Predicting Accident Severity in Wake County

DSC – 630

Srilakshmi Bodduluru

Abstract

Prediction of traffic accident severity is one of the crucial steps in the traffic accident management process. Timely recognition of key influences may play an important role for improving traffic safety. This paper presents a model system to predict severity of traffic accidents. For this project I am using US-Accidents data. Due to the limitation of computer capacity, I have focused on Wake County in the State of North Carolina. In this paper, I am going to present classification models for road traffic accidents to predict the severity of the accidents. Dataset was split into training and test dataset. Next, I have generated classification models using machine learning techniques on the training data set, and then verified the performance of the model produced using the test data set. In this study, I have utilized well-known classification models —Generalized Linear Model (GLM), Distributed Random Forest (DRF), Deep Learning (Neural Networks), Gradient Boosting Machine (GBM) and Stacked Ensembles and compared the results. The experimental results show that our classification models performed very well and achieved very good accuracy score.

Introduction /background of the problem

Motor vehicle travel is a major means of transportation in the United States, providing an unparalleled degree of mobility. Yet for all its advantages, motor vehicle crashes were the leading

cause of death. Average number of car accidents in the U.S. every year is 6 million. More than 90 people die in car accidents every day. 3 million people in the U.S. are injured every year in car accidents.

Traffic accidents are a significant source of deaths, injuries, property damage, and a major concern for public health and traffic safety. Accidents are also a major cause of traffic congestion and delay. Accurate predictions of severity can provide crucial information for emergency responders to evaluate the severity level of accidents, estimate the potential impacts, and implement efficient accident management procedures.

Methods

Initial step was focused on downloading the dataset from the site and understanding how the data is distributed across various years and states. Considering the dataset size (over 1 GB) my immediate focus was on choosing the right subset of the data by analyzing the data distribution. The next step was focused on data auditing and EDA process to ensure that only right data is selected for further processing.

I have decided to go with classification analysis for predicting the accident severity. I have decided to use few feature selection methods to find out the important features for the analysis. I have then divided data into training and test datasets for subsequent usage. Further I have extended my analysis to the chosen machine learning algorithms - Generalized Linear Model (GLM), Distributed Random Forest (DRF), Deep Learning (Neural Networks), Gradient Boosting Machine (GBM) and Stacked Ensembles. I have compared the results by their accuracy levels to find out the best model for predicting accident severity. Since this is a multi-class classification problem, I

have utilized overall accuracy, sensitivity and specificity for quantifying the accuracy of all models.

Data Sources & Data Preparation

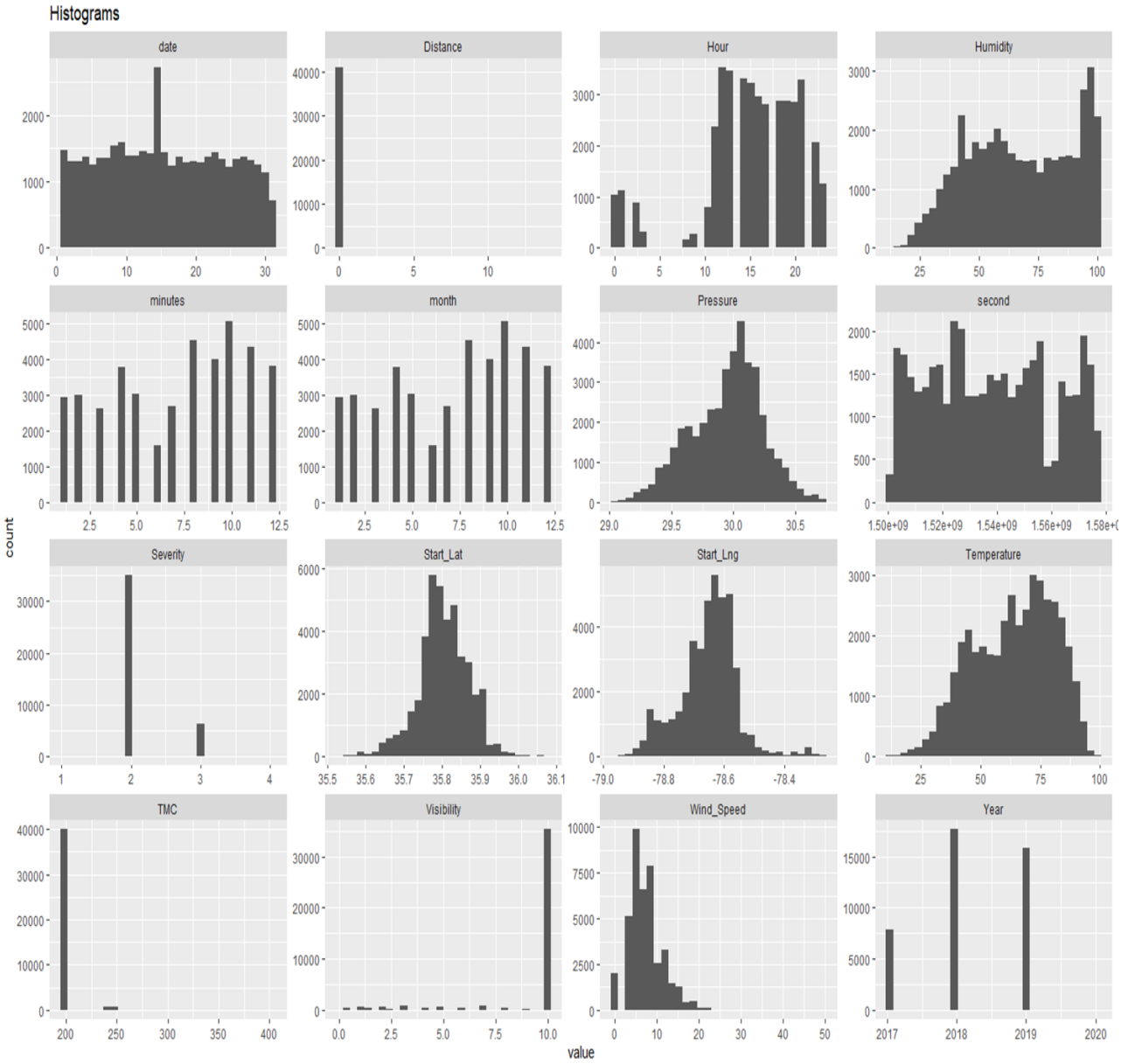
Data source**:** **US_Accidents_Dec19.csv** from kaggle datasets. Due to limitations of my computer I have restricted my study to wake county data. The data set contained 52,640 rows and 49 columns. Evaluated the data structure and summary details to confirm all variables had the right data types. I have changed some column names for easy understanding. I have renamed the target variable classes as low, medium, high and extreme levels of severity for easy understanding. Low level of severity of the accident indicates the least impact on traffic (i.e., short delay as a result of the accident) and extreme indicates a significant impact on traffic (i.e., long delay).

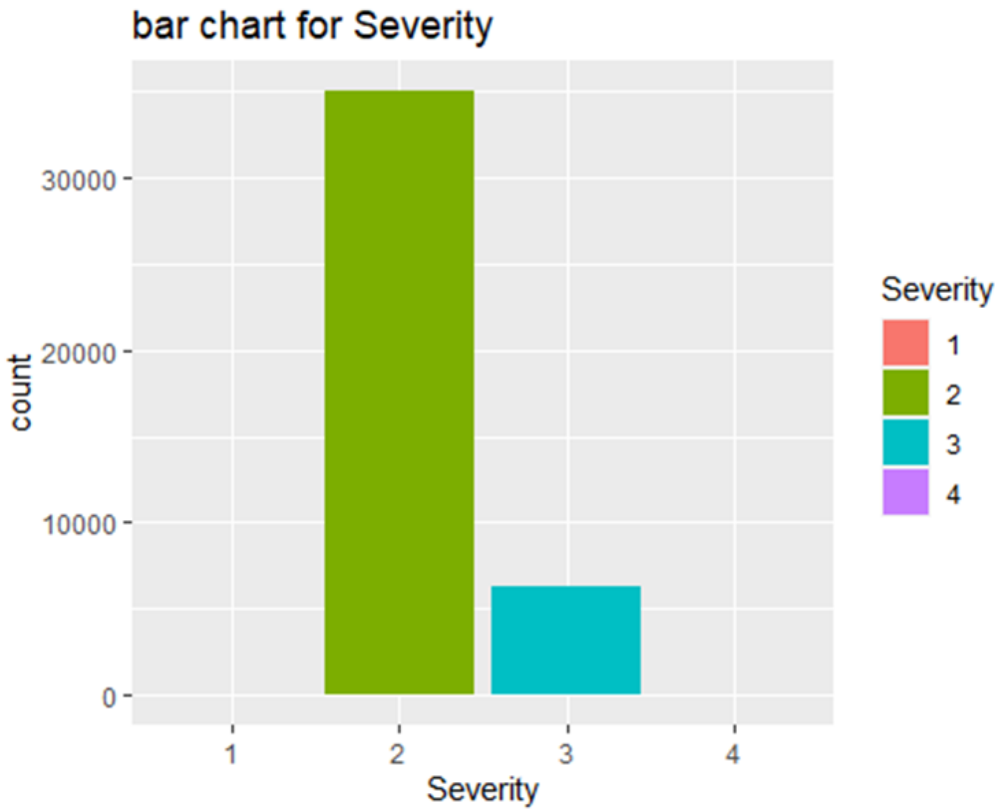
Extract year, month, day, hour, weekday, and time to clear accidents

After extracting the date and time from the data, I have decided to find out any negative time durations so that I can delete those from the data. Luckily there is no such data. I have selected some important columns for my future analysis. I have found through initial analysis that there are some columns with many missing values. To deal with missing values, one approach can be to delete all the missing value rows, however, this approach will result in loosing valuable data. So I took a different approach to treat missing values where I have separated the columns with most number of missing values and then tried to find out the relevance of those columns with the target variable and took really important columns. This process led to minimizing the loss of information. I have removed the outliers and missing rows from the data.

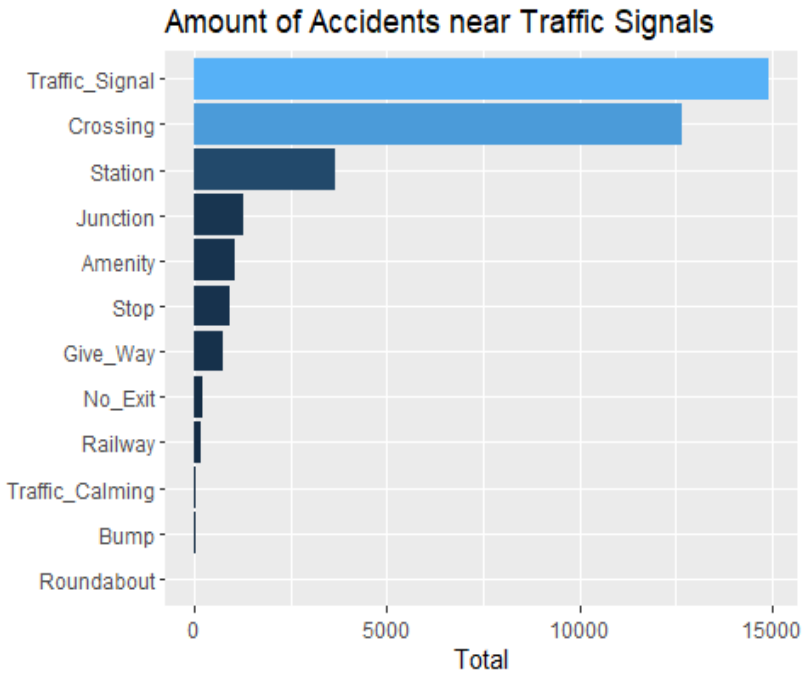
Exploratory Data Analysis (EDA)

I have decided to plot different variables to understand the distribution of the data and target variable.



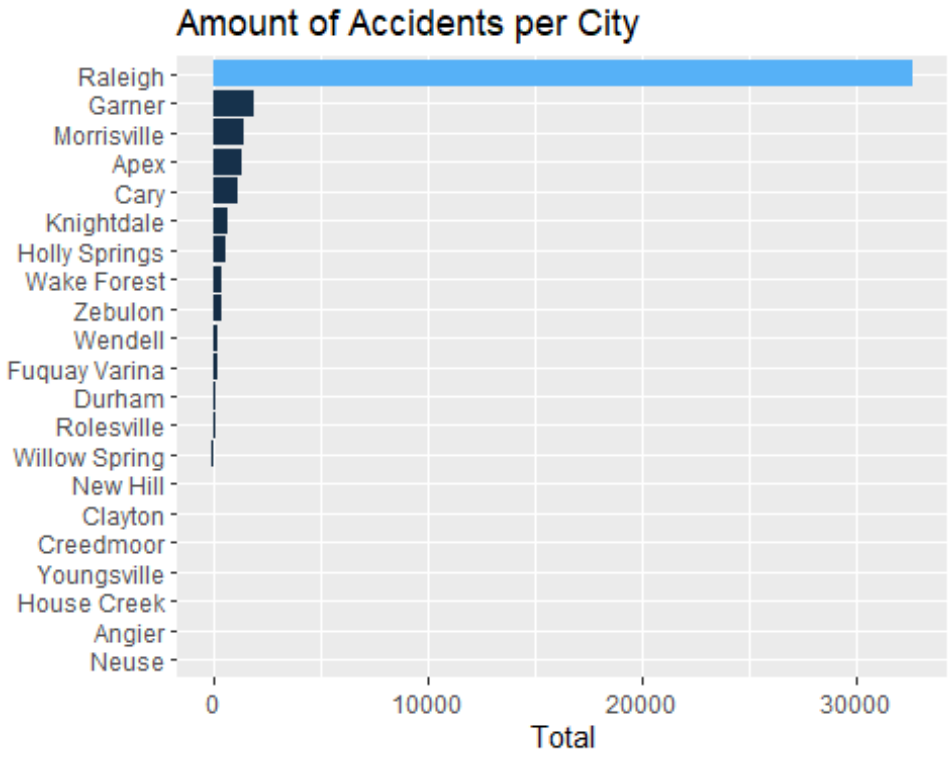


By looking at the initial plots we can conclude that most number of accidents are with severity 2 and 3. Accidents took place mostly near crossing and traffic signal during day time. Most number of accidents occurred during clear visibility days, which was surprising. There are extreme temperatures, ranging from -77 F to 170 F. Maximum wind speed was about 22 mph and Humidity reached 100 %. Precipitation, distance affected and visibility does not seem to be that influential, as most of the accidents occurred during clear visibility conditions. Another key observation is that there is an increase in the number of accidents with the increase of the wind temperature, with the peak at 75°F. The number of accidents at really cold temperatures is very small when compared with the total number of accidents.



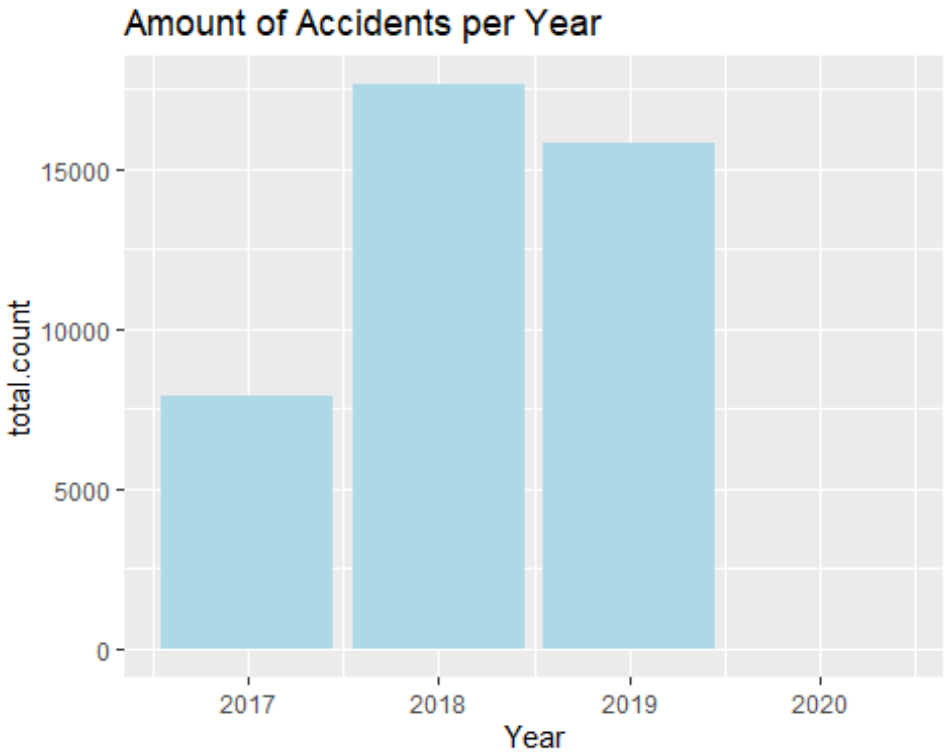
We can also note that most of the accidents happen at traffic signals, then junctions and crossings.

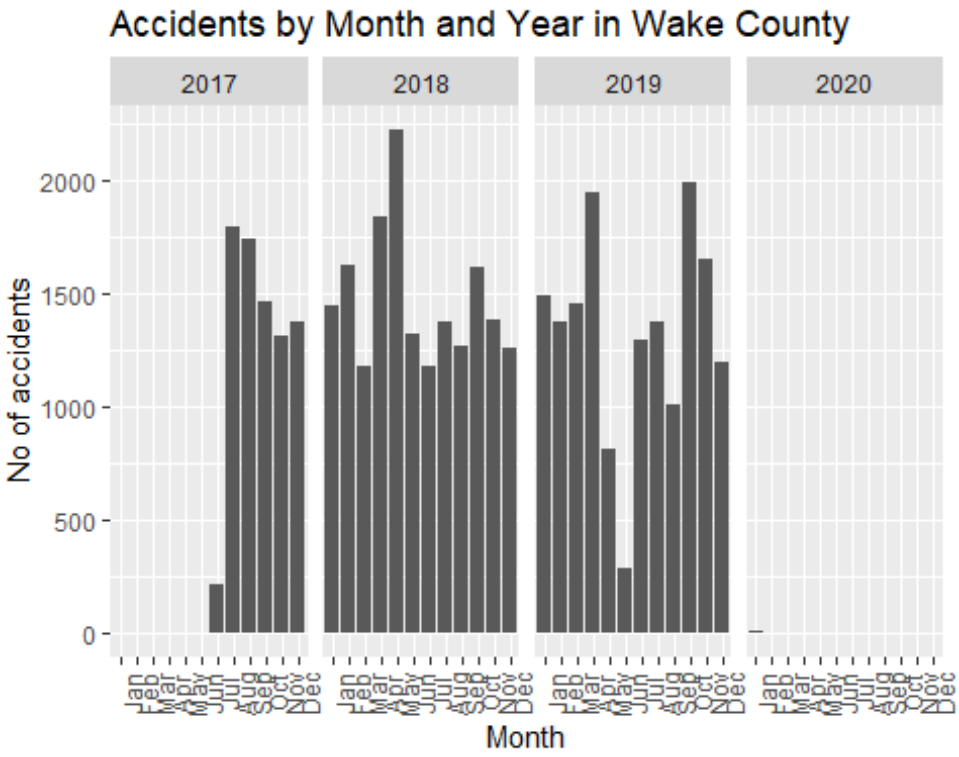
Further I have decide to find out which cities have highest number of accidents.

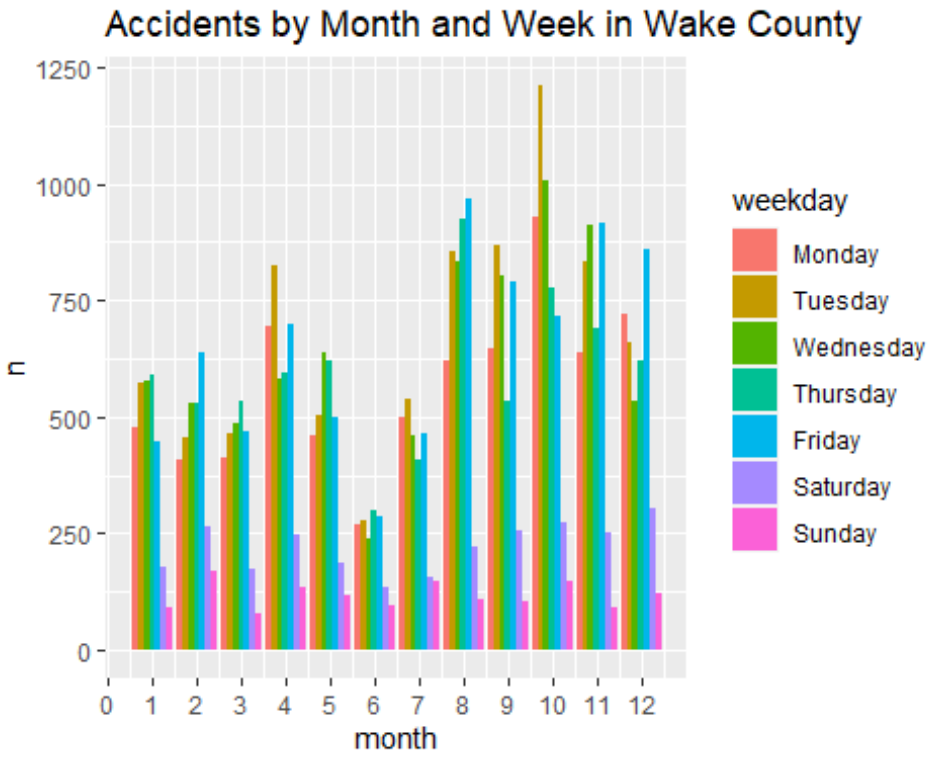


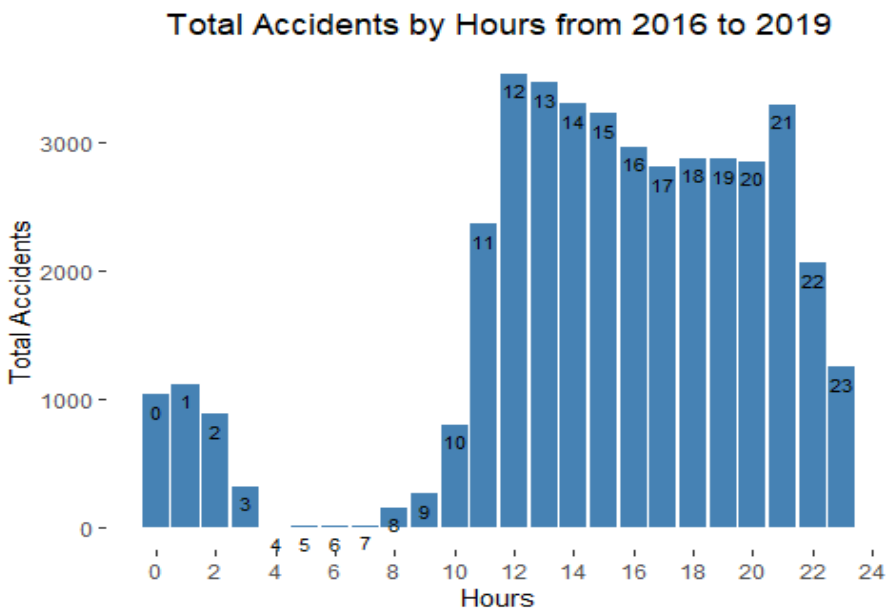
Raleigh had most number of accidents followed by Garner and Morrisville. Further I have decided to find out the impact of day and time on accidents. From the below figures it is clear that most of the accidents occurred in the year 2018. But we cannot conclude anything from this because we don't have complete data for 2017 and 2019.

Next focus was on the occurrence of accidents vs time. The first and easiest option was to look at the occurrence of accidents in specific moments, say grouped by Month (to represent time of the year) and by Hour (to obtain time of the day).

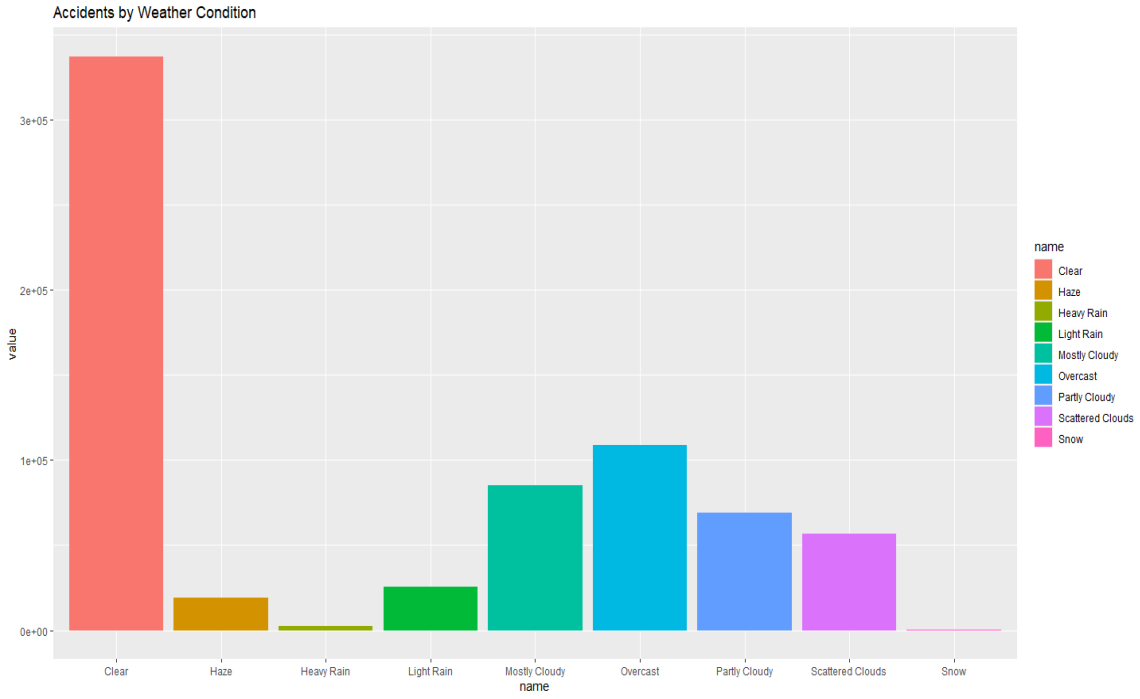



Accidents by Month and Year in Wake County
2017
2018
2019
2020
No of accidents
2000
1500
1000
500
0
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month


Accidents by Month and Week in Wake County
1250
1000
750
500
250
0
n
0
1
2
3
4
5
6
7
8
9
10
11
12
month
weekday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday



The number of accidents reduced between June and September, as well as during night time. Lowest number of accidents was recorded during weekends. It seemed that most of the accidents happen between 12 and 21 hrs. In other words, few accidents are happening during the warmest months and during the coldest hours.



To my surprise most number of accidents occurred during clear days. I have calculated the correlation between time and weather conditions using correlation matrix, and noted that they are not correlated. To support the next steps, converted all categorical variables into dummies using caret package, divided data set into training and test data and used the datasets for predicting the severity using various machine learning algorithms. I have utilized H2O for modeling. H2O is a leading open source machine learning platform that allows building models on large data sets (no sampling needed) and achieve accurate predictions. It is incredibly fast, scalable and easy to implement at any level. It does support rapid machine learning modelling & prototyping and widely utilized by almost all the major organizations.

As soon as the data set was ready for modeling, I went ahead and installed H2O package in R.

```
# Load H2O and startup an H2O cluster

library(h2o)

h2o.init(max_mem_size = "20g")

# http://localhost:54321
```

Next steps was to transfer the data from R to h2o instance. It can be accomplished using as.h2o command.

```
# import dataset (wake county)

wakecounty_dataset_path <- "C:/Users/Inni/Documents/InniFiles/Inni/Bellevue/DSC-630/newdata_transformed.csv"

wakecounty_data.hex <- h2o.importFile(path = wakecounty_dataset_path, destination_frame = "wakecounty_data.hex")
```

```
#str(wakecounty_data.hex)

# Split dataset giving the training dataset 75% of the data

wakecounty_data.split <- h2o.splitFrame(data=wakecounty_data.hex, ratios=0.75)

# Create a training set from the 1st dataset in the split

wakecounty_data.train <- wakecounty_data.split[[1]]

# Create a testing set from the 2nd dataset in the split

wakecounty_data.test <- wakecounty_data.split[[2]]
```

Modelling Results

Model Performance Graphs

Confusion Matrix

A confusion matrix is a table depicting performance of algorithm in terms of false positives, false negatives, true positives, and true negatives.

Variable Importance

Variable importance represents the statistical significance of each variable in the data in terms of its effect on the model. Variables are listed in order of most to least importance. The percentage values represent the percentage of importance across all variables, scaled to 100%. The method of computing each variable's importance depends on the algorithm.

When the model has multiple categories as prediction options we have to calculate metrics that apply for each category.

GLM

GLM algorithm in H2O can be used for all types of regression such as lasso, ridge, logistic, linear etc. A user only needs to modify the family parameter accordingly. For example: To do logistic regression, you can write family = “binomial”.

Generate a GLM model using the training dataset. X represents the predictor column, and y represents the target index.

```
wakecounty_data.glm <- h2o.glm(y = "Severity",
                    x = c(col_names_x),
                    training_frame=wakecounty_data.train,
                    family="multinomial",
                    nfolds=5,
                    alpha=0.5,keep_cross_validation_predictions = TRUE,seed = 1122)
```

retrieve the model performance

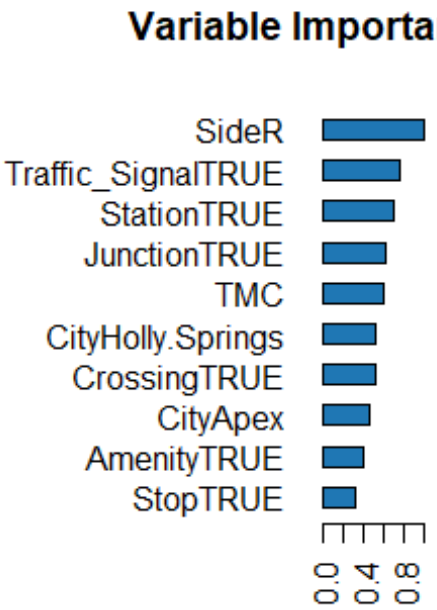
```
perf_glm <- h2o.performance(wakecounty_data.glm, wakecounty_data.test)
```

```
perf_glm
```

```
## H2OMultinomialMetrics: glm
## Test Set Metrics:
## =====================
## MSE: (Extract with `h2o.mse`) 0.08547354
```

```
## RMSE: (Extract with `h2o.rmse`) 0.2923586

## Logloss: (Extract with `h2o.logloss`) 0.2669981

## Mean Per-Class Error: 0.6723048

## Null Deviance: (Extract with `h2o.nulldeviance`) 9075.575

## Residual Deviance: (Extract with `h2o.residual_deviance`) 5539.676

## R^2: (Extract with `h2o.r2`) 0.8374537

## AIC: (Extract with `h2o.aic`) NaN

## Confusion Matrix: Extract with `h2o.confusionMatrix(<model>, <data>)`)

## =========================================================================

## Confusion Matrix: Row labels: Actual class; Column labels: Predicted class

##         extreme high low medium  Error             Rate

## extreme       0    1   0      0 1.0000 =          1 / 1

## high          0  554   0   1058 0.6563 =  1,058 / 1,612

## low           0    1   0      4 1.0000 =          5 / 5

## medium        0  288   0   8468 0.0329 =    288 / 8,756

## Totals        0  844   0   9530 0.1303 = 1,352 / 10,374
```

```
# Predict using the glm model and the testing dataset

pred_glm <- h2o.predict (object=wakecounty_data.glm, newdata=wakecounty_data.test)

pred_glm


# Variable Importance

h2o.varimp_plot(wakecounty_data.glm)

h2o.varimp(wakecounty_data.glm)
```



Random Forest

Distributed Random Forest (DRF) is a powerful classification and regression tool. When given a set of data, DRF generates a forest of classification or regression trees, rather than a single classification or regression tree. Each of these trees is a weak learner built on a subset of rows and columns. More trees will reduce the variance. Both classification and regression take the average prediction over all of their trees to make a final prediction, whether predicting for a class or numeric value.

```
rforest.model <- h2o.randomForest(y = "Severity",
                                  x = c(col_names_x),
                                  training_frame=wakecounty_data.train, ntrees = 100, mtries = 3, max_depth = 20,
```

seed = 1122,keep_cross_validation_predictions = TRUE,nfolds=5)

Model performance

h2o.performance(rforest.model)

```
## H2OMultinomialMetrics: drf
## ** Reported on training data. **
## ** Metrics reported on Out-Of-Bag training samples **
## Training Set Metrics:
## Extract training frame with `h2o.getFrame("RTMP_sid_bf30_1")`
## MSE: (Extract with `h2o.mse`) 0.09541909
## RMSE: (Extract with `h2o.rmse`) 0.3088998
## Logloss: (Extract with `h2o.logloss`) 0.3114674
## Mean Per-Class Error: 0.746566
## R^2: (Extract with `h2o.r2`) 0.8153949
## Confusion Matrix: Extract with `h2o.confusionMatrix(<model>,train =
TRUE)`)
## =========================================================================
## Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
##         extreme high low medium  Error             Rate
## extreme       0    2   0      7 1.0000 =          9 / 9
## high          0   68   0   4635 0.9855 =  4,635 / 4,703
## low           0    0   0      4 1.0000 =          4 / 4
## medium        0   19   0  26267 0.0007 =    19 / 26,286
## Totals        0   89   0  30913 0.1505 = 4,667 / 31,002
```

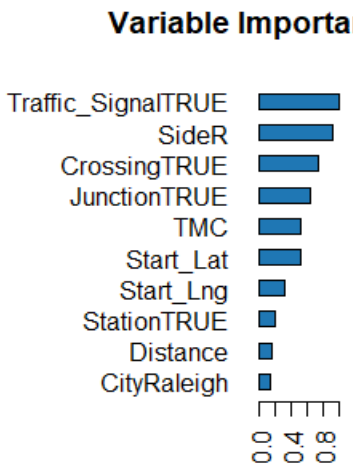
Predict using the random forest model and the testing dataset

pred_rfm <- h2o.predict(object=rforest.model, newdata=wakecounty_data.test)

pred_rfm

#check variable importance

h2o.varimp_plot(rforest.model)



GBM

Gradient Boosting Machine (for Regression and Classification) is a forward learning ensemble method. The guiding heuristic is that good predictive results can be obtained through increasingly refined approximations. H2O's GBM sequentially builds regression trees on all the features of the dataset in a fully distributed way - each tree is built in parallel.

gbm.model <- h2o.gbm(y="Severity", x=c(col_names_x), training_frame = wakecounty_data.train, ntrees = 1000, max_depth = 10, learn_rate = 0.01, seed = 1122, keep_cross_validation_predictions = TRUE,nfolds=5)

h2o.performance (gbm.model)

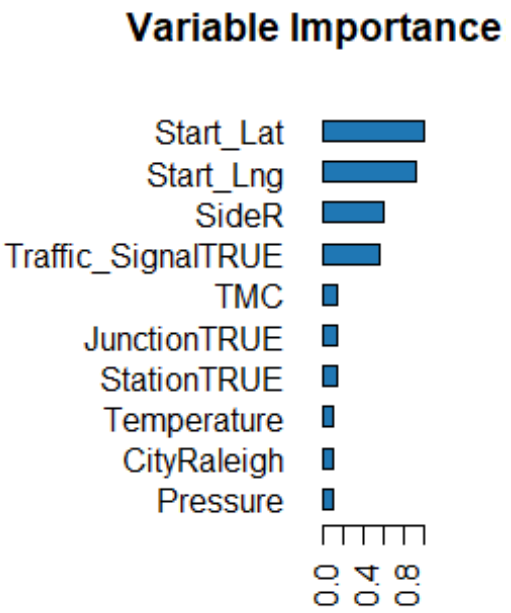
```
## H2OMultinomialMetrics: gbm
## ** Reported on training data. **
## Training Set Metrics:
## Extract training frame with `h2o.getFrame("RTMP_sid_bf30_1")`
## MSE: (Extract with `h2o.mse`) 0.01528126
## RMSE: (Extract with `h2o.rmse`) 0.1236174
## Logloss: (Extract with `h2o.logloss`) 0.06298732
## Mean Per-Class Error: 0.007737452
## R^2: (Extract with `h2o.r2`) 0.9704357
## Confusion Matrix: Extract with `h2o.confusionMatrix(<model>,train = TRUE)`)
## =========================================================================
## Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
##         extreme high low medium  Error           Rate
## extreme       9    0   0      0 0.0000 =        0 / 9
## high          0 4624   0     79 0.0168 =   79 / 4,703
## low           0    0   4      0 0.0000 =        0 / 4
## medium        0  372   0  25914 0.0142 = 372 / 26,286
## Totals        9 4996   4  25993 0.0145 = 451 / 31,002
```

#making predictions using gbm

predict.gbm <- as.data.frame(h2o.predict(gbm.model, wakecounty_data.test))

#check variable importance

h2o.varimp_plot(gbm.model)



Deep learning model

In deep learning algorithm, there exist 3 layers namely input layer, hidden layer and output layer. It works as follows:

- We feed the data to input layer.
- It then transmits the data to hidden layer. These hidden layer comprises of neurons. These neurons uses some function and assist in mapping nonlinear relationship among the variables. The hidden layers are user specified.

- Finally, these hidden layers delivers the output to output layer which then gives us the result.

Let’s implement this algorithm now.

```
#deep learning model

dlearning.model <- h2o.deeplearning(y="Severity", x=c(col_names_x), training_frame = wakecounty_data.train,

        epoch = 80,

        hidden = c(100,100),

        activation = "Rectifier",

        seed = 1122,keep_cross_validation_predictions = TRUE,nfolds=5)

perf_dl <- h2o.performance(dlearning.model)

perf_dl

## H2OMultinomialMetrics: deeplearning
## ** Reported on training data. **
## ** Metrics reported on temporary training frame with 10064 samples **
##
## Training Set Metrics:
## =====================
## MSE: (Extract with `h2o.mse`) 0.01512155
## RMSE: (Extract with `h2o.rmse`) 0.1229697
## Logloss: (Extract with `h2o.logloss`) 0.05757757
```

```
## Mean Per-Class Error: 0.3666736

## Confusion Matrix: Extract with `h2o.confusionMatrix(<model>,train =

TRUE)`)

## =========================================================================

## Confusion Matrix: Row labels: Actual class; Column labels: Predicted class

##         extreme high low medium  Error           Rate

## extreme       3    1   0      1 0.4000 =        2 / 5

## high          0 1412   0     80 0.0536 =   80 / 1,492

## low           0    0   0      1 1.0000 =        1 / 1

## medium        0  112   0   8454 0.0131 =  112 / 8,566

## Totals        3 1525   0   8536 0.0194 = 195 / 10,064
```

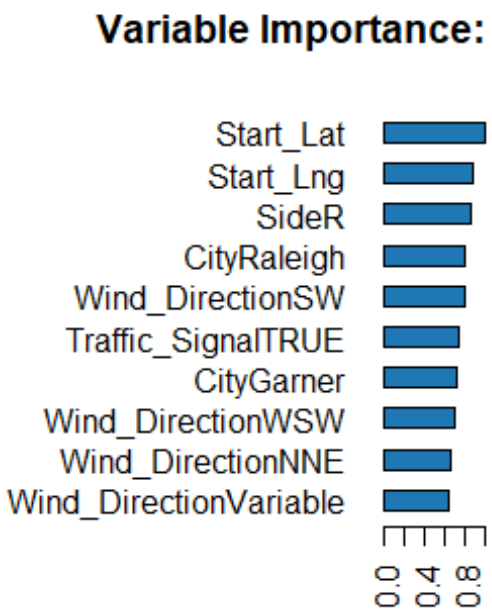
#making predictions using deep learning model

predict.dlearning <- h2o.predict(dlearning.model, wakecounty_data.test)

#check variable importance

h2o.varimp_plot(dlearning.model)

h2o.varimp(dlearning.model)



Stacked Ensemble

H2O's Stacked Ensemble method is supervised ensemble machine learning algorithm that finds the optimal combination of a collection of prediction algorithms using a process called stacking. Like all supervised models in H2O, Stacked Ensemble supports regression, binary classification and multiclass classification.

ensemble <- h2o.stackedEnsemble(y="Severity", x=c(col_names_x), training_frame = wakecounty_data.train,

model_id = "my_ensemble_binomial", base_models = list(wakecounty_data.glm,

rforest.model,gbm.model, dlearning.model))

Generate predictions on a test set (if neccessary)

```
pred_ensemble <- h2o.predict(ensemble, newdata = wakecounty_data.test)

# Eval ensemble performance on a test set

perf_ensemble <- h2o.performance(ensemble, newdata = wakecounty_data.test)

perf_ensemble
```

```
## H2OMultinomialMetrics: stackedensemble
## Test Set Metrics:
## =====================
## MSE: (Extract with `h2o.mse`) 0.04802304
## RMSE: (Extract with `h2o.rmse`) 0.2191416
## Logloss: (Extract with `h2o.logloss`) 0.1682774

## Mean Per-Class Error: 0.559168
## Null Deviance: (Extract with `h2o.nulldeviance`) 9075.575
## Residual Deviance: (Extract with `h2o.residual_deviance`) 3491.42
## AIC: (Extract with `h2o.aic`) NaN
## Confusion Matrix: Extract with `h2o.confusionMatrix(<model>, <data>)`)
## Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
##         extreme high low medium  Error           Rate
## extreme       0    1   0      0 1.0000 =        1 / 1
## high          0 1294   0    318 0.1973 =  318 / 1,612
## low           0    0   0      5 1.0000 =        5 / 5
## medium        0  345   0   8411 0.0394 =  345 / 8,756
## Totals        0 1640   0   8734 0.0645 = 669 / 10,374
```

Results

A multi-categorical classification model can be evaluated by the sensitivity and specificity of each possible class. A model that is great for predicting one category can be terrible for the others. These evaluation metrics allow us to analyze models for different purposes. Since my problem is a multi-class classification problem I chose to use overall accuracy, sensitivity and specificity as the performance measures.

Sensitivity is the metric that evaluates a model's ability to predict true positives of each available category. Specificity is the metric that evaluates a model's ability to predict true negatives of each available category. These metrics apply to any categorical model. The equations for calculating these metrics are below.

$$\textbf{Sensitivity} = \frac{\textbf{True Positives}}{\textbf{True Positives} + \textbf{False Negatives}}$$

$$\textbf{Specificity} = \frac{\textbf{True Negatives}}{\textbf{True Negatives} + \textbf{False Positives}}$$

I have calculated the sensitivity and specificity of the models for each class using the above equations and presented the results in the tabular form as shown below.

Model	Class	Evaluation Metric		
		Sensitivity	Specificity	Overall Accuracy
GLM	Low	0	0.99	0.87
	Medium	0.97	0.67	
	High	0.33	0.88	
	Extreme	0	0.99	
RF	Low	0	0.99	0.85
	Medium	1	0.81	
	High	0.02	0.85	
	Extreme	0	0.99	
GBM	Low	1	1	0.985
	Medium	0.98	0.92	
	High	0.98	0.99	
	Extreme	1	1	
DL	Low	0	0.99	0.98
	Medium	0.98	0.93	
	High	0.96	0.98	
	Extreme	0.8	0.99	
Stacked Ensemble	Low	0	0.99	0.93
	Medium	0.96	0.79	
	High	0.8	0.96	
	Extreme	0	0.99	

Conclusion

All models performed very well when overall accuracy was considered. With GBM I was able to achieve the overall accuracy of 98.5% which is very good result. Since this is a multi-class classification problem, I would like to analyze the accuracy of the models in terms of sensitivity and specificity as well. Every model performed quite well when we consider the specificity, which is a measure of predicting the true negatives of each category. But if we consider predicting the true positives, GBM performed outstandingly. Since it is very important for us to predict the severity of the accident which is our main intention of this project, I would recommend to go with

GBM which gave us extraordinary results in terms of accuracy as well as sensitivity. One key problem I have observed with the data is that, the number of records are very few for low and extreme classes of the target variable that resulted in a class-imbalance problem for the data. I strongly felt that class-imbalance problem might have been the reason for other models' poor performance in predicting some classes of the target variable. This project can be extended by treating the data with proper techniques for handling imbalanced classes before proceeding in to the modelling phase.

Acknowledgments

Many thanks to Prof. Ashley for her guidance and continuous support throughout the project. I offer my sincere appreciation for the learning opportunities provided by her. Completion of this project could not have been accomplished without the support of my classmates. I am very much thankful to my caring, loving, and supportive family, without whom this project would not have been possible.

References

[1] Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, and Rajiv Ramnath (2019). "A Countrywide Traffic Accident Dataset.", arXiv preprint arXiv:1906.05409 (2019).

[2] Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, Radu Teodorescu, and Rajiv Ramnath (2019). "Accident Risk Prediction based on Heterogeneous Sparse Data: New Dataset and Insights." In proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2019.

[3] https://towardsdatascience.com/a-comprehensive-machine-learning-workflow-with-multiple-modelling-using-caret-and-caretensemble-in-fcbf6d80b5f2

[4] https://www.analyticsvidhya.com/blog/2016/12/practical-guide-to-implement-machine-learning-with-caret-package-in-r-with-practice-problem/

[5] https://www.kaggle.com/sobhanmoosavi/us-accidents

[6] https://www.hindawi.com/journals/mpe/2013/547904/

[7] http://h2o-release.s3.amazonaws.com/h2o/rel-zahradnik/1/docs-website/h2o-docs/data-munging/splitting-datasets.html

[8] https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/

[9] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/stacked-ensembles.html#training-base-models-for-the-ensemble

[10] https://h2o-release.s3.amazonaws.com/h2o/master/4698/docs-website/h2o-docs/performance-and-prediction.html#accuracy

[11] https://towardsdatascience.com/evaluating-categorical-models-ii-sensitivity-and-specificity-e181e573cff8