Road Accident Analysis

Project Report

By

Deepika Siriah

Lee Parker

Rashmi Sawant

Shraddha Masuti

San Diego State University
MIS 620
Fall 2017

Executive Summary

This project was an analysis of traffic accidents in the United Kingdom. The goal of the project was to identify predictors of the severity of traffic accidents, categorized as serious (including fatal) and slight accidents.

A large amount of data is usually gathered at traffic accidents where police reports are involved. Demographics of the driver(s), vehicle type, vehicle movement at the time of the crash, road type and conditions, intersection classification, traffic control mechanisms, weather, etc. The goal was to identify which of these factors that were predictors of severity when an accident did occur.

The cost of road accidents, both in terms of injury and death, and in monetary costs, is substantial. In particular, the cost of serious and fatal accidents are very high. If predictors can be identified, then we can determine whether severity depends more on the driver, a certain type of road or intersections, or other conditions or parameters. The value of identifying these predictors is two fold:

- The information can be used by governments, city planners, road engineers in designing safer roads
- The information can be used by insurance to measure risk and assign liability

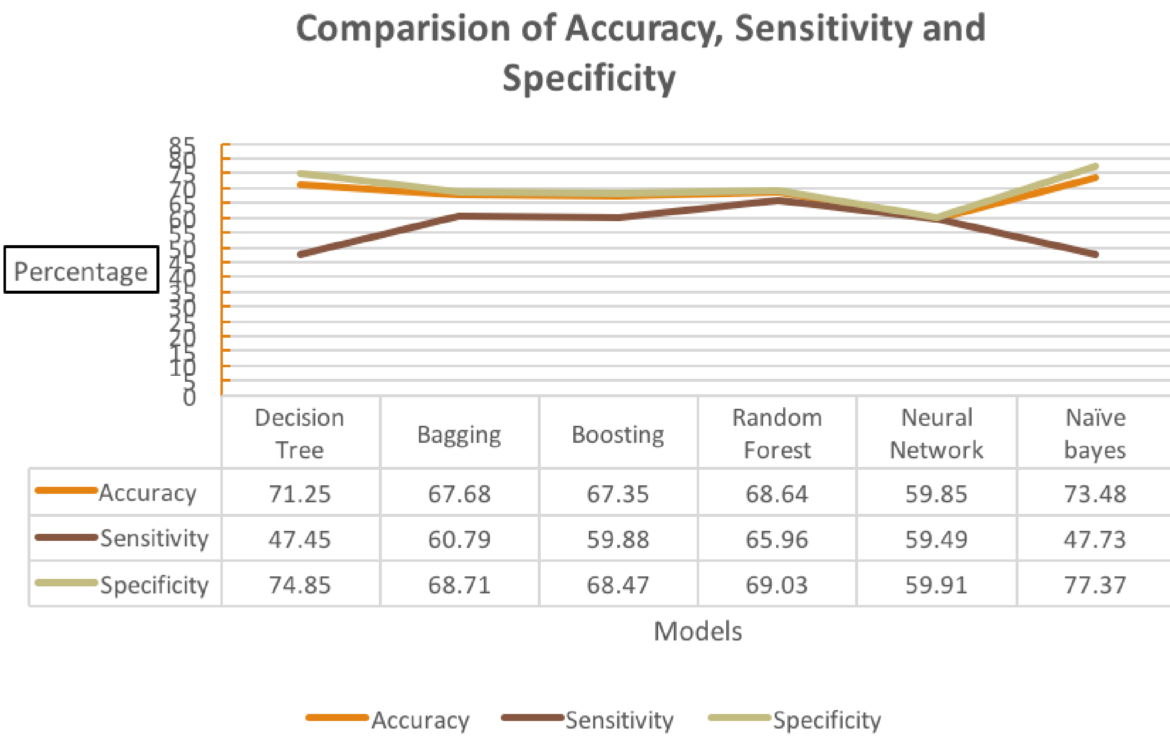
The project had the following goals:

1. Identify predictors of road accidents
2. Predict severity of road accidents
3. Determine the best predictive model of accident severity

The project's hypothesis was that factors can be identified for use in a model that can predict the severity of an accident at a rate greater than chance, i.e. 50%.

The raw dataset was ~285K rows (accident reports) and 70 columns. The final dataset after data preparation was ~145K rows and 16 columns (including the dependent variable of accident severity). Data preparation included such activities as removing columns and rows with missing data and running correlation to eliminate highly correlated variables.

The data was split 70/30 into training and test data sets. To correct imbalanced data, downsampling, SMOTE, and ROSE were used. ROSE produced the best results in preliminary testing and was used in the final training dataset.



	Decision Tree	Bagging	Boosting	Random Forest	Neural Network	Naïve bayes
Accuracy	71.25	67.68	67.35	68.64	59.85	73.48
Sensitivity	47.45	60.79	59.88	65.96	59.49	47.73
Specificity	74.85	68.71	68.47	69.03	59.91	77.37

Figure A. Accuracy, Sensitivity, and Specificity - All Models

Figure A above depicts accuracy, sensitivity, and specificity on test data for all the models. As described in the report, the project goals were met and the hypothesis satisfied. The Random Forest model was found to have the highest performance, with training and test sensitivity scores of 0.7954 and 0.6597 respectively, which exceeds chance (0.5). Overall accuracy was not the highest, but accuracy alone does not indicate the best model. We are more concerned with sensitivity, as that represents the ability to identify serious accidents. With Random Forest we have the highest performance. Predictive factors were more difficult to interpret, with the top five predictors of all of the models comprised of eight of the fifteen factors.

This initial study was successful for its established criteria. Positive predictive value and overall accuracy needs to improve to have a more confident model. Further study is recommended to include adjusting the models using different combinations and numbers of factors and obtaining additional data from other sources.

Data Analytics Lifecycle

I. Discovery

The subject of this data analytics project was road accidents and severity. The goal of the project was to identify predictors of the severity of traffic accidents, categorized as serious (which includes fatal), and slight accidents.

Severity in road accidents:

- Slight - accidents involving some damage to property, which may be significant, but no or minor injury to one or more accident victims.
- Serious - accidents involving significant damage to property and serious injury to one or more accident victims
    - Fatal - accidents involving significant damage to property and death to one or more accident victims.

A large amount of data is usually gathered at traffic accidents where police reports are involved. Demographics of the driver(s), vehicle type, vehicle movement at the time of the crash, road type and conditions, intersection classification, traffic control mechanisms, weather, etc. The goal was to identify which of these factors that were predictors of severity when an accident did occur.

The cost of road accidents, both in terms of injury and death, and in monetary costs, is substantial. In particular, the cost of serious and fatal accidents are very high.

Figure 1 depicts the cost of accidents by severity. Fatal accidents cost approximately $2.4M per casualty, and $2.7M per accident. Serious accidents cost over $250K per casualty, and over $300K per accident. Even slight accidents are of significant cost to the person of average income, being on average $20K per casualty (when they do occur) and $30K per accident. Insurance and liability are important factors in the coverage of the costs of accidents, and the safety considerations of road design are of concern to government managers and officials.

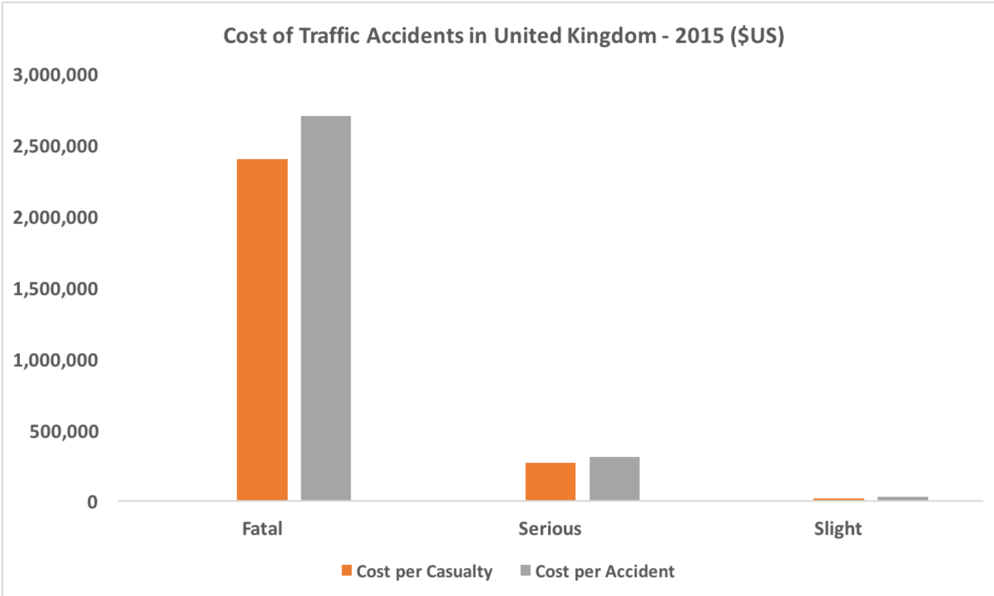


Figure 1. Cost of Traffic Accidents in UK

This brings us to the central concern for this project. If predictors can be identified, then we can determine whether severity depends more on the driver, a certain type of road or intersections, or other conditions or parameters. The value of identifying these predictors is two fold:

- The information can be used by governments, city planners, road engineers in designing safer roads
- The information can be used by insurance to measure risk and assign liability

Therefore, the project had the following goals:

- Identify predictors of road accidents
- Predict severity of road accidents
- Determine the best predictive model of accident severity

Hypothesis

In consideration of these goals, the governing hypothesis for this project is stated:

H_o: Factors can be identified for use in a model that can predict the severity of an accident at a rate greater than chance, i.e. 50%.

H_a: Factors provide no ability to predict accident severity.

To satisfy the goals of the project and allow a rigorous testing of the hypothesis, a robust dataset was need. A dataset with at least 100K records and 30 columns (predictors) was desired. A suitable dataset for this project was found on Kaggle.com, titled “Road Accidents Incidence”. The data, in a “.csv” file format, was described as data collected about road accidents in the United Kingdom (specifically England and Wales) from 1979-2015. A “Road Accidents Safety Data Guide” (.csv), which provided the definitions and categorical information of many of the columns, was also available with the data set. The numeric assignments of the categorical factors were found in this file.

Table 1 shows the basic characteristics of the raw dataset.

Table 1. Road Accidents Incidence - Basic Characteristics

Rows	285,331
Columns	70
Type of Data	Categorical/Ordinal, Numeric
Form of Data	Numeric

Each row is an accident report, and the columns are various data collected that characterize the details of each accident. The variables (columns) were in three broad categories of information:: 1) accident circumstances, 2) vehicle, and 3) casualty. Appendix A lists the variables. The dataset was found to be sufficiently complete and with enough variables of the right types of information on road accidents to commence Data Preparation.

II. Data Preparation

The next phase of the project, Data Preparation, is perhaps the most important phase as the team learns about the data, conditions the data, and then surveys the data. This is the most time consuming phase of the project and is extremely important, because if not done properly will lead to planning and building inaccurate models that make results useless and require the data preparation phase to be repeated. Therefore, these processes are done carefully so that we are confident the data is in the right forms, that there is a sufficient amount of rows and columns, and that the right experimental predictors are included to facilitate effective analysis.

The tool used in cleaning the data was Excel. The following steps were conducted in the process of data preparation:

1. Two ‘.csv’ files were constructed. All of the data in the data file, both true numeric and categorical, was in numeric form. Most of the models that would be most appropriate for this analysis required categorical data. The numeric and categorical data fields were

separated into two files. The data guide mentioned earlier contained the key to the numeric codes of the categorical data.

2. Fourteen (14) columns were observed with many “NA” or missing values. NA was very pre-dominant in these columns. With so many of these values, we concluded there was no or very little value of these columns in continuing analysis. Furthermore, many of these columns were either redundant of other columns or would have had little predictive value in any case. Lastly, since 56 columns remained with complete data and judged to be of predictive value, we could afford to remove these columns.

3. Finally, rows were also removed that had missing values. It was not necessary to impute data due to having a sufficient number of complete rows.

At this point, the dataset contained approximately 145K rows and 56 predictors. Appendix B contains the list of predictors. This prepared data was used in the next phase of Model Planning.

III. Model Planning

This phase of the project, Model Planning, involves exploring the data and selecting those variables that will be used in the models. Models are also selected that, based upon an analysis of the data and its structure, are best suited for the data and the project goals.

The tools used in this phase was Excel and R. The following steps were conducted in the process of model planning:

1. The dependent variable for the models was “accident severity”. The data was imbalanced with far more (as one might expect) accidents with “slight” severity over “serious”, which includes fatal.

2. Columns that were duplicative or highly correlated with another column were removed. A correlation routine with an 80% threshold was used to discover highly correlated variables.

3. The list of possible predictors was further reduced by using variable importance, removing columns which were judged to have no predictive value. A few examples were columns for information concerning police attending the accident, an accident index used for record keeping purposes, and geographical data such as longitude and latitude.

4. One columns of the file contained zip codes of the UK, with approximately 24K unique values. We wanted to consider in a broader sense whether location of accidents could be a predictor. However, we discovered that with so many unique values it was creating bias. Therefore, we decided to club the location based on the geography and bring

down the unique values and factor levels, bringing it down to 10 factor levels. This was accomplished manually by looking into the data dictionary of locations, referred to as Lower Layer Super Output Locations (LSOA), which is a geographical area classification scheme used in the UK. Using this scheme, locations were grouped according to proximity to each other. There were some locations which were very sparsely represented or in diverse areas and were grouped together as well.

5. Model selection. Since much of the data was categorical, the appropriate models were chosen. It was decided to do a comprehensive selection of models to ensure a robust analysis. Models selected were:
    a. Decision Tree
    b. Naive Bayes
    c. Random Forest
    d. Boosting
    e. Bagging
    f. Neural Network.

The changes that occured to the size of the data set from Discovery through Model Planning is located in Appendix C.

IV. Model Building

The Model Building phase of the project involves running the models planned for in Model Planning, and analyzing the results. The final dataset was split 70/30, with 70% of the rows allocated to the training set, and 30% to the test set. Ten-fold cross validation was performed on the training data to ensure accurate performance. Threshold adjustments were made to increase the performance (sensitivity score) of each model.

To address the imbalance in the dependent variable for the training dataset, downsampling was attempted without success. The SMOTE package from CARET was used in an attempt to balance the variable, but with poor results. This was indicated for example by much lower sensitivity (ex. 0.04 for Naive Bayes, 0.11 for Decision Tree) in preliminary testing. The ROSE package was used from CARET with better results (0.38 sensitivity in Decision Tree). Thus the dataset treated with ROSE was the final dataset produced to run the models.

For purposes of this project, correct identification of serious accidents is “true positive” identification, correct identification of slight accidents is “true negative” identification, and the error rates are false positives and false negatives. A confusion matrix depicts the numbers of true/false positives and true/false negatives used to calculate the sensitivity and specificity. The following results were obtained from the models. Appendix D contains the R code used in these models.

1. Decision Tree

Figure 2 depicts the full decision tree. The branches split on the variable “number_of_vehicles” at almost every level, and the interpretation of this is uncertain.

The list of variables by order of greatest predictive value:

	Overall
number_of_vehicles	100.0000
vehicle_type	87.4934
vehicle_manoeuvre	74.5451
speed_limit	63.6083
lsoa_of_accident_location	63.2546
urban_or_rural_area	11.4625
age_of_driver	2.2778
junction_detail	0.3038
road_type	0.0000
junction_control	0.0000
light_conditions	0.0000
weather_conditions	0.0000
sex_of_driver	0.0000
pedestrian_crossing_physical_facilities	0.0000
junction_location	0.0000

Table 2 depicts ROC, sensitivity, and specificity for the training data.

Table 2. ROC, Sensitivity, and Specificity - Decision Tree

	ROC	Sensitivity	Specificity
Training	0.6969	0.6243	0.7256

Figure 2.1 Full Decision Tree

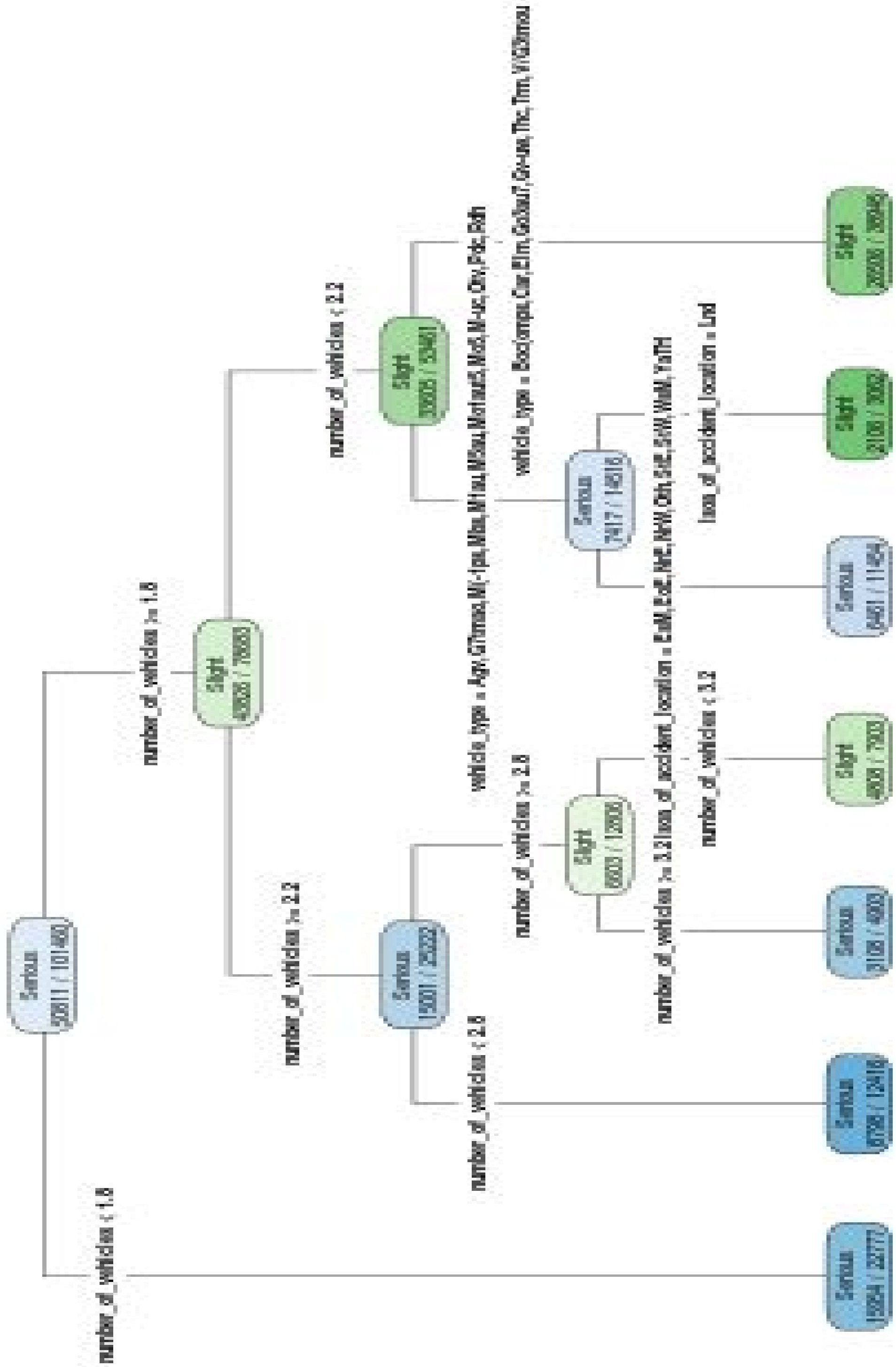


Figure 2.1 Pruned Decision Tree

2. Naive Bayes

The second model conducted was Naive Bayes. Naive Bayes produced the following list of variables by importance:

	Importance
vehicle_type	100.000
vehicle_manoeuvre	85.336
speed_limit	82.873
urban_or_rural_area	80.289
sex_of_driver	66.142
number_of_vehicles	66.094
lsoa_of_accident_location	54.709
age_of_driver	35.475
junction_control	30.709
light_conditions	23.008
pedestrian_crossing_physical_facilities	20.095
road_type	18.183
junction_detail	15.050
weather_conditions	5.365
junction_location	0.000

This list is more interpretable than that of the Decision Tree. Here we see a mix of factors associated with the vehicle, driver, and road specifics.

Table 3 depicts ROC, sensitivity, and specificity for the training data

Table 3. ROC, Sensitivity, and Specificity - Naive Bayes

	ROC	Sensitivity	Specificity
Training	0.7794	.6723	.7413

3. Random Forest

The third model conducted was Random Forest. Random Forest produced the following list of variables by importance:

Overall

number_of_vehicles	100.0000
speed_limit	57.2077
age_of_driver	47.7647
lsoa_of_accident_location	47.1810
vehicle_manoeuvre	40.2315
vehicle_type	32.7466
junction_detail	18.9141
junction_location	18.1077
weather_conditions	10.2311
pedestrian_crossing_physical_facilities	8.8081
road_type	6.7566
light_conditions	6.0491
sex_of_driver	1.0556
urban_or_rural_area	0.4584
junction_control	0.0000

Table 4 depicts ROC, sensitivity, and specificity for both the training data.

Table 4. ROC, Sensitivity, and Specificity - Random Forest

	ROC	Sensitivity	Specificity
Training	0.8876	0.7954	0.8209

4. Boosting

The fourth model conducted was boosting. Boosting produced the following list of variables by importance:

	Importance
vehicle_type	100.000
vehicle_manoeuvre	85.336
speed_limit	82.873
urban_or_rural_area	80.289
sex_of_driver	66.142
number_of_vehicles	66.094
lsoa_of_accident_location	54.709
age_of_driver	35.475
junction_control	30.709
light_conditions	23.008
pedestrian_crossing_physical_facilities	20.095

road_type	18.183
junction_detail	15.050
weather_conditions	5.365
junction_location	0.000

Table 5 depicts ROC, sensitivity, and specificity for both the training data.

Table 5. ROC, Sensitivity, and Specificity - Boosting

	ROC	Sensitivity	Specificity
Training	0.7772	0.6847	0.7250

5. Bagging

The fifth model used was Bagging. Boosting produced the following list of variables by importance:

	Overall
number_of_vehicles	100.000
speed_limit	89.781
age_of_driver	69.626
lsoa_of_accident_location	58.964
vehicle_manoeuvre	50.937
vehicle_type	37.253
junction_location	29.487
junction_detail	26.884
pedestrian_crossing_physical_facilities	12.659
weather_conditions	11.519
road_type	10.277
light_conditions	7.692
urban_or_rural_area	4.044
sex_of_driver	3.414
junction_control	0.000

Table 6 depicts ROC, sensitivity, and specificity for both the training data.

Table 6. ROC, Sensitivity, and Specificity - Bagging

	ROC	Sensitivity	Specificity
Training	0.8413	0.7623	0.7650

6. Neural Network

The sixth and final model used was Neural Network. The resultant network had 1 layer and 3 hidden nodes. Neural Network produced the following list of variables by importance:

	Overall
lsoa_of_accident_location.London	100.000
number_of_vehicles	73.047
sex_of_driver	43.585
lsoa_of_accident_location.South.East	28.767
lsoa_of_accident_location.North.West	24.720
urban_or_rural_area	18.483
vehicle_manoeuvre	16.399
lsoa_of_accident_location.South.West	14.556
lsoa_of_accident_location.North.East	13.935
lsoa_of_accident_location.East.of.England	11.034
weather_conditions	10.770
junction_control	9.019
lsoa_of_accident_location.Yorkshire.and.The.Humber	8.518
lsoa_of_accident_location.West.Midlands	7.789
lsoa_of_accident_location.Others	7.565
light_conditions	7.337
road_type	5.290
vehicle_type	5.192
junction_detail	4.680
junction_location	3.736

Table 7 depicts ROC, sensitivity, and specificity for both the training data.

Table 7. ROC, Sensitivity, and Specificity - Neural Network

	ROC	Sensitivity	Specificity
Training	0.6684	0.6245	0.6236

Results and Performance

Figures 3, 4, and 5 depict the training ROC, sensitivity, and specificity for all of the models.

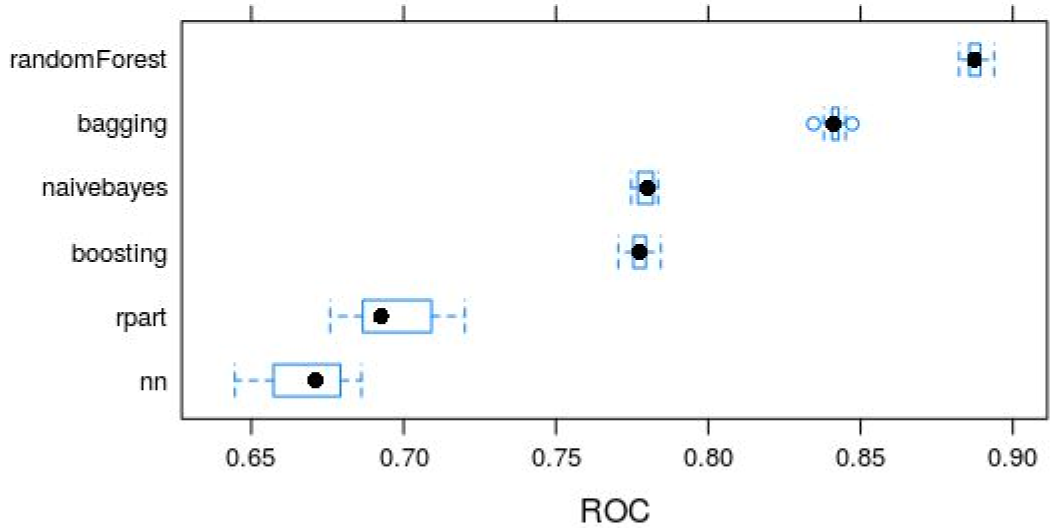


Figure 3. ROC - All Models

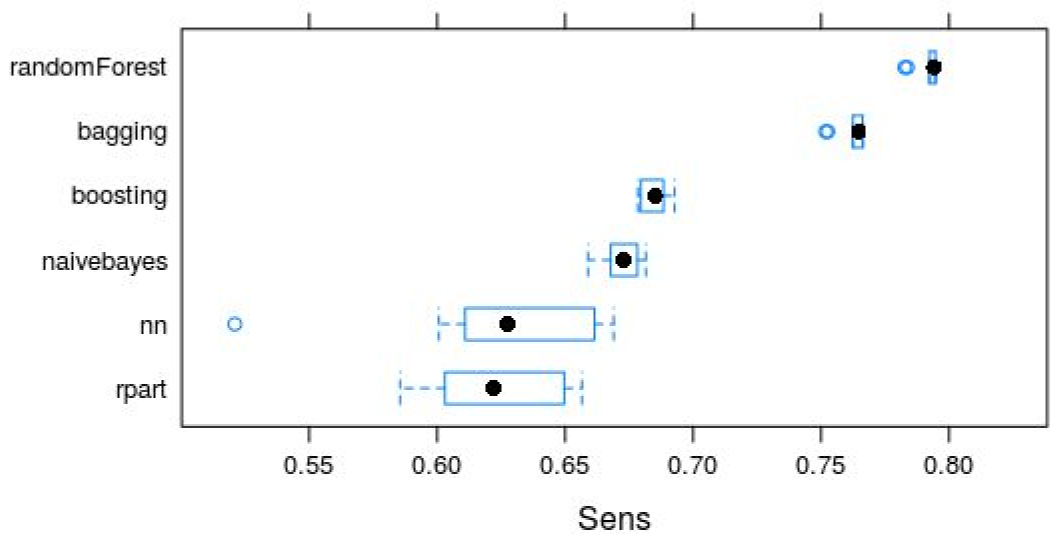


Figure 4. Sensitivity - All Models

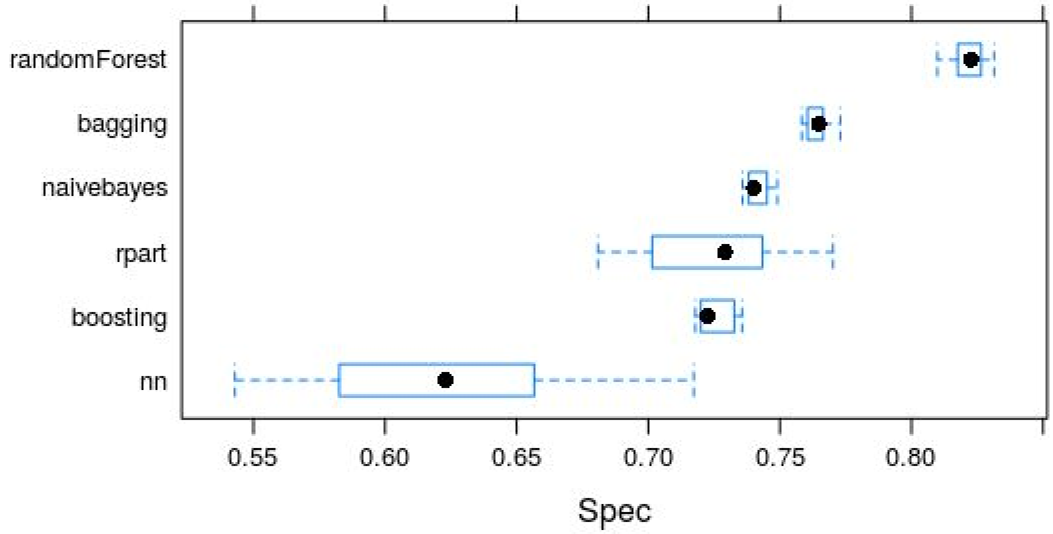


Figure 5. Specificity - All Models

We can see from these graphs that in all three, Random Forest performed the best. As can be seen from the model results above, Random Forest outperformed the other models on the test data as well.

Below are performance results of all the models on test data.

1. **Decision Tree**

Table 8 depicts ROC, sensitivity, and specificity for the test data.

Table 8. ROC, Sensitivity, and Specificity - Decision Tree

	ROC	Sensitivity	Specificity
Test	0.6217	0.4745	0.7485

Figure 6 shows the ROC curve on the test data. Greater area to the right and under the curve indicate better performance. This ROC is mediocre at best.

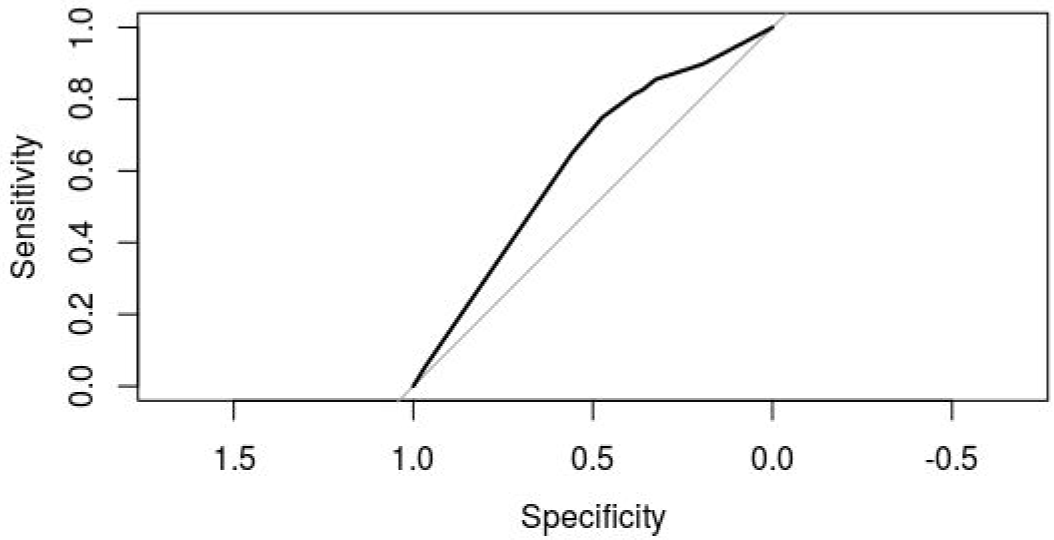


Figure 6. ROC Curve - Decision Tree

Table 9 is the confusion matrix for the decision tree, with Positive (serious) and Negative (slight) predictive values.

Table 9. Confusion Matrix - Decision Tree

Prediction	**Serious**	**Slight**	**Pos/Neg Pred. Value**
Serious	2709	9501	.2219
Slight	3000	28272	.9041

Accuracy was 0.7125 and balanced accuracy was 0.6115 respectively. While these accuracies are above our target of 0.5, what is a better indication of the performance of the model is the sensitivity score of the test data, which was 0.4745, and positive predictive value only .2219. Even though negative predictive value was high, .9041, we are more concerned with being able to correctly identify under what conditions serious accidents are likely to occur. We can see that this is below 0.5. This, in combination with the way the nodes split in the tree as mentioned above, do not enable a clear interpretation of the importance of the variables.

2. Naive Bayes

Table 10 depicts ROC, sensitivity, and specificity for both the test data.

Table 10. ROC, Sensitivity, and Specificity - Naive Bayes

	ROC	Sensitivity	Specificity
Test	0.6918	0.4773	0.7737

Figure 7 shows the ROC curve on the test data. Here we see a slight improvement over the Decision Tree, though still not extremely accurate.

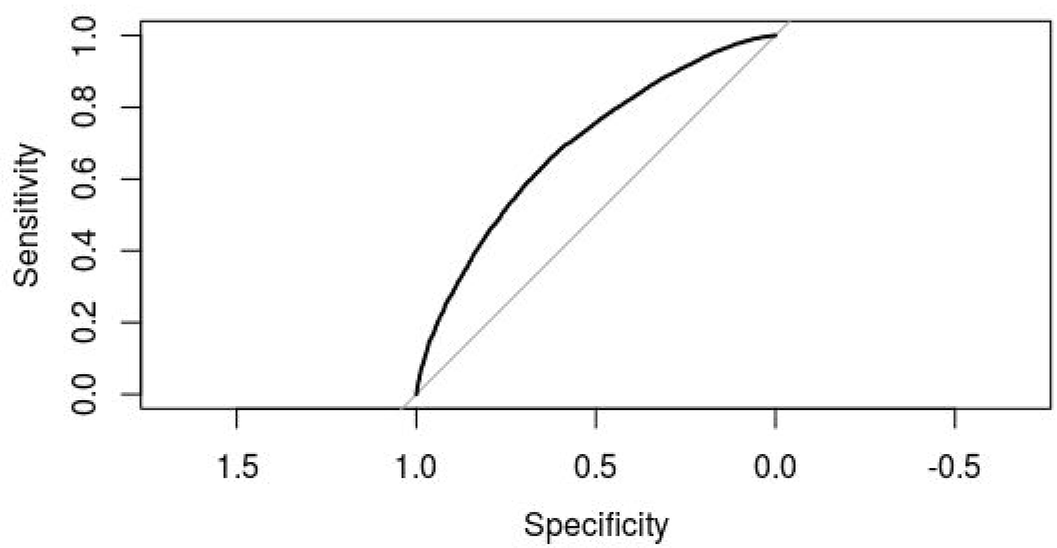


Figure 7. ROC Curve - Naive Bayes

Table 11 is the confusion matrix for Naive Bayes.

Table 11. Confusion Matrix - Naive Bayes

Prediction	**Serious**	**Slight**	**Pos/Neg Pred. Value**
Serious	2725	8548	.2417
Slight	2984	29225	.9074

Accuracy was 0.7348 and balanced accuracy was 0.62551 respectively. Sensitivity was also slightly better than Decision Tree at 0.4773, still below what is desired. The predictive values were similarly slightly higher. While variable importance was more interpretable, performance is not high enough to meet project goals.

3. Random Forest
Table 12 depicts ROC, sensitivity, and specificity for the test data.

Table 12. ROC, Sensitivity, and Specificity - Random Forest

	ROC	Sensitivity	Specificity
Test	0.7357	0.6597	0.6904

Here we see a significant improvement over the Decision Tree and Naive Bayes. Figure 8 shows the ROC curve on the test data.

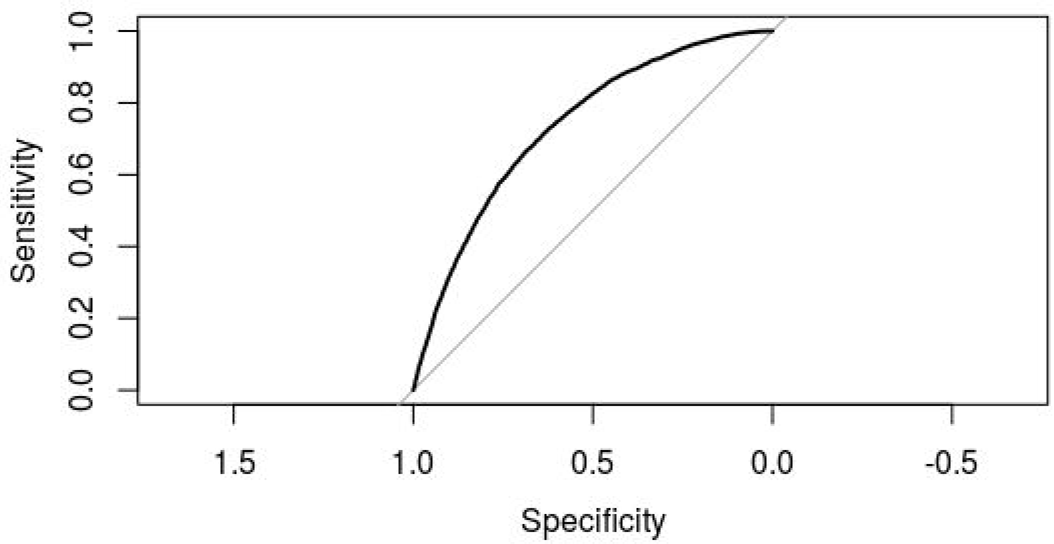


Figure 8. ROC Curve - Random Forest

Table 13 is the confusion matrix for Random Forest.

Table 13. Confusion Matrix - Random Forest

Prediction	**Serious**	**Slight**	**Pos/Neg Pred. Value**
Serious	3766	11695	0.2436
Slight	1943	26078	0.9307

Accuracy was 0.6864, and balanced accuracy was 0.6750 respectively. Sensitivity was greatly improved over the two previous models at 0.6597, above our goal of 0.5. Variable importance was more interpretable and performance met the goal on the conditional that overall accuracy needs to improve.

4. Boosting

Table 14 depicts ROC, sensitivity, and specificity for both the test data.

Table 14. ROC, Sensitivity, and Specificity - Boosting

	ROC	Sensitivity	Specificity
Test	0.6955	0.5989	0.6848

Figure 9 shows the ROC curve on the test data.

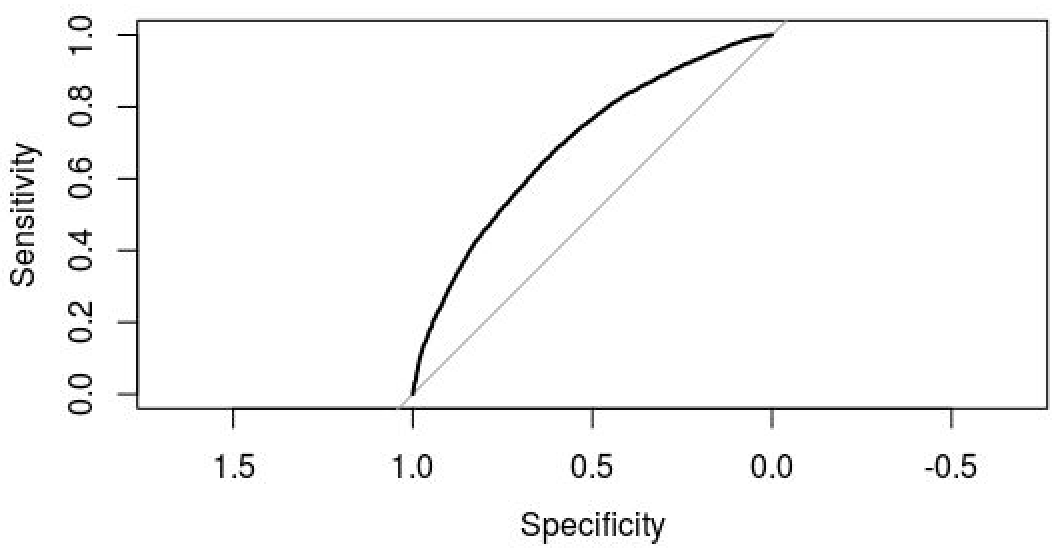


Figure 9. ROC Curve - Boosting

Table 15 is the confusion matrix for Boosting.

Table 15. Confusion Matrix - Boosting

Prediction	**Serious**	**Slight**	**Pos/Neg Pred. Value**
Serious	3419	11908	0.2231
Slight	2290	25865	0.9187

Accuracy was 0.6735, and balanced accuracy was 0.6418 respectively. Sensitivity was above 0.5, although at 0.5989 lower than with Random Forest. Variable importance was interpretable and performance met the goal on the conditional that overall accuracy needs to improve.

5. Bagging

Table 16 depicts ROC, sensitivity, and specificity for the test sdata.

Table 16. ROC, Sensitivity, and Specificity - Bagging

	ROC	Sensitivity	Specificity
Test	0.7045	0.6080	0.6872

Figure 10 shows the ROC curve on the test data.

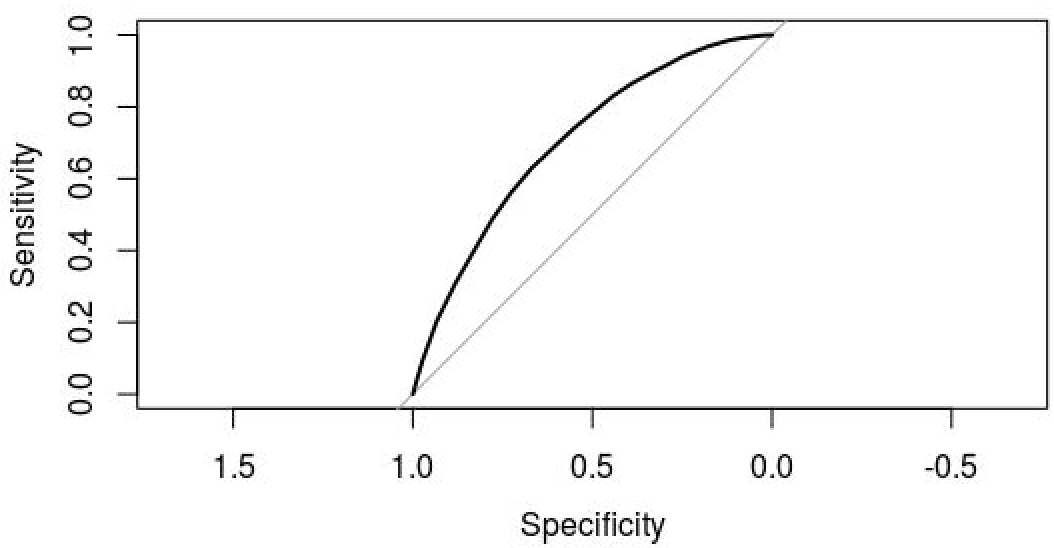


Figure 10. ROC Curve - Bagging

Table 17 is the confusion matrix for Bagging.

Table 17. Confusion Matrix - Bagging

Prediction	**Serious**	**Slight**	**Pos/Neg Pred. Value**
Serious	3471	11817	0.2270

Slight	2238	25956	0.9206

Accuracy was 0.6768, and balanced accuracy was 0.6476. Sensitivity was above 0.5, at 0.6080. Variable importance was interpretable and performance met the goal on the conditional that overall accuracy needs to improve.

6. Neural Network

Table 18 depicts ROC, sensitivity, and specificity for the test data.

Table 18. ROC, Sensitivity, and Specificity - Neural Network

	ROC	Sensitivity	Specificity
Test		0.5949	0.5991

Figure 11 shows the ROC curve on the test data. We can see that the performance is less than the other models, and similar to Decision Tree.

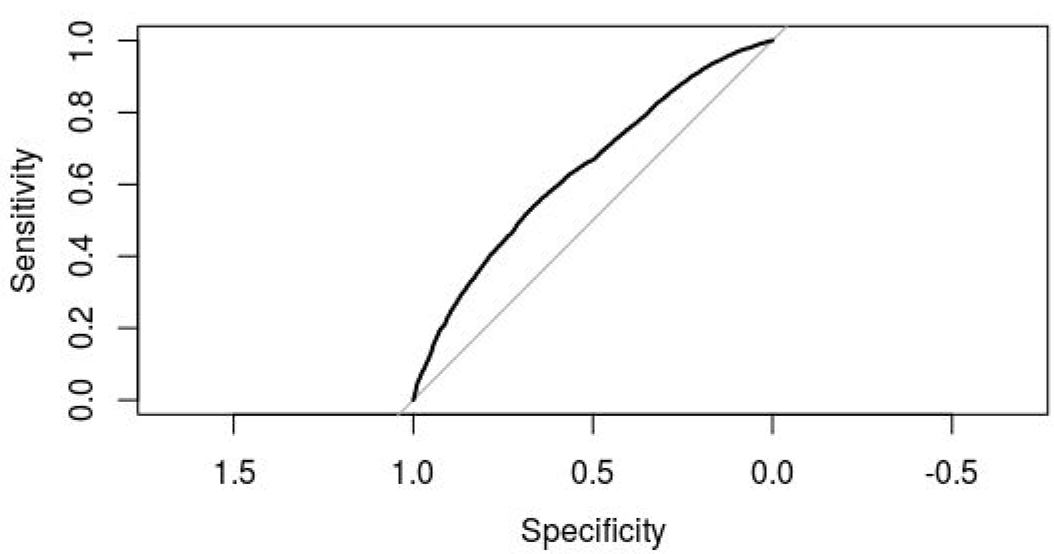


Figure 11. ROC Curve - Neural Network

Table 19 is the confusion matrix.

Table 19. Confusion Matrix - Neural Network

Prediction	Serious	Slight	Pos/Neg Pred. Value
Serious	3396	15144	0.1832
Slight	2313	22629	0.9073

Accuracy was 0.5985, and balanced accuracy was 0.5970. Sensitivity was above 0.5 at 0.5949. Variable importance was interpretable and performance met the goal on the conditional that overall accuracy needs to improve.

From the performance of each model described above, Random Forest also produced the best results on the test data. Overall accuracy was not the highest, but accuracy alone does not indicate the best model. Since, as has already been mentioned, we are more concerned with sensitivity, with Random Forest we have the highest performance and it is greater than chance (0.5).

The top five predictors from all of the models were a mix of eight of the fifteen predictors. They were (with the number of occurrences in the top five): Vehicle Manoeuvre (6), Speed Limit (5), # of Vehicles (4), Location (4), Driver Gender (3), Vehicle Type (3), Urban/Rural (3), and Driver Age (2). No predictors related to road design and conditions, except for speed limit, were in the top five, but did contribute to the most of the models to some degree. It is unclear whether the ones identified would be useful in the ways stated earlier as meeting the value proposition of the analysis.

This project demonstrated that a model can be built that satisfies the goals of the project. A model was able to produce predictive performance of accident severity at a rate greater than chance (0.5). Performance was also measured and Random Forest was identified as the best predictive model. The project satisfied the study's hypothesis.

Recommendation

The results of this project show that patterns can be derived from the data to enable a predictive model to be built. The following are recommendations for further study:

1. Continue threshold adjustments in an effort to improve sensitivity and positive predictive value.
2. Use different combinations of predictors, including removing predictors, or adding others to the model. This could enable better interpretation, and result in more valuable information.

3. Obtain data with additional variables. One variable that was absent was the speed of the vehicle at the time of the accident. There could be other factors in other studies, or more complete information.

.

References

Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data. (2015). Hoboken: John Wiley & Sons.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An Introduction to Statistical Learning: with Applications in R.* New York: Springer.

Appendix A - Initial Variables List (raw data)

Accident Circumstances	**Vehicle**	**Casualty**
Accident Index	Accident Index	Accident Index
Police Force	Vehicle Reference	Vehicle Reference
Accident Severity	Vehicle Type	Casualty Reference
Number of Vehicles	Towing and Articulation	Casualty Class
Number of Casualties	Vehicle Manoeuvre	Sex of Casualty
Date (DD/MM/YYYY)	Vehicle Location-Restricted Lane	Age of Casualty
Day of Week	Junction Location	Age Band of Casualty
Time (HH:MM)	Skidding and Overturning	Casualty Severity
Location Easting OSGR (Null if not known)	Hit Object in Carriageway	Pedestrian Location
Location Northing OSGR (Null if not known)	Vehicle Leaving Carriageway	Pedestrian Movement
Longitude (Null if not known)	Hit Object off Carriageway	Car Passenger
Latitude (Null if not known)	1st Point of Impact	Bus or Coach Passenger
Local Authority (District)	Was Vehicle Left Hand Drive	Pedestrian Road Maintenance Worker (From 2011)
Local Authority (Highway Authority - ONS code)	Journey Purpose of Driver	Casualty Type
1st Road Class	Sex of Driver	Casualty IMD Decile
1st Road Number	Age of Driver	Casualty Home Area Type
Road Type	Age Band of Driver	
Speed limit	Engine Capacity	
Junction Detail	Vehicle Propulsion Code	
Junction Control	Age of Vehicle (manufacture)	
2nd Road Class	Driver IMD Decile	

2nd Road Number	Driver Home Area Type	
Pedestrian Crossing-Human Control		
Pedestrian Crossing-Physical Facilities		
Light Conditions		
Weather Conditions		
Road Surface Conditions		
Special Conditions at Site		
Carriageway Hazards		
Urban or Rural Area		
Did Police Officer Attend Scene of Accident		
Lower Super Output Area of Accident_Location (England & Wales only)		

Appendix B - Final Variables List (models)

Accident Circumstances	**Vehicle**
Number of Vehicles	Age of Driver
Speed Limit	Sex of Driver
LSOA of Accident Location	Vehicle Manoeuvre
Junction Detail	Vehicle Type
Junction Control	Junction Location
Pedestrian Crossing - Physical Facilities	
Weather Conditions	
Road Type	
Light Conditions	
Urban or Rural Area	

Appendix C

	Number of Rows	Number of Columns
Raw data file	285331	70
Removed columns with predominant "NA"	285331	56
Removed rows with missing values	144942	56
Applied correlation analysis	144942	23
Determined variable importance and eliminated least important variables	144942	16
Divided data into training and test sets		
Test data (30%)	43382	16
Train data (70%)	101560	16
Data Balancing		
After running SMOTE on Training Data (Tried initially but while modelling sensitivity was too low) - Discarded	129404	16
After running ROSE on Training Data	101460	16

Appendix D - R Code

```
## SMOTE Part
```

```
#lets install all the packages we will sue up front to avoid conflicts

install.packages(c("ipred","parallel","iterators","lattice","ggplot2","doParallel", "caret", "mice",
"pROC", "VIM", "ipred", "ada", "randomForest"))
install.packages("caret")
library("lattice")
library("ggplot2")
library("iterators")
library("parallel")
library("doParallel")
library("ipred")
library("caret")
library("pROC")

library(mice) #imputation package will discuss later
library(VIM)

# Read the data
#set working folder to location of data files
setwd("~/Accidents files")

accidents <- read.csv("Accident_CSV_only_categorical_data.csv")
ncol(accidents)
nrow(accidents)
str(accidents)


inTrain<-createDataPartition(y=accidents$accident_severity, p=.80, list=FALSE)
nrow(inTrain)
accidents$accident_severity

imbal_accident_train <- accidents[(inTrain),]
str(imbal_accident_train)
nrow(imbal_accident_train)
imbal_accident_test <- accidents[(-inTrain),]
str(imbal_accident_test)
nrow(imbal_accident_test)
imbal_accident_train

table(imbal_accident_train$accident_severity)

install.packages("DMwR")
install.packages("grid")
```

```
library(grid)
library(DMwR)

#hybrid both up and down
set.seed(192)
str(imbal_accident_train)
summary(imbal_accident_train)

smote_train <- SMOTE(accident_severity~ ., imbal_accident_train, perc.over = 270,
perc.under=200)
table(smote_train$accident_severity)

## write the downsampled data in a csv file
write.table(smote_train, "~/Accidents files/smote_train.csv", sep=",")

write.table(imbal_accident_test, "~/Accidents files/test_data.csv", sep=",")


## ROSE Part

# Read the data
#set working folder to location of data files
setwd("~/Accidents files")

accidents <- read.csv("Accident_CSV_only_categorical_data.csv")
ncol(accidents)
nrow(accidents)
str(accidents)


inTrain<-createDataPartition(y=accidents$accident_severity, p=.70, list=FALSE)
nrow(inTrain)
accidents$accident_severity

imbal_accident_train <- accidents[(inTrain),]
str(imbal_accident_train)
nrow(imbal_accident_train)
imbal_accident_test <- accidents[(-inTrain),]
str(imbal_accident_test)
nrow(imbal_accident_test)
imbal_accident_train
```

```
table(imbal_accident_train$accident_severity)

#install.packages("DMwR")
#install.packages("grid")

#library(grid)
#library(DMwR)

#hybrid both up and down
set.seed(192)
str(imbal_accident_train)
summary(imbal_accident_train)

#smote_train <- SMOTE(accident_severity~ ., imbal_accident_train, perc.over = 270,
perc.under=200)

install.packages("ROSE")


library(ROSE)

set.seed(192)
rose_train <-ROSE(accident_severity ~ ., data  = imbal_accident_train)$data
table(rose_train$accident_severity)

#table(smote_train$accident_severity)

## write the downsampled data in a csv file
write.table(rose_train, "~/Accidents files/rose_train.csv", sep=",")

write.table(imbal_accident_test, "~/Accidents files/test_data.csv", sep=",")


##Main Code for Models and Performance

#lets install all the packages we will sue up front to avoid conflicts

install.packages(c("ipred","parallel","iterators","lattice","ggplot2","doParallel", "caret", "mice",
"pROC", "VIM", "ipred", "ada", "randomForest"))

library(lattice)
library(ggplot2)
library(caret)
```

```
library(pROC)
library(mice)
library(Rcpp)
library(doParallel)
library(MASS)
library(kernlab)
library(e1071)
library(ISLR)
library(rpart)
library(tree)
library(randomForest)
library(klaR)
library(survival)
library(dplyr)
library(plyr)
library(gbm)
library(mgcv)
library(nlme)
library(rpart.plot)

library(mice) #imputation package will discuss later
library(VIM)

# Read the data
#set working folder to location of data files
setwd("~/Accidents files")

accident_train<- read.csv("rose_train.csv")
head(accident_train)
summary(accident_train)
ncol(accident_train)
nrow(accident_train)
str(accident_train)

accident_test <- read.csv("test_data.csv")
head(accident_test)
summary(accident_test)
ncol(accident_test)
nrow(accident_test)
str(accident_test)

#with 15 variables
```

```
drops<-c("road_surface_conditions", "pedestrian_crossing_human_control",
"carriageway_hazards",
        "vehicle_location_restricted_lane", "day_of_week", "was_vehicle_left_hand_drive",
"special_conditions_at_site")

accident_train <- accident_train [,!(names(accident_train) %in% drops)]
colnames(accident_train)

## Create train data split for DV and Predictors
y.train <- accident_train$accident_severity
x.train <- accident_train [,-ncol(accident_train)]
head(y.train)
colnames(x.train)
## Create test data split for DV and Predictors
y.test <- accident_test$accident_severity
x.test <- accident_test [,-ncol(accident_test)]
head(y.test)
colnames(x.test)

setdiff(levels(x.train$lsoa_of_accident_location),
      levels(x.test$lsoa_of_accident_location))



library("doParallel")
## to run parallel with 8 cores
cl <- makeCluster(8)
registerDoParallel(cl)

##lets start modeling

#some parameters to control the sampling during parameter tuning and testing
#10 fold crossvalidation, using 10-folds instead of 10 to reduce computation time in class demo,
use 10 and with more computation to spare use
#repeated cv
ctrl <- trainControl(method="cv", number=10,
                classProbs=TRUE,
                #function used to measure performance
                summaryFunction = twoClassSummary, #multiClassSummary for non binary
                allowParallel =  TRUE)


set.seed(192)
```

```
library(rpart)
set.seed(192)

m.rpart <- train(y=y.train, x=x.train,
             method = "rpart",tuneLength=7,
             metric = "ROC",
             trControl = ctrl)

getTrainPerf(m.rpart)
varImp(m.rpart)

##ROC of Train data
#dt_train.roc<-roc(response=y.train,predictor=x.train$)

#confusionMatrix(m.rpart,y.train) #calc accuracies with confuction matrix on downsampled
training data set


#the best performing model trained on the full training set is saved
##preprocessing using predict function with caret train object will be applied to new data
p.rpart <- predict(m.rpart,x.test)
p.rpart.prob <- predict(m.rpart, x.test, type="prob")



confusionMatrix(p.rpart,y.test) #calc accuracies with confuction matrix on test set

p.rpart.newthresh <- factor(ifelse(p.rpart.prob[[1]]>0.37, "Serious", "Slight"))
p.rpart.newthresh
confusionMatrix(p.rpart.newthresh, y.test)

test.rpart.roc<- roc(response= y.test, predictor= p.rpart.prob[[1]])
plot(test.rpart.roc)

plot(test.rpart.roc)
newthresh<- coords(test.rpart.roc, x="best", best.method="closest.topleft")

p.rpart.prob[[1]]
levels(p.rpart.newthresh)
plot(p.rpart)
```

```
fit <- rpart(y.train~. ,data=x.train,
            method="class",
            control=rpart.control(minsplit=1),
            parms=list(split='information'))
plot(fit)
library(rpart.plot)
rpart.plot(fit, type=4, extra=2,clip.right.labs=FALSE, varlen=0, faclen=3)
rpart.plot(fit)


printcp(fit) #display crossvalidated error for each tree size
plotcp(fit) #plot cv error


auc(test.rpart.roc)

#we can grab this from the plotcp table automatically with
opt.cp <- fit$cptable[which.min(fit$cptable[,"xerror"]),"CP"]

#lets prune the tree
fit.pruned <- prune(fit,cp=0.016911)

#lets review the final tree
rpart.plot(fit.pruned)


##Naive Bayes

m.nb <- train(y=y.train, x=x.train,
             trControl = ctrl,
             metric = "ROC", #using AUC to find best performing parameters
             method = "nb")
m.nb

getTrainPerf(m.nb)
varImp(m.nb)
plot(m.nb)
#confusionMatrix(m.nb,y.train)


p.nb<- predict(m.nb,x.test)
p.nb.prob<- predict(m.nb,x.test,type="prob")
plot(p.nb)
```

```
confusionMatrix(p.nb,y.test) #calc accuracies with confuction matrix on test set

p.nb.newthresh <- factor(ifelse(p.nb.prob[[1]]>0.35, "Serious", "Slight"))
p.nb.newthresh
confusionMatrix(p.nb.newthresh, y.test)


test.nb.roc<- roc(response= y.test, predictor= p.nb.prob[[1]])
plot(test.nb.roc)
newthresh<- coords(test.rpart.roc, x="best", best.method="closest.topleft")

p.rpart.prob[[1]]
levels(p.rpart.newthresh)
plot(p.rpart)

auc(test.nb.roc)

## Boosting

install.packages("rpart")
library(ada)
set.seed(192)
#boosted decision trees
#using dummy codeds because this function internally does it and its better to handle it yourself
(i.e., less error prone)

m.ada <- train(y=y.train, x=x.train,
            trControl = ctrl,
            metric = "ROC", #using AUC to find best performing parameters
            method = "ada")


m.ada
getTrainPerf(m.ada)
varImp(m.ada)
plot(m.ada)
p.ada<- predict(m.ada,x.test)
confusionMatrix(p.ada,y.test)

p.ada.prob <- predict(m.ada, x.test, type="prob")
p.ada.newthresh <- factor(ifelse(p.ada.prob[[1]]>0.40, "Serious", "Slight"))
p.ada.newthresh
confusionMatrix(p.ada.newthresh, y.test)
```

```
test.ada.roc<- roc(response= y.test, predictor= p.ada.prob[[1]])
plot(test.ada.roc)

p.ada.prob[[1]]
levels(p.ada.newthresh)
plot(p.ada)

auc(test.ada.roc)

##Random Forest
#random forest approach to many classification models created and voted on
#less prone to ovrefitting and used on large datasets
library(randomForest)

m.rf <- train(y=y.train, x=x.train,
        trControl = ctrl, probmodel=TRUE,forClass=TRUE,
        metric = "ROC", #using AUC to find best performing parameters
        method = c("rf") )
m.rf
getTrainPerf(m.rf)
varImp(m.rf)
plot(m.rf)

p.rf<- predict(m.rf,x.test)
plot(p.rf)
confusionMatrix(p.rf,y.test)

p.rf.prob<- predict(m.rf,x.test,type="prob")
p.rf.newthresh <- factor(ifelse(p.rf.prob[[1]]>0.31, "Serious", "Slight"))
p.rf.newthresh
confusionMatrix(p.rf.newthresh, y.test)

test.rf.roc<- roc(response= y.test, predictor= p.rf.prob[[1]])
plot(test.rf.roc)

p.rf.prob[[1]]
levels(p.rf.newthresh)

auc(test.rf.roc)


## Bagging
```

```
library(ipred)
set.seed(192)

m.bag <- train(y=y.train, x=x.train,
            trControl = ctrl,
            metric = "ROC", #using AUC to find best performing parameters
            method = "treebag")
m.bag

getTrainPerf(m.bag)
varImp(m.bag)
plot(m.bag)

p.bag<- predict(m.bag,x.test)
plot(p.bag)
confusionMatrix(p.bag,y.test)

p.bag.prob<- predict(m.bag,x.test,type="prob")
p.bag.newthresh <- factor(ifelse(p.bag.prob[[1]]>0.30, "Serious", "Slight"))
p.bag.newthresh
confusionMatrix(p.bag.newthresh, y.test)

test.bag.roc<- roc(response= y.test, predictor= p.bag.prob[[1]])
plot(test.bag.roc)

p.bag.prob[[1]]
levels(p.bag.newthresh)

auc(test.bag.roc)


##Neural Netwrok

# Read the data
#set working folder to location of data files
setwd("~/Accidents files")

acc_num <- read.csv("Accident_CSV_only_numeric_data_Neural_network.csv")
ncol(acc_num)
nrow(acc_num)
str(acc_num)
```

```
inTrain_num<-createDataPartition(y=acc_num$accident_severity, p=.70, list=FALSE)
nrow(inTrain_num)
acc_num$accident_severity

imbal_accident_train_num <- acc_num[(inTrain_num),]
str(imbal_accident_train_num)
nrow(imbal_accident_train_num)
imbal_accident_test_num <- acc_num[(-inTrain_num),]
str(imbal_accident_test_num)
nrow(imbal_accident_test_num)
nrow(imbal_accident_test_num)
imbal_accident_train_num

table(imbal_accident_train_num$accident_severity)

#hybrid both up and down
set.seed(192)
str(imbal_accident_train_num)
summary(imbal_accident_train_num)

install.packages("ROSE")

library(ROSE)

set.seed(192)
rose_train_num <-ROSE(accident_severity ~ ., data  = imbal_accident_train_num)$data
table(rose_train_num$accident_severity)

#table(smote_train$accident_severity)

## write the downsampled data in a csv file
write.table(rose_train_num, "~/Accidents files/rose_train_num.csv", sep=",")

write.table(imbal_accident_test_num, "~/Accidents files/test_data_num.csv", sep=",")


install.packages("neuralnet")
install.packages("devtools")
install.packages("NeuralNetTools")
install.packages("ggplot2")

library("NeuralNetTools")
```

```
library("devtools")
library("neuralnet")

accident_train_num<- read.csv("rose_train_num.csv")
head(accident_train_num)
summary(accident_train_num)
ncol(accident_train_num)
nrow(accident_train_num)
str(accident_train_num)
table(accident_train_num$accident_severity)

accident_test_num <- read.csv("test_data_num.csv")
head(accident_test_num)
summary(accident_test_num)
ncol(accident_test_num)
nrow(accident_test_num)
str(accident_test_num)
table(accident_test_num$accident_severity)


#with 15 variables

drops<-c("road_surface_conditions", "pedestrian_crossing_human_control",
"carriageway_hazards",
        "vehicle_location_restricted_lane", "day_of_week", "was_vehicle_left_hand_drive",
"special_conditions_at_site")

accident_train_num <- accident_train_num [,!(names(accident_train_num) %in% drops)]
colnames(accident_train_num)


## Create train data split for DV and Predictors
y.train_num <- accident_train_num$accident_severity
x.train_num <- accident_train_num [,-ncol(accident_train_num)]

dummy_accident_xtrain <- dummyVars(" ~ .", data = x.train_num)
x.train_num1 <- data.frame(predict(dummy_accident_xtrain, newdata = x.train_num))
#print(trsf)
head(x.train_num1)
names(x.train_num1)


head(y.train_num)
```

```
colnames(x.train_num1)

accident_test_num <- accident_test_num [,!(names(accident_test_num) %in% drops)]

#dummy_accident_test_num <- dummyVars(" ~ .", data = accident_test_num)
#accident_test_num1 <- data.frame(predict(dummy_accident_test_num, newdata =
accident_test_num))
## Create test data split for DV and Predictors
y.test_num <- accident_test_num$accident_severity
x.test_num <- accident_test_num [,-ncol(accident_test_num)]
names(x.test_num)
dummy_accident_xtest1 <- dummyVars(" ~ .", data = x.test_num)
x.test_num1 <- data.frame(predict(dummy_accident_xtest1, newdata = x.test_num))
#print(trsf)
#x.test_num1
head(y.test_num)
colnames(x.test_num1)
names(x.test_num1)

#setdiff(levels(x.train_num$lsoa_of_accident_location),
#       levels(x.test_num$lsoa_of_accident_location))



#library("doParallel")
## to run parallel with 8 cores
#cl <- makeCluster(8)
#registerDoParallel(cl)

##lets start modeling

#some parameters to control the sampling during parameter tuning and testing
#10 fold crossvalidation, using 10-folds instead of 10 to reduce computation time in class demo,
use 10 and with more computation to spare use
#repeated cv
ctrl <- trainControl(method="cv", number=10,
                     classProbs=TRUE,
                     #function used to measure performance
                     summaryFunction = twoClassSummary, #multiClassSummary for non binary
                     allowParallel =  TRUE)


set.seed(192)
```

```
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}


accident_norm_train <- as.data.frame(lapply(x.train_num1, normalize))
accident_norm_test <- as.data.frame(lapply(x.test_num1, normalize))
accident_norm_train_total<-cbind(x.train_num1,y.train_num)
head(accident_norm_train_total)
set.seed(192)



nn <- train(y=y.train_num, x=x.train_num1,
        method = "nnet",tuneLength=2,
        metric = "ROC",
        trControl = ctrl)


getTrainPerf(nn)
varImp(nn)

plot(nn)
plotnet(nn)


p.nn<- predict(nn,x.test_num1)
plot(p.nn)
confusionMatrix(p.nn,y.test_num)

p.nn.prob <- predict(nn, x.test_num1, type="prob")
test.nn.roc<- roc(response= y.test_num, predictor= p.nn.prob[[1]])
plot(test.nn.roc)
p.nn.prob[[1]]
auc(test.nn.roc)

#compare training performance
#create list of cross validation runs (resamples)
rValues <- resamples(list(rpart=m.rpart, naivebayes=m.nb, randomForest=m.rf, bagging=m.bag,
boosting=m.ada, nn=nn))
```

```
#create plot comparing them
bwplot(rValues, metric="ROC")
bwplot(rValues, metric="Sens") #Sensitvity
bwplot(rValues, metric="Spec")

#create dot plot comparing them
dotplot(rValues, metric="ROC")
dotplot(rValues, metric="Sens") #Sensitvity
dotplot(rValues, metric="Spec")

xyplot(rValues, metric="ROC")
xyplot(rValues, metric="Sens") #Sensitvity
xyplot(rValues, metric="Spec")

summary(rValues)

#using no probability as positive class
rpart.roc<- roc(y.test, rpart.prob$Serious)
nb.roc<- roc(y.test, nb.prob$no)


#lets see auc
auc(rpart.roc)
auc(nb.roc)
```