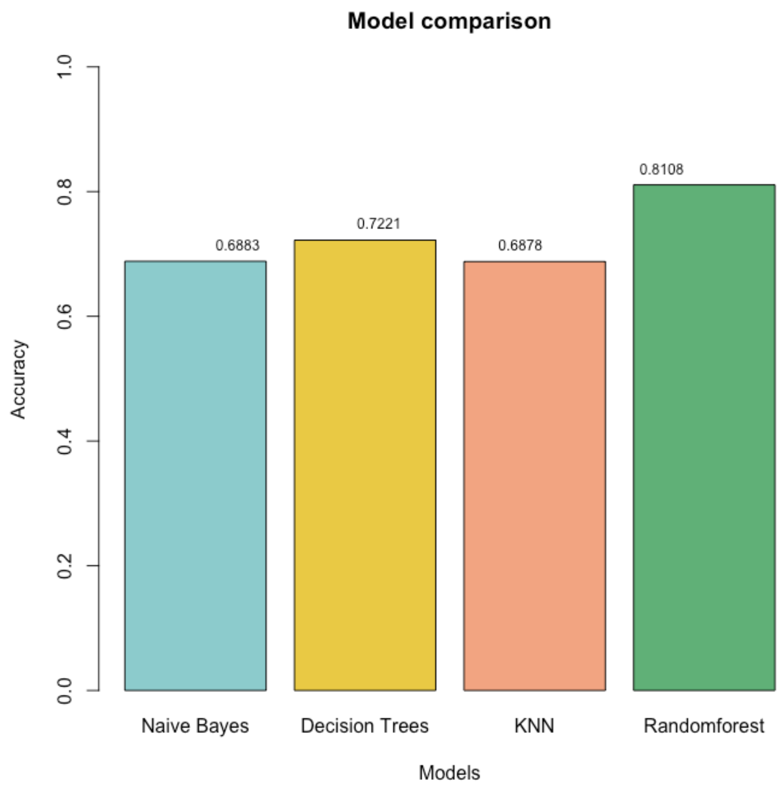


```
> rf$mtry
[1] 4
> rf$inbag
NULL
> rf$ntree
[1] 500
> rf$mtry
[1] 4
> rf$importance
                      Not Severe      Severe MeanDecreaseAccuracy MeanDecreaseGini
TMC                  0.109277300 0.103933335          0.106598074        2976.6952
Year                 0.047448673 0.020859147          0.034149837         695.2044
Month                0.036461849 0.014510822          0.025484138        1011.0988
Day                  0.004939510 0.008405816          0.006672940        1144.5038
Hour                 0.014203771 0.015588571          0.014897086        1102.0600
Wday                 0.014806916 0.007613533          0.011208975         845.3209
Start_Lat            0.016328035 0.203151915          0.109756166        3452.5294
Start_Lng            0.020888193 0.209850112          0.115385893        3579.8397
Distance.mi.         0.024316889 0.029644300          0.026979259        1234.2225
Side                 0.020950782 0.033358514          0.027155502        1086.7197
Temperature.F.       0.009080359 0.022550979          0.015817057        1277.3924
Humidity...          0.008776990 0.020536706          0.014656694        1285.2343
Pressure.in.         0.010467686 0.022929764          0.016697654        1395.4088
Visibility.mi.       0.002900134 0.004280036          0.003590072         369.6220
Wind_Speed.mph.      0.007919152 0.014148080          0.011032692         992.7398
Weather_Condition    0.030963629 0.019728440          0.025341527        1093.5173
Junction             0.006036414 0.003751042          0.004893969         178.2646
Traffic_Signal       0.011348717 0.014542117          0.012945279         439.6781
Sunrise_Sunset       0.003541988 0.003977462          0.003759708         126.1141
Civil_Twilight       0.002419356 0.003662574          0.003041156         111.7269
Nautical_Twilight    0.003950658 0.004397607          0.004174862         133.5594
Astronomical_Twilight 0.003865799 0.003769833          0.003818233         143.2140
> rf$err.rate
            OOB Not Severe    Severe
  [1,] 0.2721427  0.2857297 0.2585012
  [2,] 0.2630655  0.2609671 0.2651812
  [3,] 0.2580162  0.2496386 0.2663609
  [4,] 0.2592927  0.2498095 0.2687260
  [5,] 0.2499167  0.2428794 0.2569475
```

Conclusion

Model Iterations	Testing accuracy	Training accuracy
Naive Bayes	68.83%	68.29%
Decision Trees	72.21%	75.7%
KNN	68.78%	100%
Random Forest	81.08%	100%

Figure 16. Model Run Comparisons



The Random forest algorithm seems to perform the best out of all models. What's interesting is 3 of the models (Decision Trees, KNN, and Random forest) were more accurate at predicting severe accidents. In the Random forest model, the highest importance features in predicting severe traffic accidents are longitude/latitude coordinates. This could mean that there are specific coordinate locations where a severe traffic accident is more likely. Other than the Start Longitude and Start Latitude variables, no other variables clearly stick out as important predictors in accident severity

The second-best model, Decision (Classification) tree, is also more accurate at predicting severe accidents than non-severe accidents. It seems like tree-based models performed better and are more suited to this type of classification problem, specifically where independent variables are not clearly related.

In completing this analysis, we gained new insight in EDA and experienced limiting computing power first hand. In our data cleaning process, variables with majority NA values were deliberately left out, except for TMC code. While exploring the relationship between TMC code and severity level, it was evident that the NA values of this variable had a large impact on all severity levels. Instead of dropping the NA values with the others, it was treated as a new level of TMC. Seems counterintuitive to keep NA values of any variable however if the influence is significant, it is worth investigating further.

There were some challenges in running support vector machines, and it was left out of our analysis. The size of the dataset proved to be difficult to work with despite cleaning and selecting a specific focus. If this analysis were to be continued, support vector machines is another model run that may contribute greatly to our findings. Perhaps to get it to run with limiting computing power, it will be necessarily to narrow down our focus even further.

Appendix

Details about features in the original dataset:

Traffic Attributes (12)

ID: This is a unique identifier of the accident record.

Source: Indicates source of the accident report (i.e. the API which reported the accident.).

TMC: A traffic accident may have a Traffic Message Channel (TMC) code which provides more detailed description of the event.

Severity: Shows the severity of the accident, a number between 1 and 4, where 1 indicates the least impact on traffic (i.e., short delay as a result of the accident) and 4 indicates a significant impact on traffic (i.e., long delay).

Start_Time: Shows start time of the accident in local time zone.

End_Time: Shows end time of the accident in local time zone.

Start_Lat: Shows latitude in GPS coordinate of the start point.

Start_Lng: Shows longitude in GPS coordinate of the start point.

End_Lat: Shows latitude in GPS coordinate of the end point.

End_Lng: Shows longitude in GPS coordinate of the end point.

Distance(mi): The length of the road extent affected by the accident.

Description: Shows natural language description of the accident.

Address Attributes (9)

Number: Shows the street number in address field.

Street: Shows the street name in address field.

Side: Shows the relative side of the street (Right/Left) in address field.

City: Shows the city in address field.

County: Shows the county in address field.

State: Shows the state in address field.

Zipcode: Shows the zipcode in address field.

Country: Shows the country in address field.

Timezone: Shows timezone based on the location of the accident (eastern, central, etc.).

Weather Attributes (11)

Airport_Code: Denotes an airport-based weather station which is the closest one to location of the accident.

Weather_Timestamp: Shows the time-stamp of weather observation record (in local time).

Temperature(F): Shows the temperature (in Fahrenheit).

Wind_Chill(F): Shows the wind chill (in Fahrenheit).

Humidity(%): Shows the humidity (in percentage).

Pressure(in): Shows the air pressure (in inches).

Visibility(mi): Shows visibility (in miles).

Wind_Direction: Shows wind direction.

Wind_Speed(mph): Shows wind speed (in miles per hour).

Precipitation(in): Shows precipitation amount in inches, if there is any.

Weather_Condition: Shows the weather condition (rain, snow, thunderstorm, fog, etc.).

POI Attributes (13)

Amenity: A Point-Of-Interest (POI) annotation which indicates presence of amenity in a nearby location.

Bump: A POI annotation which indicates presence of speed bump or hump in a nearby location.

Crossing: A POI annotation which indicates presence of crossing in a nearby location.

Give_Way: A POI annotation which indicates presence of give_way sign in a nearby location.

Junction: A POI annotation which indicates presence of junction in a nearby location.

No_Exit: A POI annotation which indicates presence of no_exit sign in a nearby location.

Railway: A POI annotation which indicates presence of railway in a nearby location.

Roundabout: A POI annotation which indicates presence of roundabout in a nearby location.

Station: A POI annotation which indicates presence of station (bus, train, etc.) in a nearby location.

Stop: A POI annotation which indicates presence of stop sign in a nearby location.

Traffic_Calming: A POI annotation which indicates presence of traffic_calming means in a nearby location.

Traffic_Signal: A POI annotation which indicates presence of traffic_signal in a nearby location.

Turning_Loop: A POI annotation which indicates presence of turning_loop in a nearby location.

Period-of-Day (4)

Sunrise_Sunset: Shows the period of day (i.e. day or night) based on sunrise/sunset.

Civil_Twilight: Shows the period of day (i.e. day or night) based on civil twilight.

Nautical_Twilight: Shows the period of day (i.e. day or night) based on nautical twilight.

Astronomical_Twilight: Shows the period of day (i.e. day or night) based on astronomical twilight.

Raw Code

```
###############
## Library ##
###############
library(tidyverse)
library(lubridate)
library(ROSE)
library(caret)
library(e1071)
library(party)
library(kknn)
library(randomForest)


###########################################
## Import dataset and check ##
###########################################
US_Accidents <- read.csv("US_Accidents_Dec19.csv", header = TRUE)
head(US_Accidents,5)
str(US_Accidents)
summary(US_Accidents)
dim(US_Accidents)
# [1] 2974335      49

###########################################
## Data Preprocessing for EDA ##
```

```
###########################################
## Deal with variables
# 1. remove variables with high NA proportion and not useful variables
# variables with NA proportion larger than 50% cannot give enough information to our analysis.
US_Accidents %>% summarise_all(~ mean(is.na(.))) %>%
  pivot_longer(1:49, names_to = "Variables to drop", values_to = "NA proportion") %>%
  filter(`NA proportion` >= 0.5)
# so we drop these five variables:
# End_Lat, End_Lng, Number, Wind_Chill.F., Precipitation.in.
na_cols <- c("End_Lat", "End_Lng", "Number", "Wind_Chill.F.", "Precipitation.in.")

# there are some variables will not give us insights about traffic accidents
# or be useful in predicting severity levels
not_useful <- c("ID", "Source", "Description", "Timezone", "Airport_Code", "Weather_Timestamp",
"Wind_Direction")

data_drop <- US_Accidents %>%
  select(-all_of(na_cols), -all_of(not_useful))

dim(data_drop)
# 2974335 observations and 37 variables now

# 2. location related variables
# There are several variables indicating the location of the accident.
# Apart from the accurate coordinate, longitude and latitude,
# the dataset also contains state, city, county and even street address.
# However, we want to find some nationwide patterns or statewide patterns from this dataset.
# So we can romove them now.
address <- c("Country", "City", "County", "Street", "Zipcode")

data_drop %>%
  select(all_of(address)) %>%
  head(5)

data_US <- data_drop %>%
  select(-all_of(address))

dim(data_US)
#  2974335     32

# 3. transform time variables
# Time variables are in a format that is difficult to manipulate in the original dataset,
# so we transform two time variables to some new variables can reflect like hourly, weekly or monthly patterns.
data_US <- data_US %>%
  separate(Start_Time, into = c("Date", "Time"), sep = " ") %>%
  mutate("Year" = str_sub(Date, 1, 4), "Month" = str_sub(Date, 6, 7), "Day" = str_sub(Date, 9, 10),
        "Wday" = as.character(wday(Date)), "Hour" = str_sub(Time, 1, 2)) %>%
  select(-c("Date", "Time", "End_Time")) %>%
  select(TMC, Severity, Year, Month, Day, Hour, Wday, everything())
# after transformation
data_US %>%
  select(Year, Month, Day, Hour, Wday) %>%
  head(5)
```

```
dim(data_US)
# 2974335      35

# 4. TMC variable problem
# check the relationship betweet TMC and Severity
data_US %>%
  ggplot(aes(factor(TMC), ..prop..)) +
  geom_bar(aes(group = Severity, fill = factor(Severity)), show.legend = F) +
  facet_wrap(~ Severity, scales = "free") +
  labs(x = "TMC",
       y = "Proportion",
       title = "TMC distribution in each severity level") +
  theme(axis.text.x = element_text(angle = 60, vjust = 0.6),
        legend.position = "top") +
  scale_fill_brewer(palette = "Set1")
# we find NA value is an important feature of severity level 4,
# so we decide to treat NA value as a new level of TMC.
data_US <- data_US %>%
  mutate(TMC = replace_na(TMC, "NA_TMC"))

# 5. weather condition NA problems
# we can remove all records containing this variable's NA value because all weather variables are related
table(data_US$Weather_Condition)
data_US$Weather_Condition[data_US$Weather_Condition==""] <- "NA"

sum(is.na(data_US$Weather_Condition))

data_US <- data_US %>%
  filter(!is.na(Weather_Condition))
# data_US$Weather_Condition <- as.character(data_US$Weather_Condition)
# data_US$Weather_Condition <- as.factor(data_US$Weather_Condition)

dim(data_US)
#  2908403      35

# 6. other missing values
summary(data_US)
str(data_US)
# there are still some records containing NA values in continuous variables
# we can replace these NA values with the mean of the corresponding variable
data_US <- data_US %>%
  mutate_if(is.numeric, ~ replace_na(., mean(., na.rm = T)))

# there are still some records containing "NA" values in factor variables
data_US$Side[data_US$Side==" "] <- "NA"
data_US$Sunrise_Sunset[data_US$Sunrise_Sunset==""] <- "NA"
data_US$Civil_Twilight[data_US$Civil_Twilight==""] <- "NA"
data_US$Nautical_Twilight[data_US$Nautical_Twilight==""] <- "NA"
data_US$Astronomical_Twilight[data_US$Astronomical_Twilight==""] <- "NA"
sum(is.na(data_US))

data_US <- data_US %>%
  filter(!is.na(Side)) %>%
  filter(!is.na(Sunrise_Sunset)) %>%
  filter(!is.na(Civil_Twilight)) %>%
  filter(!is.na(Nautical_Twilight)) %>%
  filter(!is.na(Astronomical_Twilight))

dim(data_US)
# 2908322      35

# save a tidy data
write_csv(data_US, "tidydata_US.csv")

############################
## Visualization ##
############################
# set correct type to each variable
data_US <- read.csv("tidydata_US.csv", header = TRUE)

data_US <- data_US %>%
  type_convert() %>%
  mutate(TMC = factor(TMC), Severity = factor(Severity), Year = factor(Year), Wday = factor(Wday)) %>%
  mutate_if(is.logical, factor) %>%
  mutate_if(is.character, factor)
data_US <- data_US %>%
  mutate_if(is.numeric, ~ replace_na(., mean(., na.rm = T)))

# 1. Accident count

# states with maximum number of accidents
data_US %>%
  select(State) %>%
  group_by(State) %>%
  summarise(AccdtCount = sum(duplicated(State))) %>%
  arrange(desc(AccdtCount)) %>%
  ggplot(aes(x=State, y=AccdtCount)) +
  geom_col(color= "grey")

# 2. Accident count in each severity level
data_US %>%
  group_by(Year, Severity) %>%
  count() %>%
  group_by(Year) %>%
  mutate(sum = sum(n)) %>%
  mutate(Proportion = n / sum) %>%
  ggplot(aes(Severity, Proportion)) +
  geom_col(aes(fill = Year), position = "dodge") +
  labs(x = "Severity",
       y = "Proportion",
       title = "Severity proportion changes by year")

# 3. Accident account in different months
data_US %>%
  count(Month) %>%
  ggplot(aes(Month, n)) +
  geom_line(aes(group = 1)) +
  geom_point() +
  labs(y = "Count",
       x = NULL,
       title = "Pattern between accident counts and month") +
  scale_x_discrete(labels = c("Jan", "Feb", "Mar", "Apr", "May",
                              "Jun", "Jul", "Aug", "Sep", "Oct",
                              "Nov", "Dec"))
# 4. Accident account in different day of the week
data_US %>%
  count(Wday) %>%
  ggplot(aes(Wday, n, fill = Wday)) +
  geom_bar(position="dodge",stat = "identity") +
  geom_point() +
  labs(y = "Count",
       x = NULL,
       title = "Pattern between accident counts and day of the week") +
  scale_x_discrete(labels = c("Mon", "Tue", "Wed", "Thr", "Fri",
                              "Sat", "Sun"))
# 5. Accident account in different hour of day
data_US %>%
  ggplot(aes(Hour, fill = !Hour %in% c("07", "08", "16", "17"))) +
  geom_bar(show.legend = F) +
  labs(x = "Hour",
       y = "No of Accidents",
       title = "Hourly Distribution of Accidents")

# 6. Weather with Accident
Weather <- data_US %>% group_by(Weather_Condition) %>% count()

Weather_data <- data.frame(
  name=c("Clear", "Haze" , "Heavy Rain", "Light Rain" ,"Overcast",          "Partly Cloudy",
         "Mostly Cloudy           " ,"Scattered Clouds","Snow") ,
  value=c(808142,34314, 12063, 141063, 382475, 295434, 412520, 204656, 4796))

ggplot(Weather_data, aes(fill = name, x=name, y=value)) +
  geom_bar(stat = "identity")

##### heatmap ########

############################
## CA ##
############################

US_data <- read.csv("tidydata_US.csv", header = TRUE)
summary(US_data$Start_Lat)
summary(data_US)
CA_data <- US_data[US_data$State=="CA",]

library(ggplot2)
library(ggmap)
library(RColorBrewer)
```

```
CA_map <- select(CA_data,Start_Lat,Start_Lng)

max(CA_map$Start_Lat)
min(CA_map$Start_Lat)

max(CA_map$Start_Lng)
min(CA_map$Start_Lng)

map_bounds <- c(left = -125, right = -113, bottom = 32, top =43)
coords.map <- get_stamenmap(map_bounds, zoom = 6, maptype ="toner-lite")


coords.map <- ggmap(coords.map, extent="device", legend="none")

coords.map <- coords.map +
  stat_density2d(data=CA_map,aes(x=Start_Lng, y=Start_Lat,
                                fill=..level.., alpha=..level..), geom="polygon")

coords.map <- coords.map + scale_fill_gradientn(colours=rev(brewer.pal(10,"Spectral")))
coords.map <- coords.map + theme_bw()
ggsave(filename="./CA_map_12.png",height=10,width=10)

############################
## Florida ##
############################
#
FL_data <- US_data[US_data$State=="FL",]
FL_map <- select(FL_data,Start_Lat,Start_Lng)

map_bounds <- c(left = -89, right = -79, bottom = 23, top =32)
coords.map <- get_stamenmap(map_bounds, zoom = 6, maptype ="toner-lite")
coords.map <- ggmap(coords.map, extent="device", legend="none")
coords.map <- coords.map +
  stat_density2d(data=FL_map,aes(x=Start_Lng, y=Start_Lat,
                                fill=..level.., alpha=..level..), geom="polygon")
coords.map <- coords.map + scale_fill_gradientn(colours=rev(brewer.pal(10,"Spectral")))
coords.map <- coords.map + theme_bw()
ggsave(filename="./FL_map.png",height=10,width=10)

############################
## Texas ##
############################
#
TX_data <- US_data[US_data$State=="TX",]
TX_map <- select(TX_data,Start_Lat,Start_Lng)

max(TX_map$Start_Lat)
min(TX_map$Start_Lat)
max(TX_map$Start_Lng)
min(TX_map$Start_Lng)
map_bounds <- c(left = -107, right = -93, bottom = 25, top =37)
coords.map <- get_stamenmap(map_bounds, zoom = 6, maptype ="toner-lite")
coords.map <- ggmap(coords.map, extent="device", legend="none")
coords.map <- coords.map +
  stat_density2d(data=TX_map,aes(x=Start_Lng, y=Start_Lat,
                         fill=..level.., alpha=..level..), geom="polygon")
coords.map <- coords.map + scale_fill_gradientn(colours=rev(brewer.pal(10,"Spectral")))
coords.map <- coords.map + theme_bw()
ggsave(filename="./TX_map.png",height=10,width=10)

############################
## NYC ##
############################
#
summary(data_US$Start_Lng)
US_data <- data_US
NY_data <- US_data[which(US_data$Start_Lng > -74.26 & US_data$Start_Lng < -73.70 &
                    US_data$Start_Lat > 40.49 & US_data$Start_Lat < 40.91), ]
dim(NY_data)
summary(NY_data$Start_Lat)
summary(NY_data$Start_Lng)

NY_map <- select(NY_data,Start_Lat,Start_Lng)

max(NY_map$Start_Lat)
min(NY_map$Start_Lat)
max(NY_map$Start_Lng)
min(NY_map$Start_Lng)
map_bounds <- c(left = -74.26, right = -73.70, bottom = 40.49, top =40.91)
coords.map <- get_stamenmap(map_bounds, zoom = 11, maptype ="toner-lite")
coords.map <- ggmap(coords.map, extent="device", legend="none")
coords.map <- coords.map +
  stat_density2d(data=NY_map,aes(x=Start_Lng, y=Start_Lat,
                         fill=..level.., alpha=..level..), geom="polygon")
coords.map <- coords.map + scale_fill_gradientn(colours=rev(brewer.pal(10,"Spectral")))
coords.map <- coords.map + theme_bw()
ggsave(filename="./NYC_map.png",height=10,width=10)

############################
###### Los Angeles ###########
############################
#
LA_data <- US_data[which(US_data$Start_Lng > -118.67 & US_data$Start_Lng < -118.15 &
                    US_data$Start_Lat > 33.70 & US_data$Start_Lat < 34.34), ]

LA_map <- select(LA_data,Start_Lat,Start_Lng)
fix(LA_map)
map_bounds <- c(left = -118.67, right = -118.15, bottom = 33.70, top =34.34)
coords.map <- get_stamenmap(map_bounds, zoom = 11, maptype ="toner-lite")
coords.map <- ggmap(coords.map, extent="device", legend="none")
coords.map <- coords.map +
  stat_density2d(data=LA_map,aes(x=Start_Lng, y=Start_Lat,
                         fill=..level.., alpha=..level..), geom="polygon")
coords.map <- coords.map + scale_fill_gradientn(colours=rev(brewer.pal(10,"Spectral")))
coords.map <- coords.map + theme_bw()
```

```
ggsave(filename="./LA_map.png",height=10,width=10)

############################
## San Francisco ##
############################
#
SF_data <- US_data[which(US_data$Start_Lng > -122.52 & US_data$Start_Lng < -122.24 &
                         US_data$Start_Lat > 37.71 & US_data$Start_Lat < 37.91), ]

SF_map <- select(SF_data,Start_Lat,Start_Lng)

map_bounds <- c(left = -122.52, right = -122.24, bottom = 37.71, top =37.91)
coords.map <- get_stamenmap(map_bounds, zoom = 12, maptype ="toner-lite")
coords.map <- ggmap(coords.map, extent="device", legend="none")
coords.map <- coords.map +
  stat_density2d(data=SF_map,aes(x=Start_Lng, y=Start_Lat,
                             fill=..level.., alpha=..level..), geom="polygon")
coords.map <- coords.map + scale_fill_gradientn(colours=rev(brewer.pal(10,"Spectral")))
coords.map <- coords.map + theme_bw()
ggsave(filename="./SF_map.png",height=10,width=10)


################################
## Preprocessing for modeling ##
################################
# 1. Narrow down the data - California
# Because of the large size of this dataset and the limited computing power,
# We choose California as our target state.
data_CA <- data_US %>%
  filter(State == "CA") %>%
  select(-State)
dim(data_CA)
# 650316    34

table(data_CA$TMC)
table(data_CA$Weather_Condition)

# 2. remove some levels with few recordings or only one level
# Some weather condition or TMC levels only have a few records,
# which may cause imbalanced problems when we split the dataset into training and test data.
# So we will drop some of them according to the number of records (remove value of records less than 20)
TMC_drop <- data_CA %>%
  count(TMC) %>%
  filter(n < 20) %>%
  select(TMC)
TMC_drop <- TMC_drop$TMC %>% unlist()

data_CA <- data_CA %>%
  filter(!TMC %in% TMC_drop) %>%
  mutate(TMC = factor(TMC))

weather_drop <- data_CA %>%
  count(Weather_Condition) %>%
  filter(n < 20) %>%
  select(Weather_Condition)
weather_drop <- weather_drop$Weather_Condition %>% unlist()

data_CA <- data_CA %>%
  filter(!Weather_Condition %in% weather_drop) %>%
  mutate(Weather_Condition = factor(Weather_Condition))

dim(data_CA)
# 650185     34

# 3. check the balance of the target variable
ggplot(data_CA, aes(x = Severity, fill = Severity)) + geom_bar() +
  labs(y = "Count", title = "Unbalanced severity levels")
# the data is seriously unbalanced in different severity levels
# and most of the accidents are classified as level 2 and level 3,
# so we decided to group the 4 levels into 2 levels.
# Level 1 and level 2 will be grouped as "Not Severe",
# and level 3 and level 4 will be grouped as "Severe".

data_group <- data_CA %>%
  mutate("Status" = factor(ifelse(Severity == "3" | Severity == "4", "Severe", "Not Severe"),
                         levels = c("Not Severe", "Severe"))) %>%
  select(-Severity)

ggplot(data_group, aes(Status, fill = !Status == "Severe")) + geom_bar() +
  scale_fill_discrete(name = "Severity", labels = c("Severe", "Not Severe")) +
  labs(y = "Count", x = "Severity", title = "More balanced severity levels")
# the data looks better, then we can split the data into training and test data

# 4. check the balance of independent categorical variables
summary(data_group[,18:33])

# from the result, we can see most of the data have the same values for some predictors,
# What's worse, when we split the dataset, the levels in the training dataset and test dataset may not match.
# so we can remove them.
zero_variance <- c("Amenity", "Bump", "Crossing", "Give_Way", "No_Exit",
                 "Railway", "Roundabout", "Station", "Stop", "Traffic_Calming", "Turning_Loop")

data_group <- data_group %>%
  select(-all_of(zero_variance))
dim(data_group)
# 650185     23


write_csv(data_group, "tidydata_CA.csv")

# 5.Split the data
# we split the data into two parts

data_group <- read.csv("tidydata_CA.csv", header = TRUE)

set.seed(1234)
```

```
index <- sample(x=nrow(data_group), size=.80*nrow(data_group))
data_train <- data_group[index, ]
data_test <- data_group[-index, ]
dim(data_train)
# 520148     23
dim(data_test)
# 130037     23

# 6. sampling
# we reduce the data size to a scale that is more easily to manipulate.
# and create possibly balanced samples
data_train_sample <- ovun.sample(Status~., data = data_train, method = "both", p = 0.5, N = 50000, seed =
1)$data
###########################################
#### Applying Machine Learning Models #####
###########################################

# Medeling#
# Naive Bayes
bnc <- naiveBayes(Status~., data=data_train_sample)
bnc$tables

# Decision Trees
ctree <- ctree(Status~., data=data_train_sample)
ctree
# KNN
knn <- kknn(Status~.,train = data_train_sample,test = data_test,na.action = na.omit(),
           k=223, scale=TRUE, distance = 2)  # k= sqrt(N)
knn
# Random Forest
rf <- randomForest(Status~., data = data_train_sample, importance=TRUE)

rf$ntree
rf$mtry
rf$importance
rf$err.rate


### Training set predictions

# Naive Bayes
pred_train_bnc <- predict(bnc, data_train_sample)
p_train_bnc <- mean(pred_train_bnc == data_train_sample$Status)

# Decision Trees
pred_train_ctree <- predict(ctree, data_train_sample)
p_train_ctree <- mean(pred_train_ctree == data_train_sample$Status)

# KNN does not have the process of training the model,
# because the test data actually needs to be predicted through
# the training data set.
p_train_knn <- 1
```

```
# Random Forest
pred_train_rf <- predict(rf, data_train_sample)
p_train_rf <- mean(pred_train_rf == data_train_sample$Status)


### Test set predictions

# Naive Bayes
pred_test_bnc <- predict(bnc, data_test)
p_bnc <- mean(pred_test_bnc == data_test$Status)

# Decision Trees
pred_test_ctree <- predict(ctree, data_test)
p_ctree <- mean(pred_test_ctree == data_test$Status)

# KNN
pred_knn <- fitted(knn)
p_knn <- mean(pred_knn == data_test$Status)


# Random Forest
pred_test_rf <- predict(rf, data_test)
p_rf <- mean(pred_test_rf == data_test$Status)

# Confusion Matrix Analysis
confusionMatrix(table(pred_test_bnc, data_test$Status))
confusionMatrix(table(pred_test_ctree, data_test$Status))
confusionMatrix(table(pred_knn, data_test$Status))
confusionMatrix(table(pred_test_rf, data_test$Status))



####################
##### Conclusion ###
####################
##Training set predictions
p_train <- as.vector(c(p_train_bnc, p_train_ctree, p_train_knn, p_train_rf))
p_train <- round(p_train,4)
barplot(p_train, ylim = c(0,1), main = "Training set scores", ylab = "Accuracy",
        xlab = "Models", names.arg = c("Naive Bayes","Decision Trees","KNN","Randomforest"),
        col = c("darkslategray3", "gold2", "lightsalmon1", "mediumseagreen"))
text(p_train, labels = as.character(p_train), pos = 1, cex = 0.75)

# plot
p <- as.vector(c(p_bnc, p_ctree, p_knn, p_rf))
p_round <- round(p,4)
barplot(p_round, ylim = c(0,1), main = "Model comparison", ylab = "Accuracy",
        xlab = "Models", names.arg = c("Naive Bayes","Decision Trees","KNN","Randomforest"),
        col = c("darkslategray3", "gold2", "lightsalmon1", "mediumseagreen"))
text(p, labels = as.character(p_round), pos = 3, cex = 0.75)
```