Vehicle Crash Survivability

Najah Moussa
24 August 2020

Introduction

Automotive accidents result in over 30,000 fatalities in the United States annually [1]. The National Automotive Sampling System (NASS) provides a nationally representative sample of police reported collisions and is made available to researchers and the general public.

The research question for this project is to identify and quantify factors which impact the survivability of various crash types (rear-end, sideswipe, etc) using R, and create a web app using the shiny package to predict survivability for given inputs using regression.

The techniques will include web-scraping the publicly available data on the NASS website, parsing the resultant XML and data cleaning of real-world dataset, exploratory analysis to identify relevant factors, feature engineering, and regression.

The source code for this project is available on github at https://github.com/gmyrland/capstone_project.

References:

[1] World Health Organization. (2015). Global status report on road safety 2015. Accessed from http://www.who.int/violence_injury_prevention/road_safety_status/2015/TableA2.pdf?ua=1

Literature Review

Several publications were reviewed with emphasis being placed on determining potential factors which may have significant effects on vehicle crash survivability.

An Indiana University paper (2014) noted that vehicle inequalities (e.g., height, rigidity, weight) had a significant impact on survivability in head-on collisions. This driver survival risk factor study found that "the driver's chance of survival was increased by driving a vehicle with a higher mass, driving a newer vehicle, being younger, being a male, using a seatbelt and having the airbag deployed in the crash." [2]

Some studies examined the effect of vehicle age on survivability. For example, an Association for the Advancement of Automotive Medicine study (2006) showed decreases in the casualty rate for newer cars in frontal impacts. [3]

A 2014 conference paper examined the risk factors associated with the survival of drivers in head on collisions. In order to control for vehicle speed, vehicles involved in head-on collisions were paired and logistic regression was used to model the effect of other factors such as vehicle mass, vehicle age, and passenger demographics. [4]

Finally, the World Health Organization report on road traffic injury prevention (2004) identified speed as a key risk factor in road traffic injuries. Further, driver speed choice was found to be influenced by a number of factors, including: driver-related factors such as age, gender, alcohol level, and number of people in the vehicle; road and vehicle factors such as road layout, surface quality, vehicle power, and maximum speed; and traffic- and environment-related such as traffic density and composition, prevailing speed, and weather conditions. [5]

References:

[2] Indiana University. (2014). Car crash survival rates increase with being younger, male and driving a big vehicle. Accessed from http://www.eurekalert.org/pub_releases/2014-11/iu-ccs111814.php

[3] Frampton, R., Page, M., & Thomas, P. (2006). Factors Related to Fatal Injury in Frontal Crashes Involving European Cars. Annual Proceedings / Association for the Advancement of Automotive Medicine, 50, 35-56.

[4] Kirbiyik, U., Dixon, B., & Zollinger, T.W. (2014). Factors affecting survival in head-on vehicle collisions. 142nd APHA Annual Meeting and Exposition 2014. Accessed from https://www.researchgate.net/publication/266775960_Factors_affecting_survival_in_head-on_vehicle_collisions

[5] World Health Organiztion. (2004). World report on road traffic injury prevention. Accessed from http://www.who.int/violence_injury_prevention/publications/road_traffic/world_report/speed_en.pdf

Dataset

The data used for this project can be found at http://www.nhtsa.gov/NASS. The section "*NASS CDS Case Viewer - XML Viewer (2004-Present)*" provides a search interface of the existing case data. When a case id is known, it can be used to extract XML data for the specific collision, allowing for collection of all case data.

> "Information collected in NASS, with all personal identifiers removed, is made available to other researchers and organizations involved in the highway safety effort. They include other Federal agencies; state and local governments; universities; research institutions; the automobile, trucking, and insurance industries; and the general public" (National Automotive Sampling System, 2008).

The specific dataset used for analysis in this project is formed by extracting key attributes from the raw XML case data as explained in the approach section below.

The attributes used include: Crash Configuration, Eyewear, Race, Age, Sex, Airbag Deployment, Posture, Seatbelt status, Entrapment, Alcohol Presence, Roadway Alignment, Posted Speed Limit, Avoidance Maneuver, etc.

Attributes not used include: metadata such as CaseStr (a string case identifier) and paths to image files, EMS data such as type of care administered, detailed vehicle damage information, towing information, accident reconstruction calculated values, detailed restraint information, injuries other than fatalities, and other detailed information beyond the scope of this project.

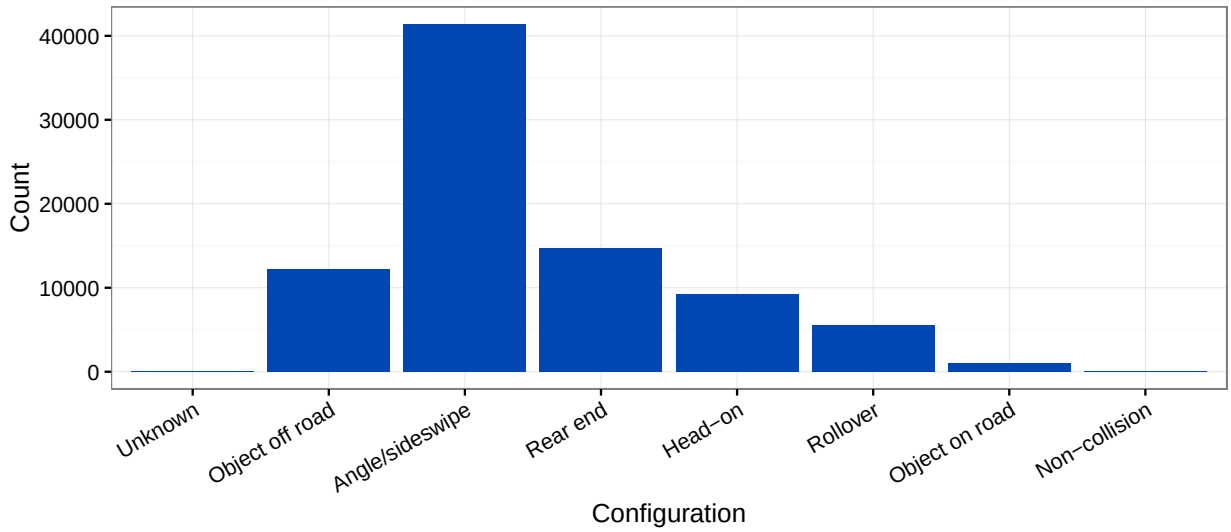

Figure 1: Counts of the various crash configurations.

Approach

The approach to be taken is shown in the graph below, and is described in the following subsections.

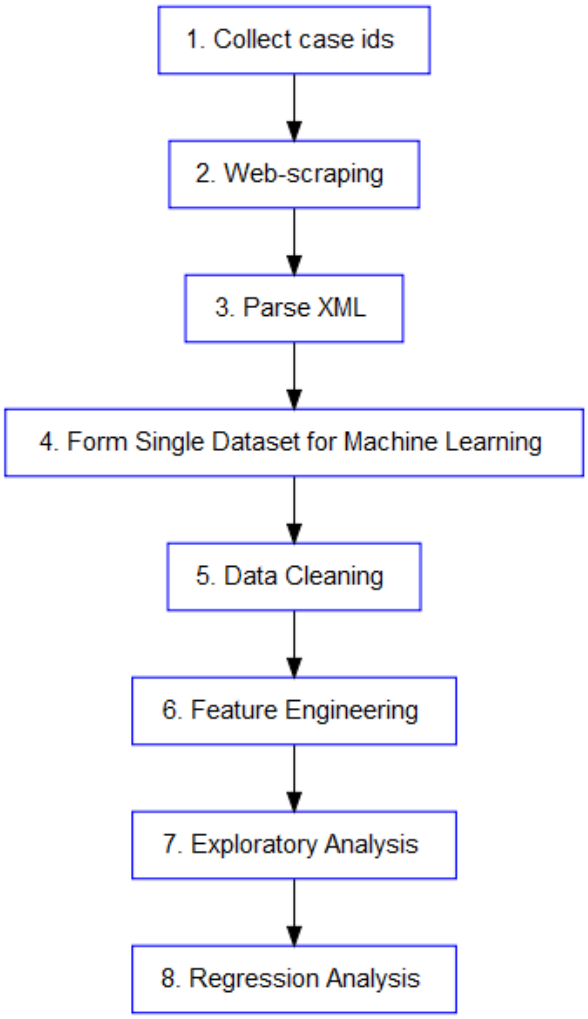

Figure 2: Box Diagram of approach taken

Step 1: Compile complete list of case ids

Case data in XML for each collision can be found using a url of the form:
http://www-nass.nhtsa.dot.gov/nass/cds/CaseForm.aspx?GetXML&caseid=112007272.

In order to obtain data for each collision, it was necessary to obtain all case ids. As there was no obvious source for the complete set of ids, and the numerical values of the ids were too sparse for brute-force web scraping, a method was devised to quickly pull all ids from a search results list containing all cases.

The complete set of results can be found using the link http://www-nass.nhtsa.dot.gov/nass/cds/ListForm.aspx and clicking "*Search*".

A Windows application True X-Mouse Gizmo was used to emulate the Linux behaviour of copying any selected text to the system clipboard. A macro in the vim text editor was then used to paste the clipboard

contents to a text file at a rate of once per second. Navigating through the result list and selecting all text allowed for quick harvesting of all 49,345 case ids in an unstructured format (nass_case_ids.txt). The result was then filtered for only unique lines containing the regular expression "`[0-9]{9}$`", again using vim. This provided the original tabular result data in a tidy, tab-delimited file, with the last field being the case id (nass_case_ids_filtered.txt).

Step 2: Scrape case data using case ids

Using R, the case data was scraped from the NASS website and stored locally as XML. Two functions were written to perform the web scrape and are located in R/scrape.R. Given a single case id, `download_case` downloads the case data and saves it as a single text file containing XML with the case id as the name. The function `download_all_cases` uses `download_case` to iteratively download all cases. If local data already exists for any case, then the case data is not re-downloaded.

Step 3: Rectangularize key XML fields

Using the xml2 package in R, many fields in the XML tree were read and stored to a data frame. Initial attempts using the XML package found that this package suffered from a severe memory leak bug. Online research revealed no known workarounds, so the code was re-written to use the newer `xml2` package.

Review of the XML structure revealed that there were many fields could be stored on a one-to-one or one-to-many basis with respect to the Case File id. The `parse_xml` function in R/parse.R iterates through the local case data files and produces a single data frame containing the key fields. Attributes with a one-to-many relationship were stored within nested data frames inside the master dataframe, and were subsequently joined prior to writing to SQLite. In total, ten SQLite tables were created: Cases, GeneralVehicle, Persons, Events, Safety, Vehicles, EMS, Occupants, Vehicle Exterior, and VehicleInterior. Each table contains a `CaseId` attribute to identify the original case file.

Step 4: Form a Single Dataset for Machine Learning

The response variable for the research question is whether or not the occupant of a collision survived. As such, it makes sense to arrange the data such that each row represents a single unique occupant from the set of all occupants in the sample. The `build_clean_dataset` function in R/cleaning.R joins the SQLite tables to form a single dataset by occupant.

This resulted in 105,296 rows containing 704 attributes. However, the mortality was not known for all occupants, so the dataset was filtered to only include occupants with a mortality of 'Fatal' or 'Not Fatal', leaving 84,277 records.

Step 5: Data Cleaning

The data collected is real world data and contained missing values. The missing values were encoded in a number of different ways, such as 'Unknown', 'N/A', 'Not Reported', or ''. These values were all coerced to `NA` in R.

Furthermore, the data has been collected over time, the XML schema has changed yearly. Coding and Analytical Manuals for the data are located here, and were referred to while reconciling the data to a more consistent schema.

Some attributes were scaled based on unit of measurement attributes. For example, ages reported in either months or years in the original data were converted to strictly years in the clean dataset.

Attributes with very few values were removed from the dataset. The response variable was converted to an integer value with 0 representing Non-Fatal and 1 representing Fatal.

Unknown values for text attributes was then converted to the string 'Unknown' in order to include unknown values in the regression, as there were considerable `NA` values in the dataset. All text values were coerced to factor variables in R.

Numeric values were scaled, and missing values were imputed using the median value.

Step 6: Feature Engineering

Several additional attributes were derived from the data, including `is_weekend`, a binary feature indicating whether the crash occurred on the weekend. The number of unique text values was reduced by combining similar fields, or by coercing very infrequent values to 'Unknown' in order to reduce the degrees of freedom.

Step 7: Exploratory Analysis

Once the data was cleaned, exploratory analysis was performed. This included searching for existing correlations in the data, and identifying attributes that would likely be useful in the regression analysis. The StepAIC function was also used to explore combinations of features that led to improved classification outcomes, and which added little value.

Step 8: Regression

Finally, a regression analysis was performed to build a model to predict survivability of collisions based on the key inputs identified in the previous steps. The outcome was validated by performing the same regression on all attributes in the cleaned dataset on Amazon Web Services Machine Learning Platform.

Results

The data from the cleaned dataset was explored using basic R functions as well as the StepAIC function in R to determine reasonable candidate features for the model. The cleaned dataset contains 83 potential predictors. However, only 20 were used to reduce computation.

Features Used

The chosen features are summarized in the table below.

Table 1: Class and count of unique values for each attribute.

Attribute	TypeOf	Unique
compartment_integrity_loss	character	9
alcohol_test	character	3
posture	character	7
age	double	121
entrapment	character	3
seatbelt_used	character	4
race	character	10
alcohol_present	character	3
avoidance_maneuver	character	15
damage_plane	character	9
crash_config	character	8

Attribute	TypeOf	Unique
fire	double	2
rollover_qtr_turns	character	7
posted_speed	double	18
eyewear	character	3
airbag_deployment	character	4
seat_orientation	character	3
roadway_alignment	character	3
preimpact_location	character	3
sex	character	3

Several of the features are scaled double floating point vectors. The remaining features are factor variables. A summary of the factor levels is shown in the table below.

Table 2: Class and count of unique values for each attribute.

Attribute	Values
compartment_integrity_loss	No Integrity loss, Side window, Unknown, Windshield, Roof glass, Rear ...
alcohol_test	No Test Performed, Test Performed, Unknown
posture	Normal Posture, Unknown, In a Child Seat, Sitting sideways or turned, ...
entrapment	No, Yes, Unknown
seatbelt_used	Yes, No, Unknown, Child Seat
race	White, Unknown, Black, Asian or Pacific Islander, American Indian, Esk ...
alcohol_present	No, Unknown, Yes
avoidance_maneuver	Steering left, No avoidance maneuver, Unknown, Braking and steering ri ...
damage_plane	Front, Left, Unknown, Back, Undercarriage, Right, Top, Not a motor veh ...
crash_config	Object off road, Angle/sideswipe, Rear end, Head-on, Rollover, Object ...
rollover_qtr_turns	No Rollover, 4, 3, 2, Unknown, 1, More
eyewear	No, Yes, Unknown
airbag_deployment	Not deployed, Unknown, Deployed, No air bag available for this crash
seat_orientation	Forward facing seat, Unknown, Other
roadway_alignment	Curve left, Curve right, Straight
preimpact_location	Other, Stayed on roadway but left original travel lane, Stayed in orig ...
sex	Male, Female, Unknown

The relationships between several key features are shown in the following plots. Mosaic plots are used to show relationships for categorical attributes, and violin plots are used for numerical attributes.

Partitioning

The data were partitioned into a test and training set using a 70/30 split.

```
set.seed(1234)
n <- nrow(df)
shuffled <- df[sample(n),]
train <- shuffled[1:round(0.7 * n),]
test <- shuffled[(round(0.7 * n) + 1):n,]
```

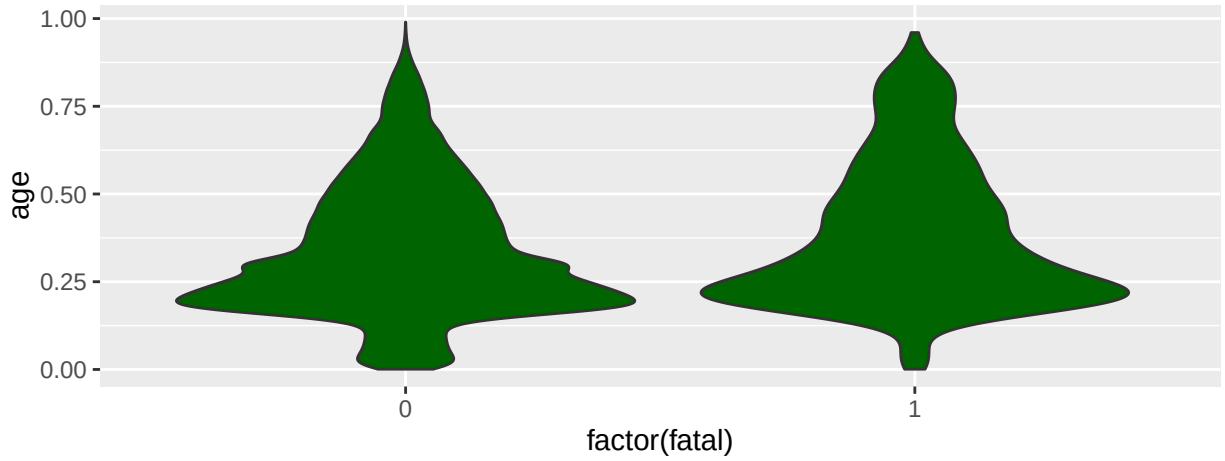

Figure 3: Age

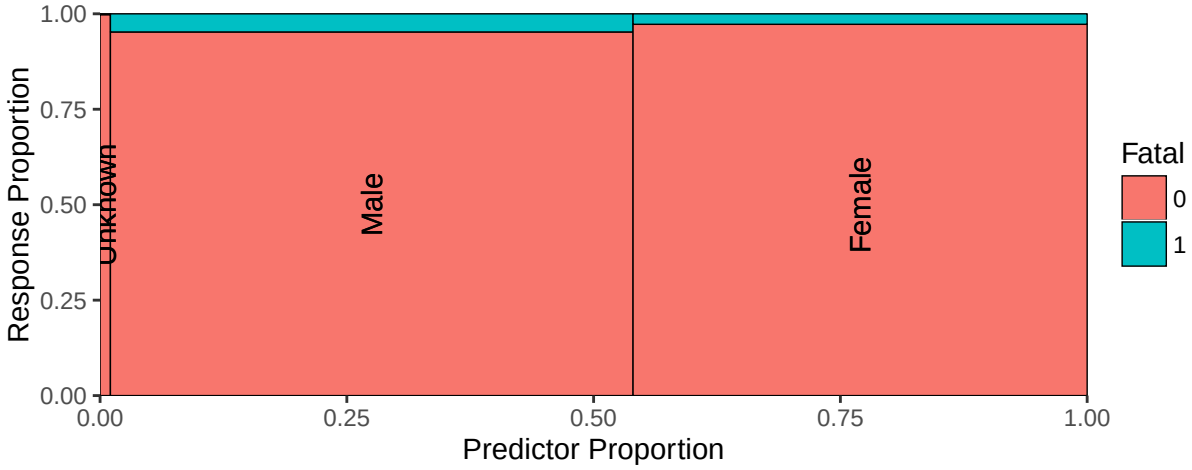

Figure 4: Sex

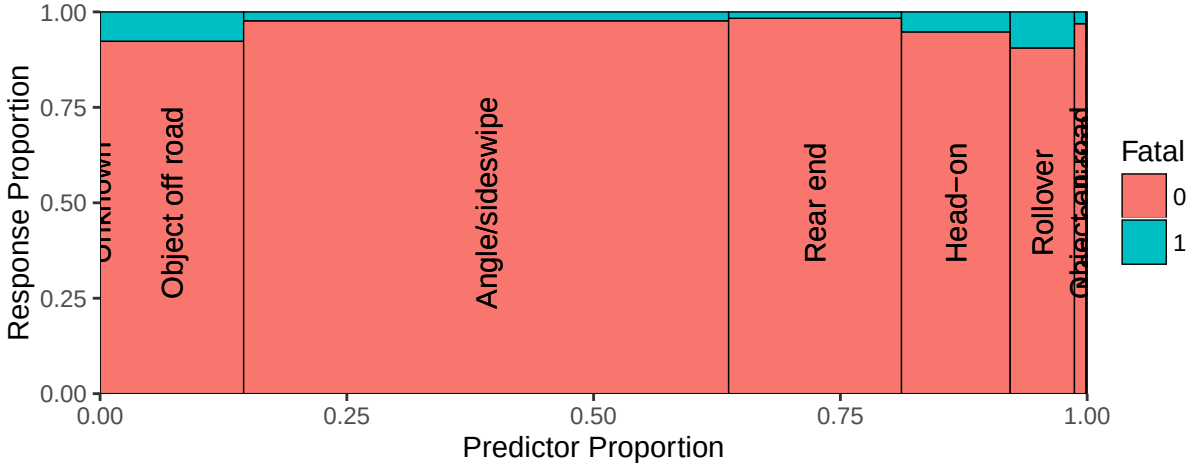

Figure 5: Crash Configuration

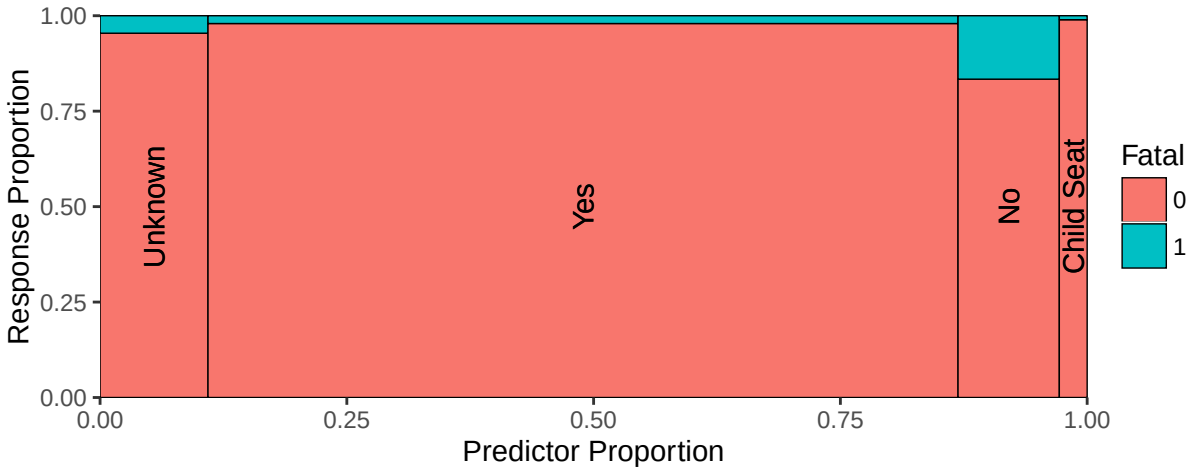

Figure 6: Seatbelt Used

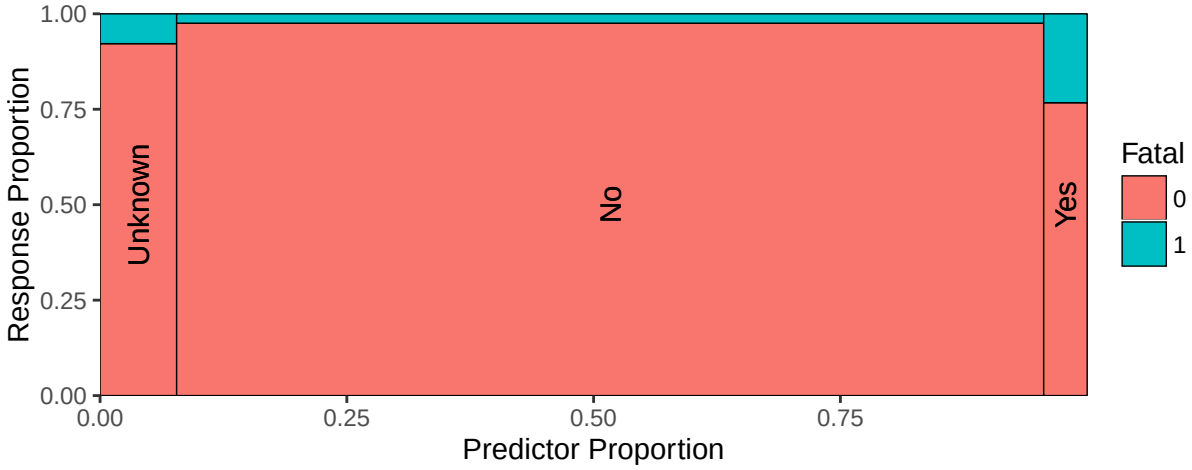

Figure 7: Entrapment

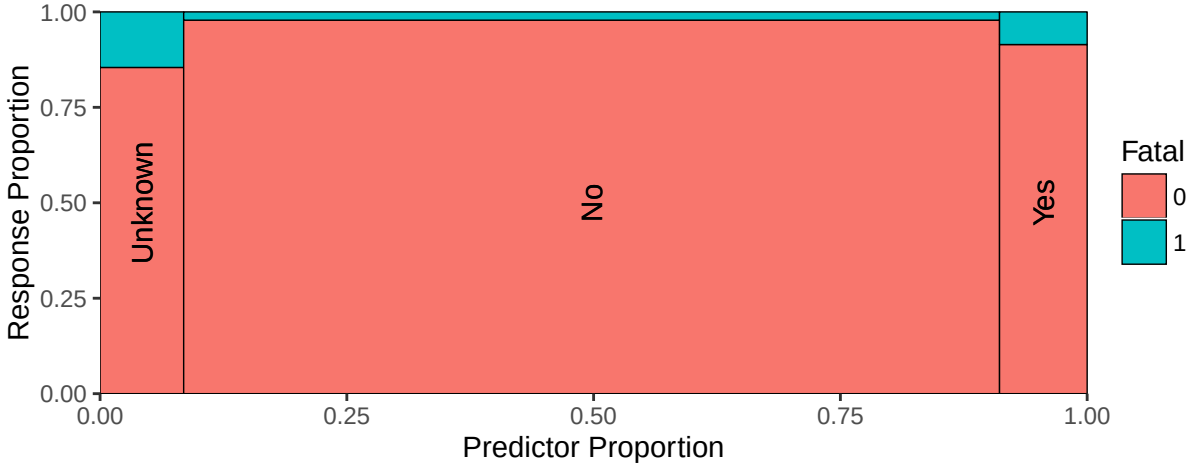

Figure 8: Alcohol Present

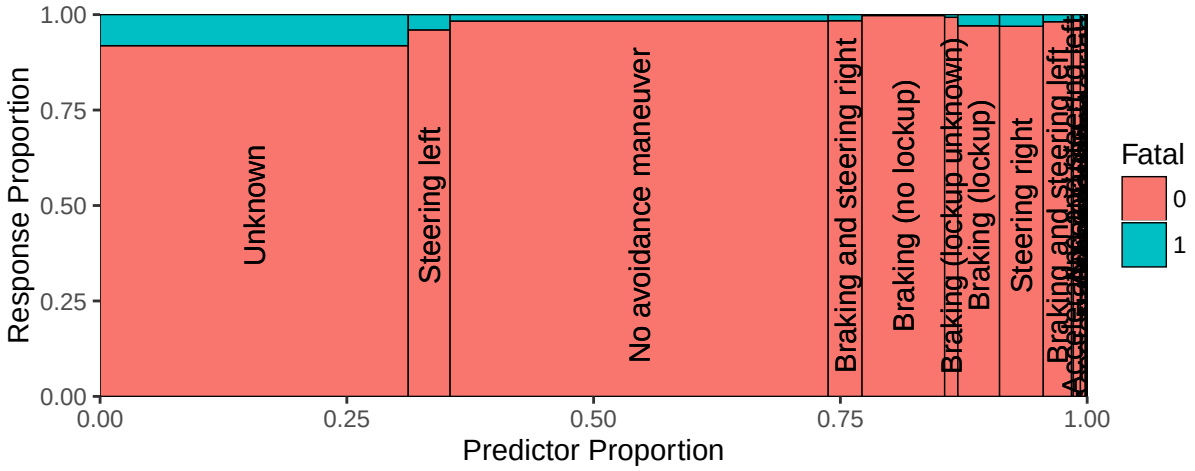

Figure 9: Avoidance Maneuver

Model Fitting

The model was fit using a binomial logistic regression with the `glm` function in R, with `family = binomial` on the training data.

```
f <- reformulate(terms, "fatal")
fit <- glm(f, family = binomial(link="logit"), data=train)
```

Performance

Probabilities for the response variable based on the test data were assigned using the `predict` function.

```
probs <- predict(fit, test, type="response")
```

This allows us to see the distribution of the predicted response variables, shown in the following historgram.

As would be expected, the model is heavily weighted towards an occupant surviving any given collision.

Confusion Matrix

Using a cut-off value of 0.5 for the classifier, a confusion matrix of the predicted outcomes can summarize the performance of the model.

Table 3: Confusion matrix for the classifier.

	0	1
0	24163	196
1	601	323

Based on the confusion matrix, this classifier appears to perform well for a dataset with this degree of class imbalance.

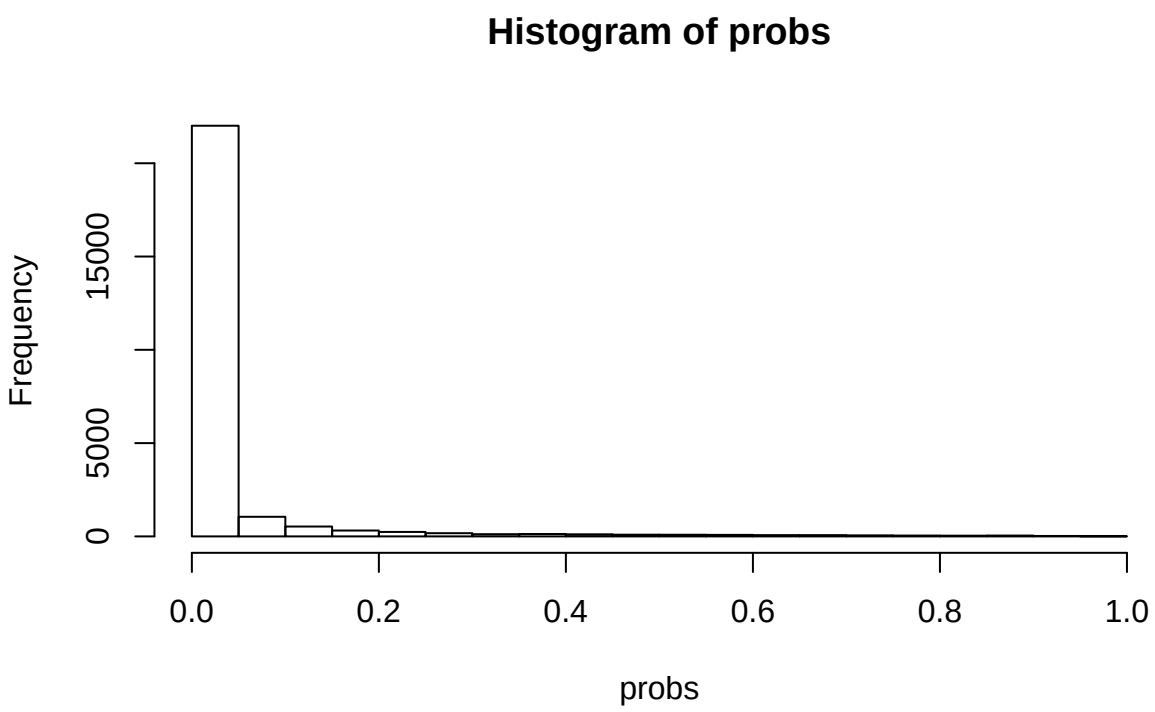

Figure 10: Histogram of predicted probabilities

Precision, Recall, and Accuracy

Precision, recall, and accuracy are all common measures used to gauge model performance.

Precision is given by `TP/(TP+FP)`. It is a measure of how likely a fatal outcome predicted by the model actually represents a fatality. For this model, this gives a value of 0.62.

Recall is given by `TP/(TP+FN)`. This is a measure of how likely an actual fatal outcome is correctly predicted by the model. For this model, this gives a value of 0.35.

Accuracy is a measure of the proportion of records correctly classified, and is given by `(TP+TN)/(TP+FP+TN+FN)` For this model, the accuracy is 0.97.

Accuracy vs. cut-off threshold

The accuracy can be plotted as a function of the chosen threshold value. The figure below shows that the accuracy is maximized near a threshold value of 0.5, and that there is little variance in the accuracy for threshold values above 0.25.

```
acc_at_thresh <- function(threshold) {
    conf <- table(test$fatal, probs > threshold)
    sum(diag(conf))/sum(conf)
}
threshold <- seq(from=0.001, to=1, length=100)
accuracy <- sapply(threshold, acc_at_thresh)
ggplot() + geom_line(aes(x=threshold, y=accuracy))
```

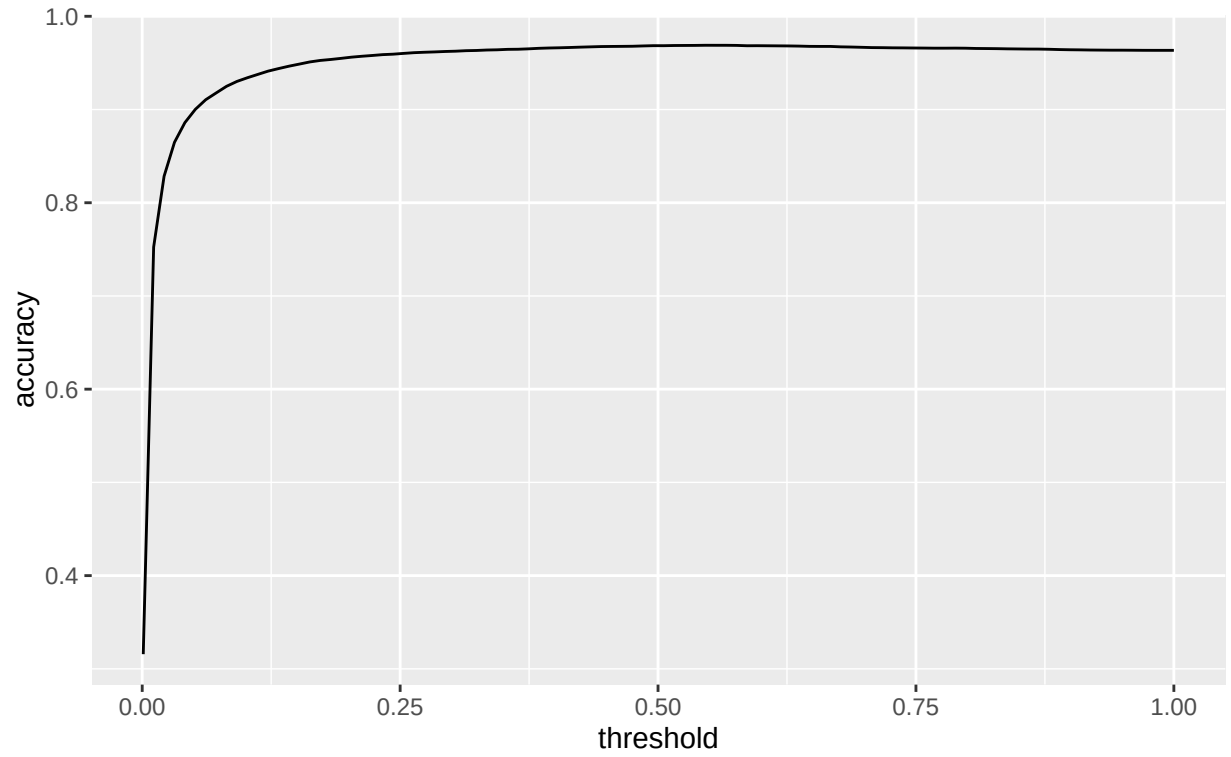

Figure 11: Effect of varying cut-off threshold on model accuracy.

ROC curve and AUC

A good way to review model performance of a binary classifier is to generate a Receiver Operating Characteristic (ROC) curve.

The ROC curve provides the performance that the model can achieve by varying the cut-off threshold used, and illustrates the trade-off of doing so. In particular, it shows that if a higher true positive rate (TPR) is demanded of the model, it will lead to a higher false positive rate (FPR).

```
pred <- prediction(probs, test$fatal)
plot(performance(pred, "tpr", "fpr"))
```

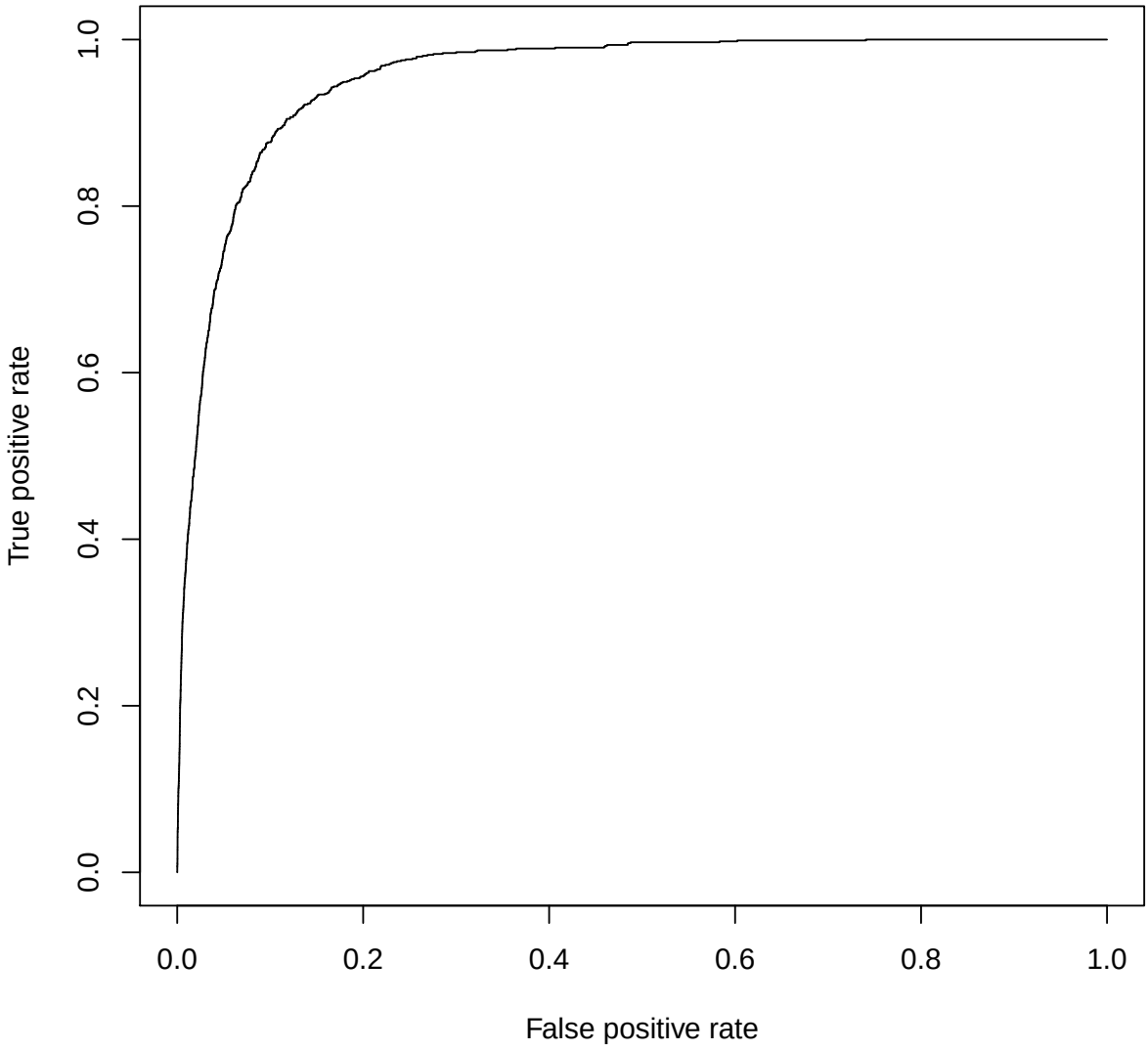

Figure 12: ROC curve for the classifier.

The area under the ROC curve (AUC), is a good general indication of the performance of a model. An AUC of 0.5 indicates that the model is no better than random chance. On the other hand, an AUC of 1.0 indicates that the model perfectly explains the response within the test set.

In this case, the AUC is 0.955. This is a good AUC for a machine learning model in general, however, the ROC curve is not particularly well suited to data with large class imbalances, since it is not sensitive to the base-rate of the predicted classes.

Recall-Precision Curve and AUC

Another method of gauging model performance is the recall-precision (RP) curve. Similar to the ROC curve, the recall-precision curve shows the effect on model performance as the cut-off threshold varies. As stated previously, the precision is the how likely a predicted positive outcome is correct, and recall is how likely the model correctly identifies a positive outcome.

```
RP.perf <- performance(pred, "prec", "rec")
plot (RP.perf, asp=1)
```

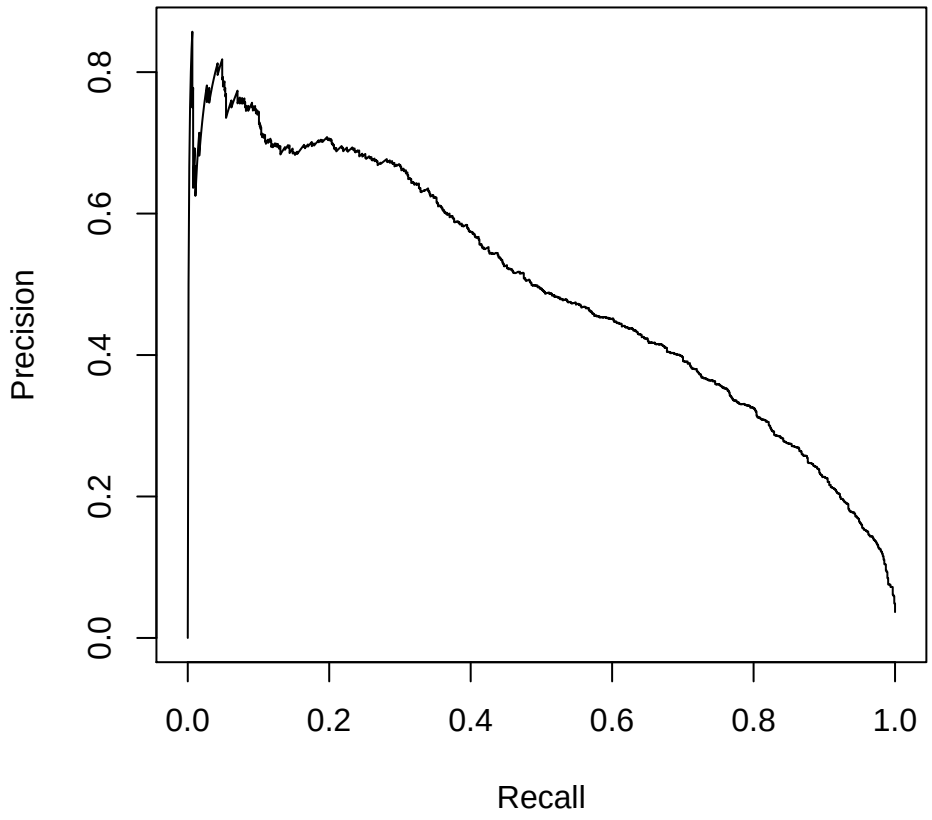

Figure 13: RP curve for the clasifier.

Here we can see the effect of the class imbalance. As the prevalence of the positive case in the data becomes increasingly rare, it becomes more difficult for a model to correctly predict the true positive cases without a significant increase in false positives. A canonical example of this is the case of cancer screening, where the base-rate can lead to false positives greatly outnumbering true positives, and where the potential harm of a false positive can be high.

The RP curve is sensitive to the response class base-rate, with the effect being that the area under the RP curve tends to decrease with greater imbalance in the response classes.

The area under the RP curve for this model is shown below.

```
RP.perf@y.values[[1]][is.nan(RP.perf@y.values[[1]])] <- 1 # Remove single NaN
caTools::trapz(x=RP.perf@x.values[[1]], y=RP.perf@y.values[[1]])
```

```
## [1] 0.4960697
```

While the area under the ROC curve and the area under the RP curve cannot be directly compared, review of the RP curve and the area under it suggests that the performance of the model is overstated by the area under the ROC curve.

Performance by Crash Configuration

The following figures show the performance of the generalized classifier for various crash configurations.

The classifier appears to do a reasonable job of classifying the survivability of all crash configurations, although classification in crashes involving rollover performs noticeably poorer than the other configurations.

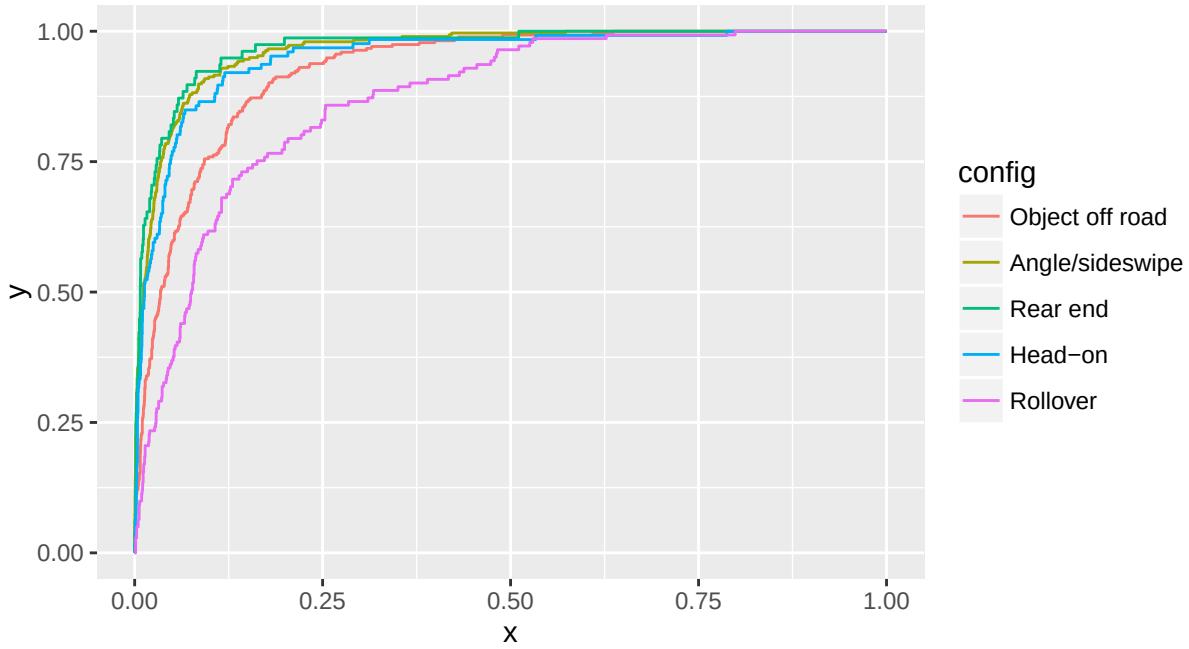

Figure 14: ROC curves for various crash configurations within the test set.

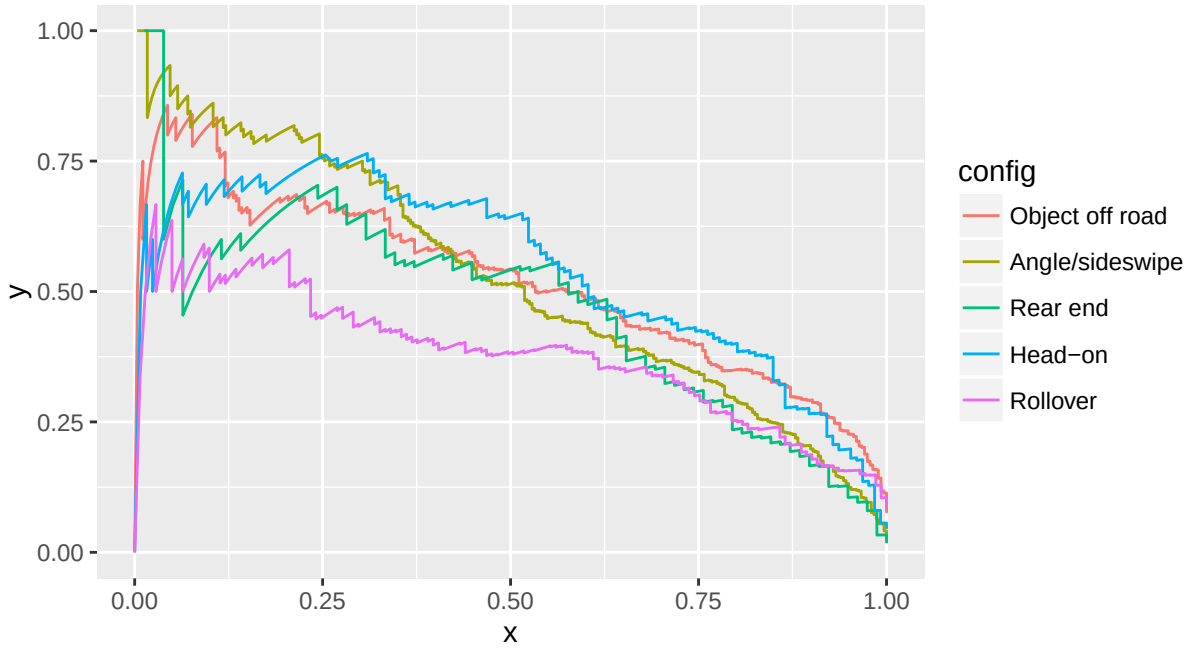

Figure 15: RP curves for various crash configurations within the test set.

Amazon Web Services

To validate the findings, Amazon Web Service's cloud based Machine Learning was used to build an additional binary classification model with the entire cleaned dataset, again partitioning a training and test set using a 70/30 split.

The resulting model had an AUC of 0.95, similar to the model developed in R. The confusion matrix for the model is shown in the table below.

Table 4: Resulting confusion matrix form AWS Machine Learning.

	0	1
0	23866	141
1	677	306

The precision was reported as 0.6846, the recall as 0.3133, and the accuracy as 0.9673. These values are all similar to the values of the model developed in R using only the 20 selected attributes.

Conclusions

Overall, the binary classifier performed well, with an accuracy of 97%. This is largely due to prevalence of a single class (Non-fatal), however, the model outperforms the naive approach of classifying every response as Non-fatal, which would have an accuracy of only 96.3%. The model predicts 35% of fatal outcomes correctly, and 62% of cases classified as fatal are indeed fatal.

The model had a very high area under the ROC curve of 0.955. Typically, this is considered very good performance for a binary classifier. However, the class imbalance in the response variable manifested itself in a lower area under the RP curve than would be ideal, of 0.496.

There was some agreement on the features indentified in the literature review. In particular, the presence of alcohol played an important role in the classifier. Age, sex, seatbelt use, and airbag deployement were also important features. However, some features identified in the literature review such as the number of people in the vehicle did not appear to play a major role.

Replicating the analysis in AWS for all features of the cleaned dataset showed additional features beyond the 20 chosen did not add substantial value, however, additional feature engineering might result in improved model performance.