Machine Learning & Statistical Data Mining
Assignment 2, 2017-18

Predicting Fatalities in US Police shootings data 2010-2016

Introduction

This project analyses data collected by VICE that looks at both fatal and non-fatal shootings from the 50 largest local police departments found in the United States from 2010 to 2016. I was interested in how this data was collected and to use some of their findings to show different results and similar reports. I chose this data after reading the VICE report on the subject. I had already found another dataset that collected all fatal shootings by the police but did not record non- fatal shootings. Looking at the VICE data, I was interested to see if the 'Fatal' field could be predicted by using the identity factors of the officer, subject or the area in which the incident occurred.

I have used machine learning classification techniques to predict if an incident was fatal using the remaining fields as predictors. The models I have chosen to use are Random Forest, Logistic Regression, Neural Networks and KNN. In carrying out this project it was important for me to understand the limitations of the dataset as well as the potential for bias with multiple sources feeding into the dataset. In their report, VICE describe the data collection methods as "directly from law enforcement agencies and district attorneys, though we also sometimes relied on local media reports". They also discuss how some departments would give limited results, leading to blank fields. I wanted test if these 'unknown' and 'NA' fields would be predictive, and so, as far as possible I tried to leave these in.

Data Cleaning and pre-processing

The data processing and data cleaning stage ended up being a much larger task than I anticipated. I downloaded the data straight from the VICE report page, which gives you the option to download the full dataset or the data from each police department. On their own GitHub page, they describe the data cleaning techniques they undergo. The data I have downloaded has already gone through their 'initial data pre-processing' However, to achieve the analysis I intended, I had to go through further pre-processing.

The following techniques were applied to the full dataset (before splitting to training and test sets). I started by removing errors, blank spaces and bad characters from each feature. I then updated any empty or missing values where the information was contained in the 'notes' or 'full description' fields. Looking at the 'Officer Race' and 'Officer Gender' fields, in cases where there were more than one officer present, these fields would contain multiple characters. This would be hard to read for the machine learning models, so I split them out.

I also recoded some of the 'Subject Race' and 'Officer Race' characters that were outside the range described by VICE in their report. Many of these decisions had to be a judgement call, rather than going by text book answer. For example, the 'Subject Age' field contained many different types of answer, from the age as a numeric answer, to being in a range such as '20-29.' Though I am aware it is best practice to have the numbers, I chose to group these all into 'Subject Age Range' categories. This allowed me to use more data in that field (rather than recoding many as 'unknown.' Finally, I removed some features that I would not need for my analysis. I chose to do all these processes before splitting the data as they are more 'data cleaning' techniques rather than pre-processing and do not use the outcome variable.

The first pre-processing technique I applied was to create 'dummy variables' from the categorical features with more than two categories. This is due to some of the models I planned to use not allowing for grouped categories. I kept both grouped and independent methods in my dataset as Random Forest allows for both, so I wanted to test which performed better.

Secondly, I looked at the numeric features with missing values and techniques to impute these. I chose to leave the categorical missing values as these can be predictive. I chose to drop the 'Number of Shots' numeric category as it had very high percentage of missing values. For the remaining numeric value, 'Number of Officers' I used the MICE package to impute the missing values with the mean of the feature. I made sure to apply this to the training set first, and then apply to the test set, using the mean of the training set. This feature also displayed significant skewness, so I used the BoxCoxTrans method from the CARET package to resolve this. Again, this technique was applied to the training and test sets, using the result from the training set.

Many of the categorical features had a near zero variance. I created a duplicate of the dataset where I removed these. This was to be used on later models that cannot handle near zero variance as well as Random Forest. Once this was applied, I looked at the remaining dummy variables and removed one from any full set of categories that remained. This is to avoid the 'dummy variable trap.' Some pre-processing techniques such as looking at the correlation of the features could not be used due to all but one of my features being categorical.

Overview of data with exploratory analysis

The training set data contains 2383 rows and 121 variables (including the 'Fatal' outcome value). Many of the categorical features have been coded with a letter to describe the value (ie B for Black under race value). The full code descriptions can be found on the VICE GitHub page, under 'Variables.'

After pre-processing, there is only one numeric predictor column, the Number of Officers. I used bar chart to look at the cases of 'Fatal' and 'Non-Fatal' incidents, compared to the number of Officer. The chart shows the mean number of officers for each outcome. It's important to note that the 'Number of Officers' field has already been transformed here and so the chart serves to depict the increase or decrease per outcome only.

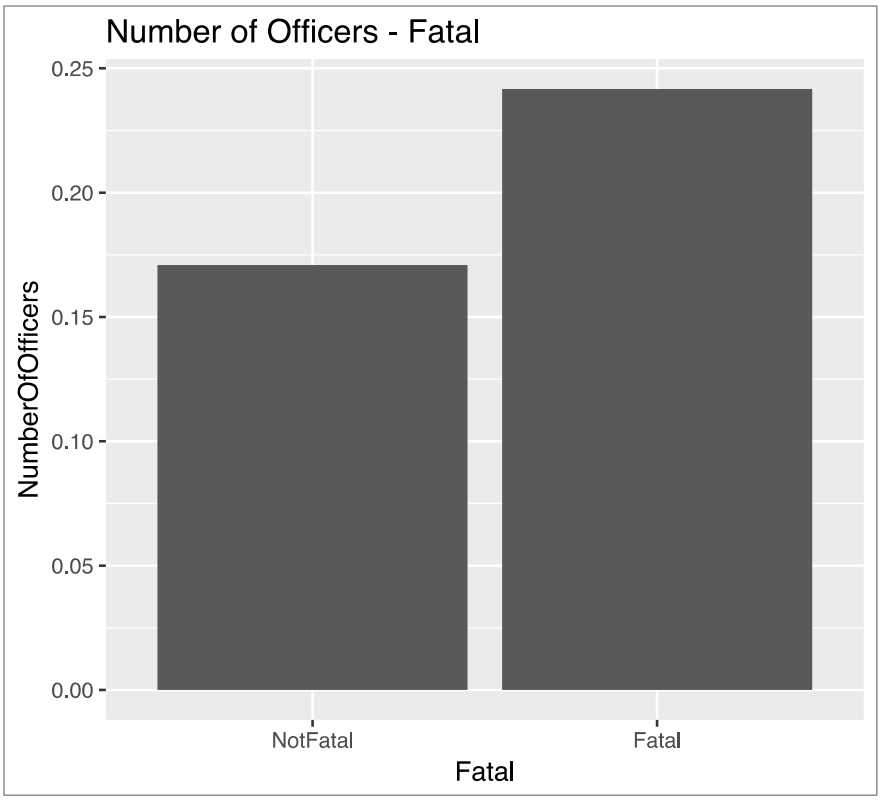

Figure 1 Bar chart - Number of Officer to Fatal

I then continued to examine some of the features, looking at the frequency and 'Fatal' rate of 'Subject Age Range,' 'Subject Gender,' and 'Subject Armed.' The dark grey depicts non-fatal incidents and the light grey fatal incidents. You will notice that some categories are very large and some very small, some of these smaller features are removed later as a result of applying 'near zero variance' to the features.

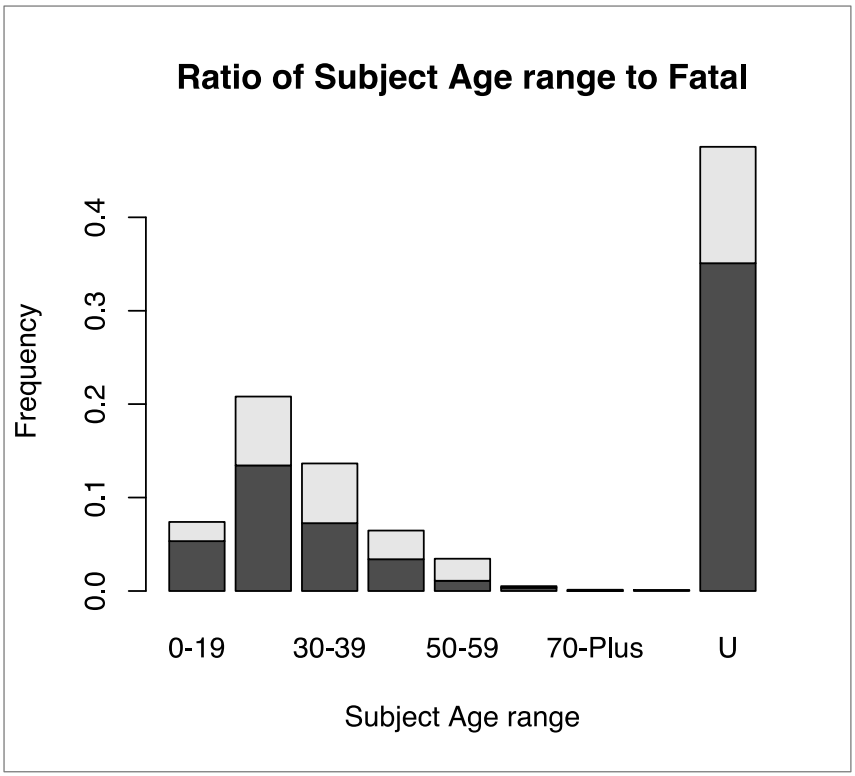

Figure 2 Bar Chart - Subject Age Range - Fatal

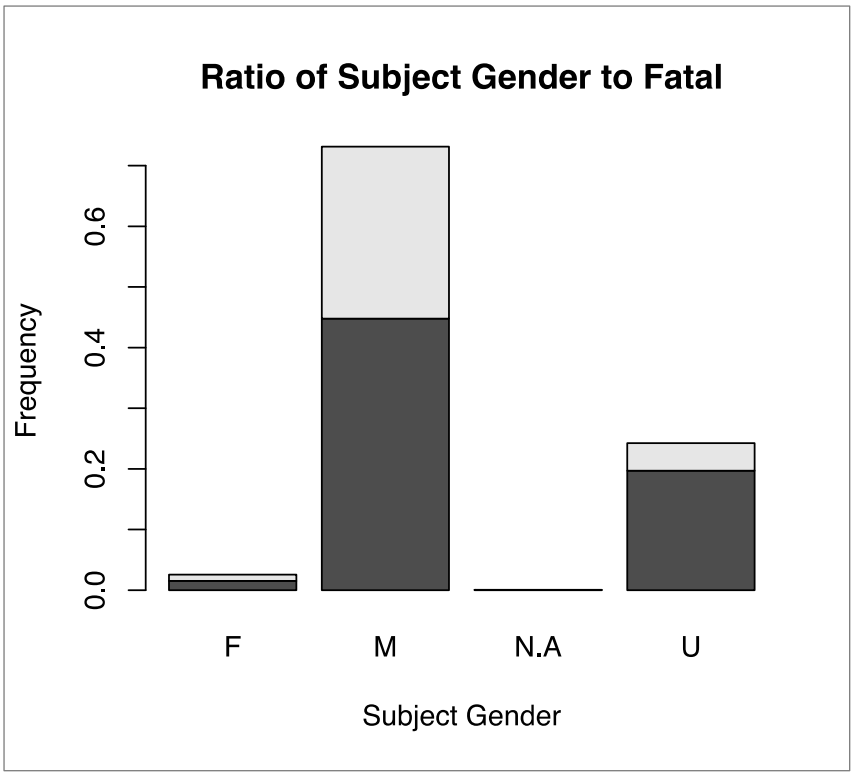

Figure 3 Bar Chart - Subject Gender - Fatal

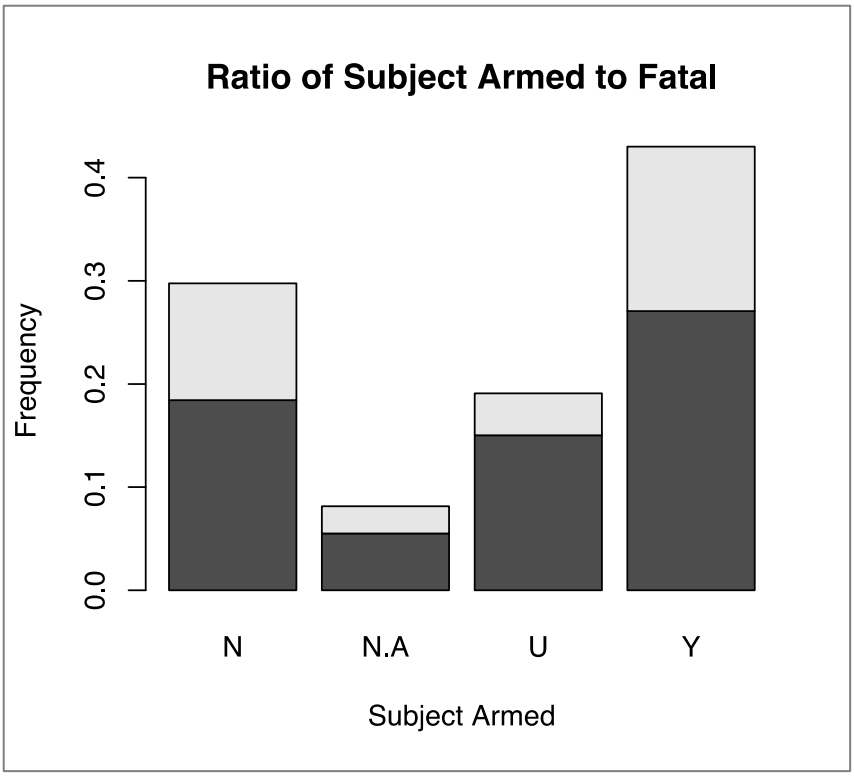

Figure 4 Bar Chart - Subject Armed - Fatal

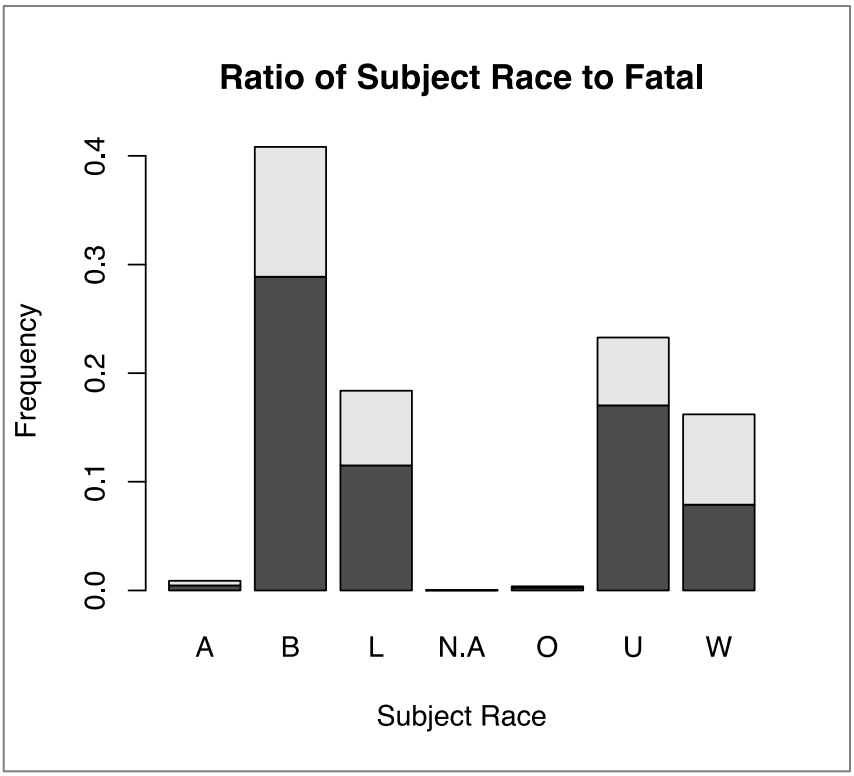

Figure 5 Bar Chart - Subject Race - Fatal

Looking at the outcome 'Fatal' value, I can see there is an imbalance in the class distribution. 66% of the cases are classed as 'Not-Fatal,' leaving only 34% classed as 'Fatal.' This imbalance is less severe as some cases, but it is still enough to potentially skew the results.

Data Splitting and reporting stats

As mentioned above, there are a few different tests I am looking to apply meaning the data needs to be split a few different ways. Firstly, to split the data into train, test and validation sets, I used the CARET models function 'createDataPartition.' This splits the data based on the outcome variable, allowing for an even distribution across all sets. I chose to split the data 80/20 training to test and training to validation respectively.

Secondly, for the models using Logistic Regression, Neural Networks and KNN, I created a new split excluding values with near zero variance. Thirdly, to tackle the potential class imbalance issues described above, I used the original training set to create new samples using three techniques; Upsample, Downsample and SMOTE.

Lastly, I divided the data into a set with only the original grouped (factor) categorical features and one using just the dummy variable features.

After creating the data splits, I used a 'ctrl' variable to set the training control for all model tuning. Using this variable, I chose to apply 10-fold cross validation to all models and to hold five stats to report on; ROC, sensitivity, specificity, accuracy and Kappa.

Random Forest: Grouped vs Independent categorical variables

I started by using Random Forest to test if I get better results using the grouped, factor variables or the separate dummy variables. For all random forest models in this project, I used the control method defined above, set the number of trees to 1,500 and set the tuning parameter to five.

Both models chose mtry as 2 for from the tuning results. As you can see from the ROC plots below, the predicted results for the validation and test sets show not much of a difference between the grouped and independent tests. In fact, the actual predictions are the same for both tests, the AUC shows a very small increase for the independent model on the validation set and a very small increase for the grouped model on the test set. I decided to use the independent set for the next set of tests using random forest. The difference in results was very small and so I based my decision on some variables have a very high number of factors (cities has 42) in which dummy variables are advised. It was interesting to see that both models leaned heavily on the 'Non-Fatal' predictions, further evidence of class imbalance. This is also shown as all results have a high sensitivity but a low specificity.

Model results – Training set

Data	mtry	ROC	Sens	Spec	Accuracy	Kappa
Grouped	2	0.69863	1	0.00246914	0.6609296	0.00324825
Independent	2	0.6977299	0.9987342	0.00493827	0.6609332	0.00481994

Model results – Validation set

Data	AUC	Sens	Spec	Accuracy	Kappa
Grouped	0.744	1	0.01316	0.6849	0.00178
Independent	0.7599	1	0.01316	0.6849	0.0178

Model results – Test set

Data	AUC	Sens	Spec	Accuracy	Kappa
Grouped	0.7271	0.998	0	0.6586	-0.0027
Independent	0.7246	0.998	0	0.6586	-0.0027

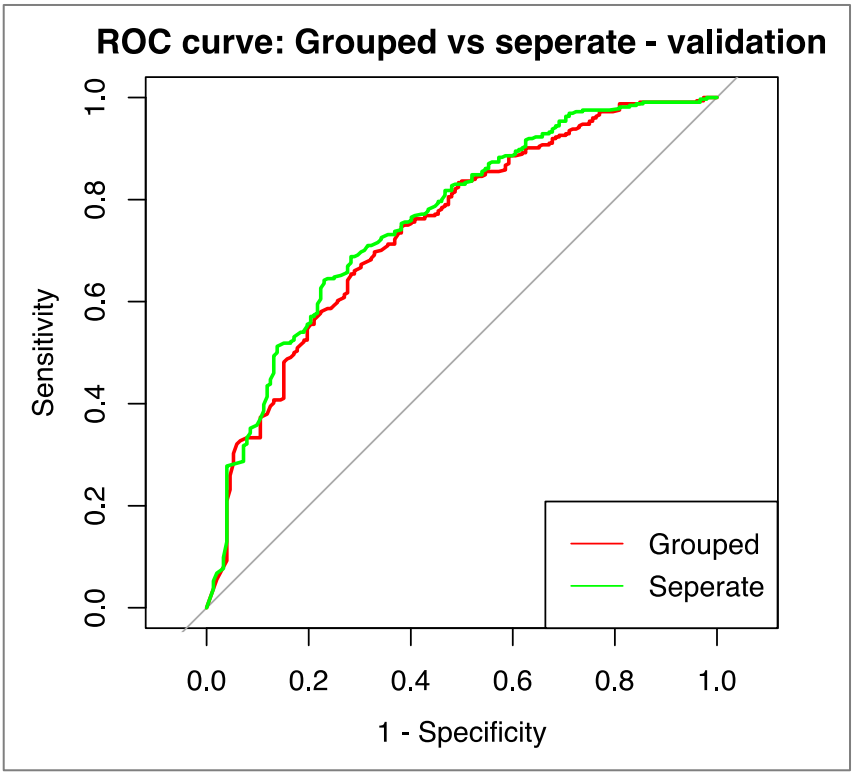

Figure 6 Random Forest ROC curve: Grouped vs Separate sets - Validation

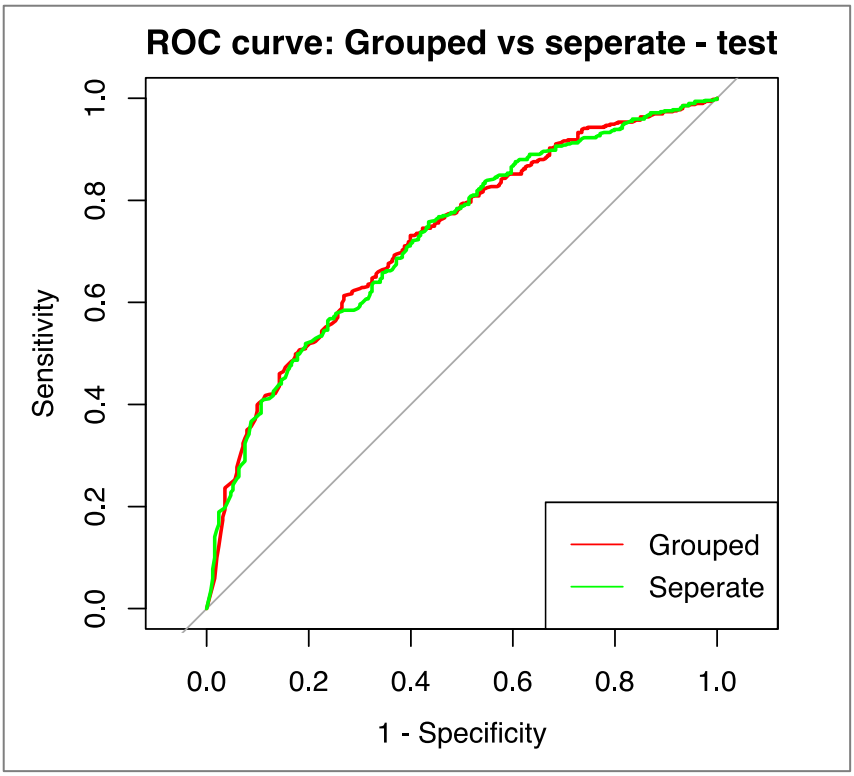

Figure 7 Random Forest ROC Curve - Grouped vs Separate - Test

Random Forest: Normal vs Resampled

My next test using Random Forest I built new models, using the three-resampling data I created with Upsampling, Downsampling and SMOTE techniques. From the tuning methods, the Downsampled model chose mtry as 2, the Upsampled model chose 56 and the SMOTE model chose 29. All models use only the independent features set (as defined in the test above), the 'Normal' model below depicts the original training set with only these independent features. The results from the training set show the SMOTE model has the high ROC, with the Upsampled model second. The Downsample model results shows a decrease in ROC from the normal result, so worse performance. However, when applying the models to the validation and test sets, the Upsample model wins out with the highest AUC on both sets. As you can see from the plot below, the validation set shows the biggest difference applying the new sampling techniques, although the test results still show a small increase in AUC for all re-sampling models. It is also interesting to see how the sensitivity and specificity change in these re-sampling methods. Looking at the training results, the upsampled model has a much higher specificity than sensitivity, whereas the downsampled model shows much more similar results.

Model results – Training set

Data	mtry	ROC	Sens	Spec	Accuracy	Kappa
Normal	2	0.6977299	0.9987342	0.00493827	0.6609332	0.00481994
Upsampled	56	0.8562336	0.698706	0.8232323	0.7609781	0.5219487
Downsampled	2	0.690535	0.6753086	0.6024691	0.6388889	0.2777778
SMOTE	29	0.8980834	0.8833333	0.75	0.8166667	0.6333333

Model results – Validation set

Data	AUC	Sens	Spec	Accuracy	Kappa
Normal	0.7599	1	0.01316	0.6849	0.0178
Upsampled	0.9801	0.9568	0.9342	0.9496	0.8848
Downsampled	0.7279	0.6852	0.6382	0.6702	0.2988
SMOTE	0.9269	0.8488	0.8816	0.8592	0.6928

Model results – Test set

Data	AUC	Sens	Spec	Accuracy	Kappa
Normal	0.7246	0.998	0	0.6586	-0.0027
Upsampled	0.7371	0.7128	0.6285	0.6841	0.3266
Downsampled	0.731	0.6719	0.6945	0.6868	0.3443
SMOTE	0.7356	0.7251	0.5968	0.6815	0.312

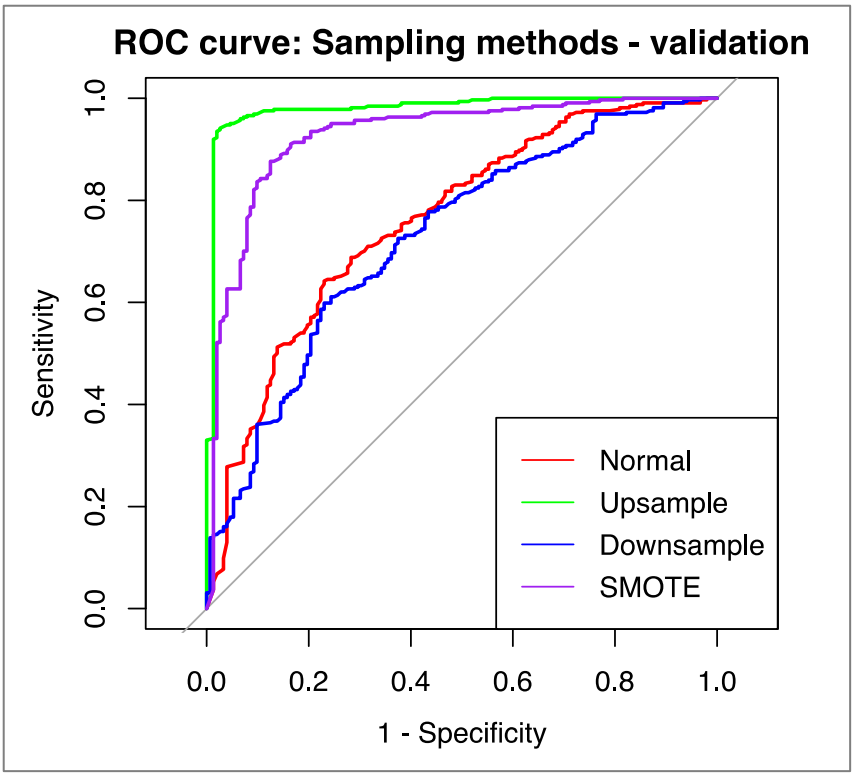

Figure 8 Random Forest ROC Curve - Sampling Methods - Validation

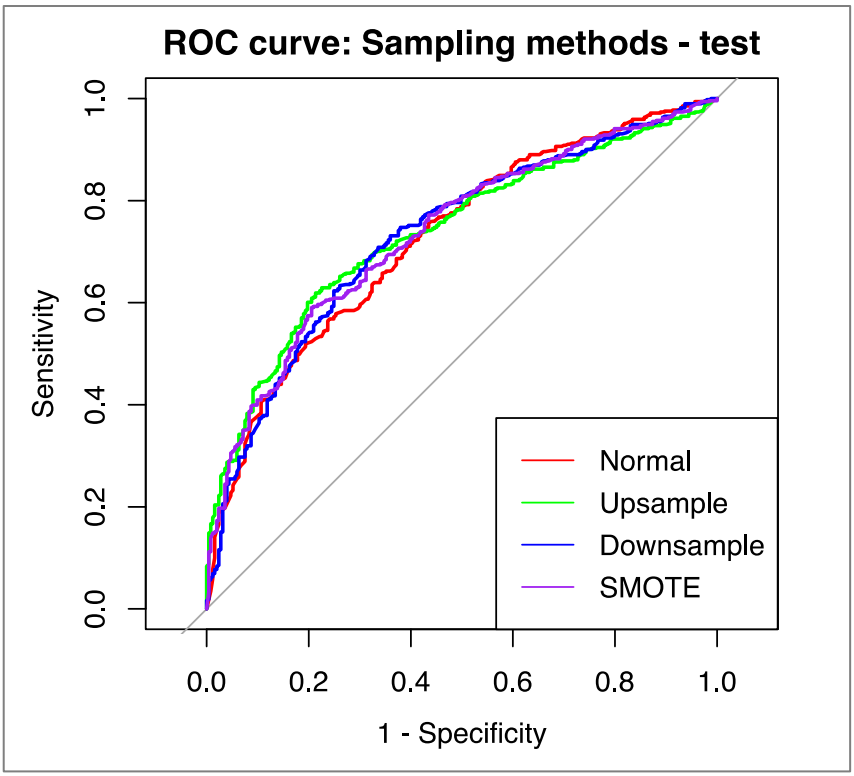

Figure 9 Random Forest ROC Curve - Sampling Methods - Test

Random Forest: Important features

The final Random Forest test I applied was to look at the 'important features' found from the best performing model in the previous tests. For this I used the 'VarImpPlot' function from the Random Forest package. This gave me a diagram (shown below) of the most important features used when building the model. These variables have a large mean decrease in accuracy and are the most important for the classification of the data. It is interesting to see the 'Number of Officers' as having a much higher importance rate than the other variables. This could be due to the fact that it is the only numeric feature. However, it could also suggest this is potentially a strong predictor.

Using these important features with the Upsampled data, I fitted a final Random Forest model. In the tables and plots below, I compare this model to the original Upsampled model fitted above (described as 'Normal' below). Looking at the results the new model shows a decrease in performance with a lower ROC on the training model and a lower AUC on both the validation and test set.

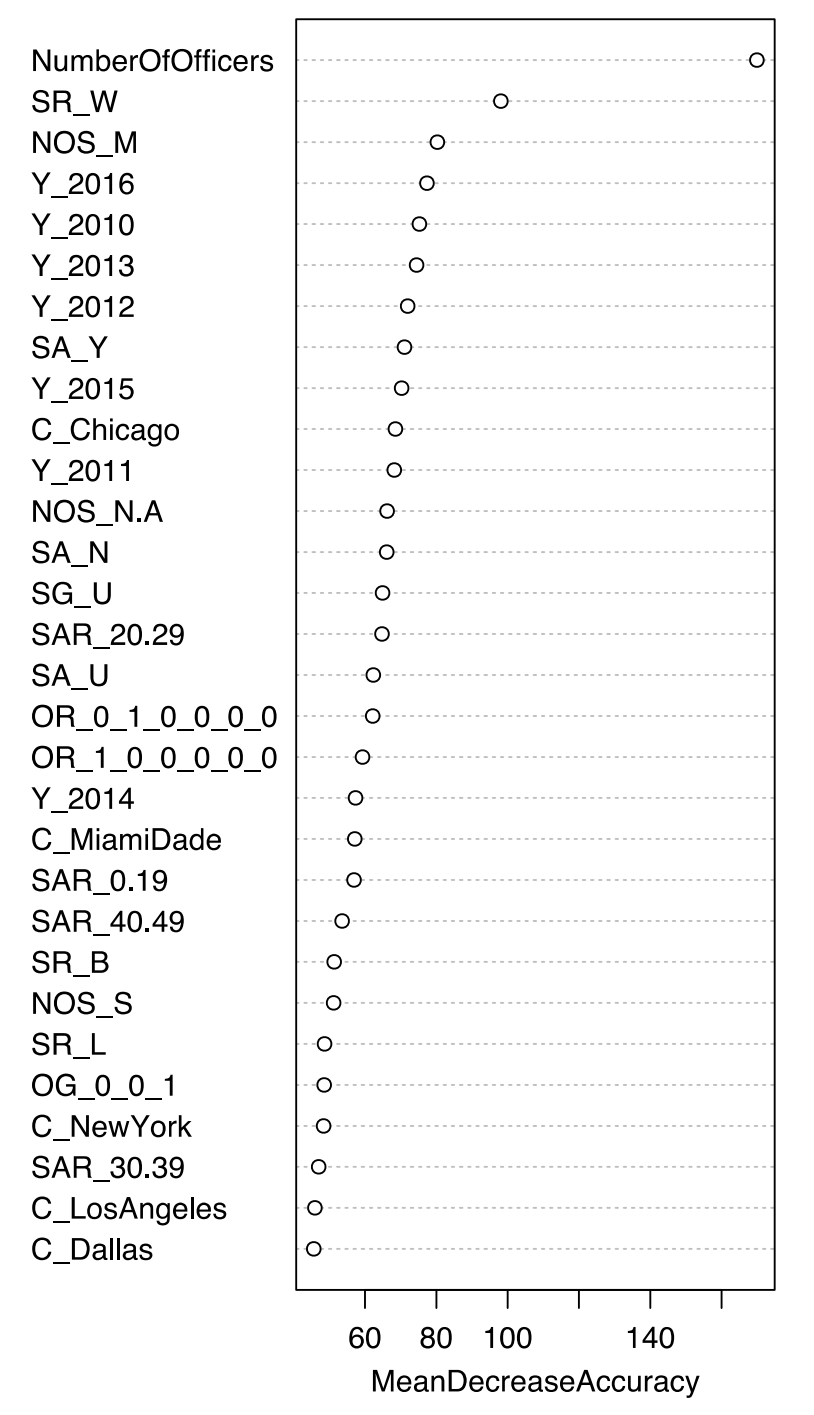

Figure 10 Random Forest - Important Features

Model results – Training set

Data	mtry	ROC	Sens	Spec	Accuracy	Kappa
Normal	56	0.8562336	0.698706	0.8232323	0.7609781	0.5219487
Important	16	0.8332837	0.6897444	0.7901274	0.7399454	0.4798808

Model results – Validation set

Data	AUC	Sens	Spec	Accuracy	Kappa
Normal	0.9801	0.9568	0.9342	0.9496	0.8848
Important	0.9532	0.9321	0.9079	0.9244	0.8284

Model results – Test set

Data	AUC	Sens	Spec	Accuracy	Kappa
Normal	0.7371	0.7128	0.6285	0.6841	0.3266
Important	0.7193	0.721	0.5929	0.6774	0.304

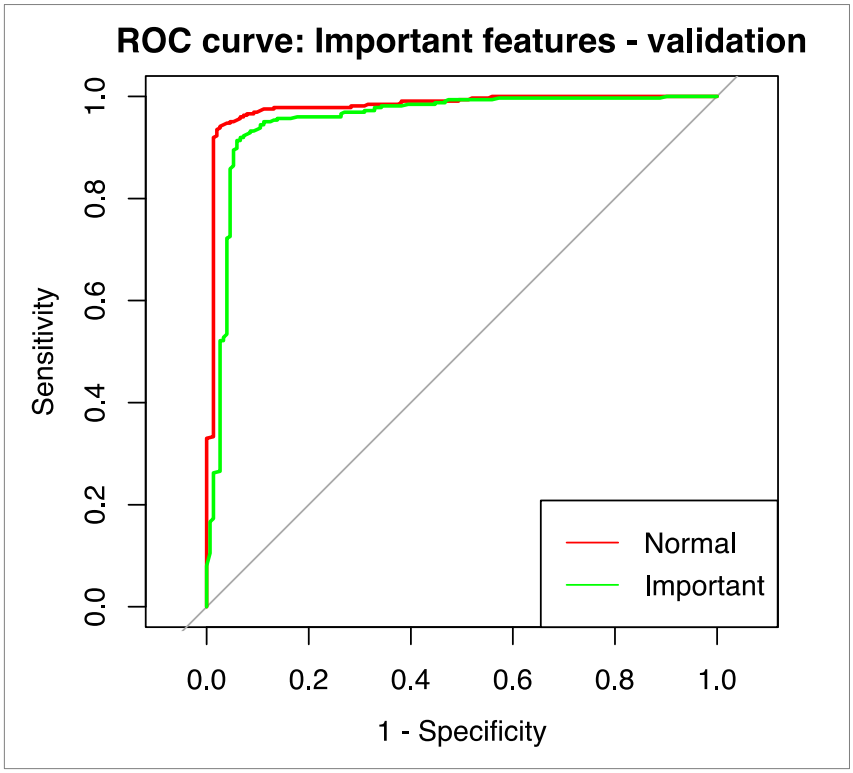

Figure 11 Random Forest ROC Curve - Normal vs Important Features - Validation

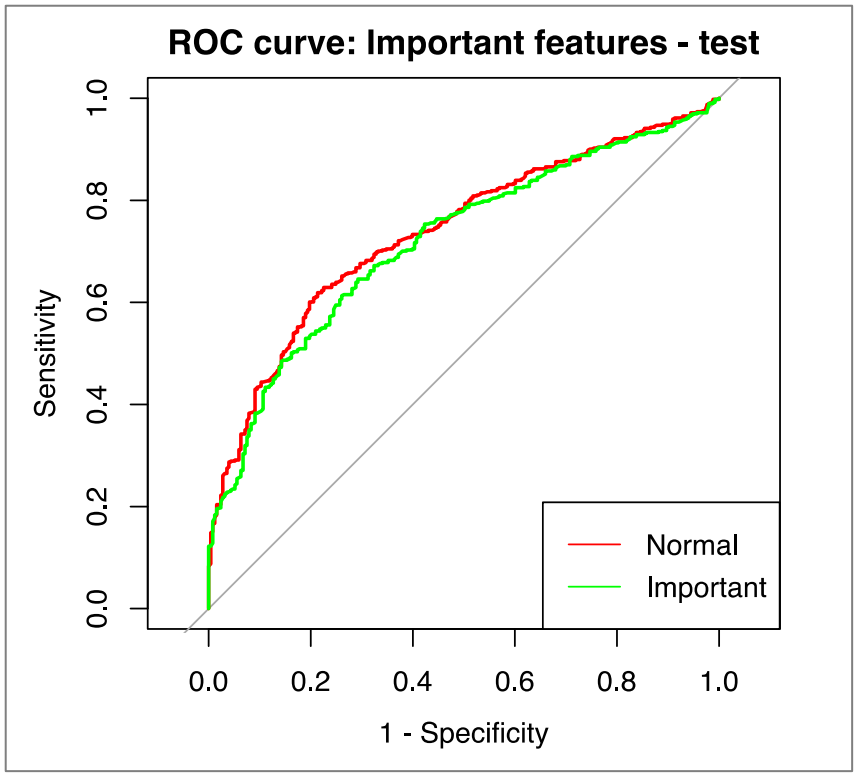

Figure 12 Random Forest ROC Curve - Normal vs Important Features - Test

Logistic Regression: Normal vs Resampled

Using logistic regression, I fitted four models using the original training set (described as normal below) and the Upsampled, Downsampled and SMOTE sets. As mentioned at the pre-processing stage, I removed the near zero variance features on all sets as logistic regression cannot handle these well. As with the Random Forest test, the results show the Upsampled data achieves the best performance in ROC and AUC across the training, validation and test sets. However, the difference is very small, as you can see from the ROC plots. The only large difference is in the decrease for the SMOTE model when predicting on the test set.

Model results – Training set

Data	ROC	Sens	Spec	Accuracy	Kappa
Normal	0.6986973	0.8792147	0.3444444	0.6974315	0.2487541
Upsampled	0.7101788	0.6693784	0.6522575	0.6608123	0.3216316
Downsampled	0.6982777	0.6716049	0.6345679	0.6530864	0.3061728
SMOTE	0.6818339	0.6660494	0.6037037	0.6348765	0.2697531

Model results – Validation set

Data	AUC	Sens	Spec	Accuracy	Kappa
Normal	0.6916	0.8889	0.3092	0.7038	0.2252
Upsampled	0.6956	0.6698	0.6053	0.6492	0.2542
Downsampled	0.6933	0.6728	0.6184	0.6555	0.287
SMOTE	0.6474	0.6728	0.5658	0.6387	0.223

Model results – Test set

Data	AUC	Sens	Spec	Accuracy	Kappa
Normal	0.7578	0.8798	0.3478	0.6989	0.2534
Upsampled	0.7596	0.7271	0.6917	0.7151	0.3975
Downsampled	0.7527	0.7026	0.7154	0.707	0.3904
SMOTE	0.6748	0.6782	0.5652	0.6398	0.2327

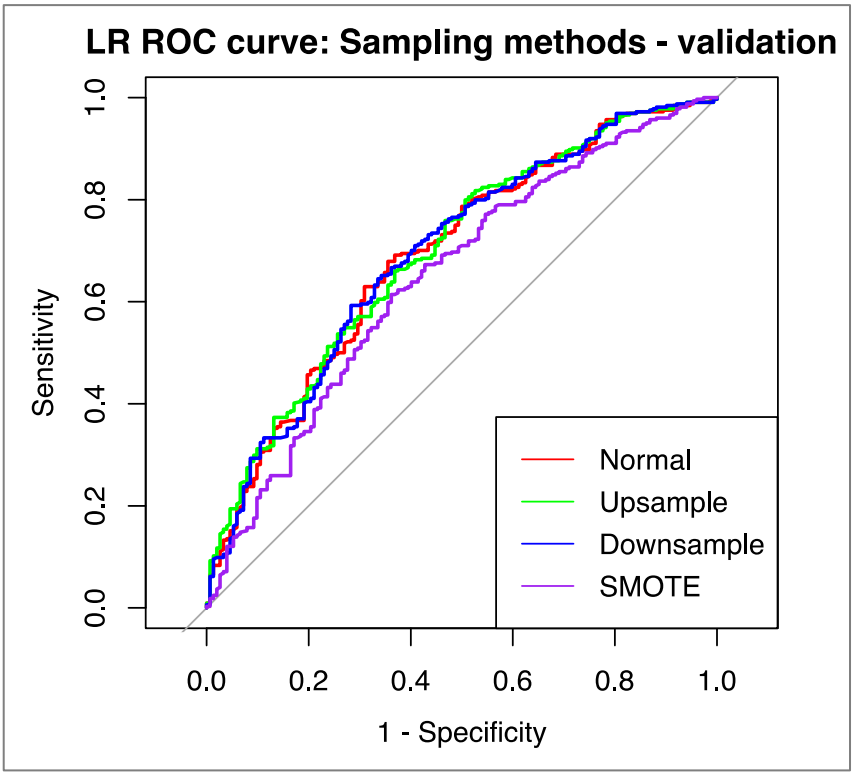

Figure 13 Logistic Regression ROC Curve - Resampling - Validation

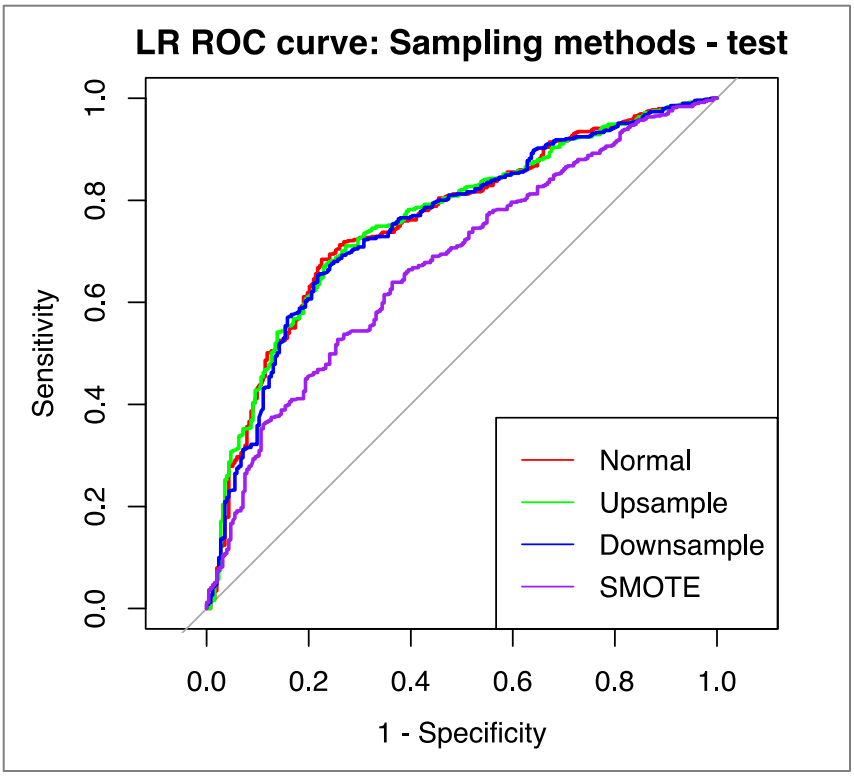

Figure 14 Logistic Regression ROC Curve - Resampling - Test

Logistic Regression: Significant features

In the final test using logistic regression I looked at the features with the best p-values from the Upsampled model described above. The table below depicts these values. It is interesting to see the Number of Officers again featuring with the best result. Its positive coefficient suggests an increase in the number of officers, increases the chance of the subject being fatal. Other examples of features with good p-values are the dummy variables for city; New York and Philadelphia. There negative coefficients suggest the incident taking place in those areas decreases the chance of the subject being fatal.

Coefficients	Estimate	P value	
NumberOfOfficers	1.47809	9.42E-14	***
SA_N	0.21782	0.01942	*
SA_U	-0.81582	9.18E-08	***
SG_U	-0.7828	0.004632	**
SAR_0.19	-0.95002	1.34E-04	***
SAR_20.29	-0.88549	4.26E-05	***
SAR_30.39	-0.67447	0.002682	**
SAR_40.49	-0.64906	8.60E-03	**
SAR_U	-0.9767	2.42E-05	***
NOS_M	0.63293	0.006025	**
OR_0_0_0_1_0_0	0.35823	0.036385	*
OR_1_0_0_0_0_0	0.37631	0.010876	*
C_LosAngeles	0.57783	0.002508	**
C_NewYork	-0.97769	2.02E-05	***
C_Philadelphia	-1.10614	1.49E-06	***

Using the p-values as an indicator, I built six more logistic regression models, using different amounts of significant and non-significant p-value features. All models use the Upsample training data with near zero variance features removed. In the first test I used only features with *** significance for their p-value (shown above). This gave a decrease in ROC value. For the next 2-6 tests, I looked at all features and removed two at a time where the p-value is largest. The table below depicts the results.

	ROC	Sens	Spec	Accuracy	Kappa
All Features	0.7045727	0.6580142	0.6516004	0.6547949	0.3096058
Test 1	0.7117008	0.6668669	0.6446061	0.6557341	0.3114709
Test 2	0.7125776	0.6668709	0.6427114	0.6547858	0.3095781
Test 3	0.7133094	0.6706845	0.6529146	0.6617808	0.3235857
Test 4	0.7144009	0.6719544	0.6465331	0.6592393	0.3184839
Test 5	0.715136	0.6655527	0.648452	0.6569926	0.3139988
Test 6	0.7140929	0.6718858	0.642655	0.6573299	0.3145782

As you can see the optimal ROC value was obtained using 'Test 5.' I used this model to predict with the validation and test set. I then compared this model result with the original model using all features. The results show a very small decrease on both the validation and test AUC.

Model results – Validation set

Data	AUC	Sens	Spec	Accuracy	Kappa
Normal	0.6956	0.6698	0.6053	0.6492	0.2542
New	0.695	0.6605	0.5987	0.6408	0.2387

Model results – Test set

Data	AUC	Sens	Spec	Accuracy	Kappa
Normal	0.7596	0.7271	0.6917	0.7151	0.3975
New	0.7587	0.7291	0.6917	0.7164	0.3998

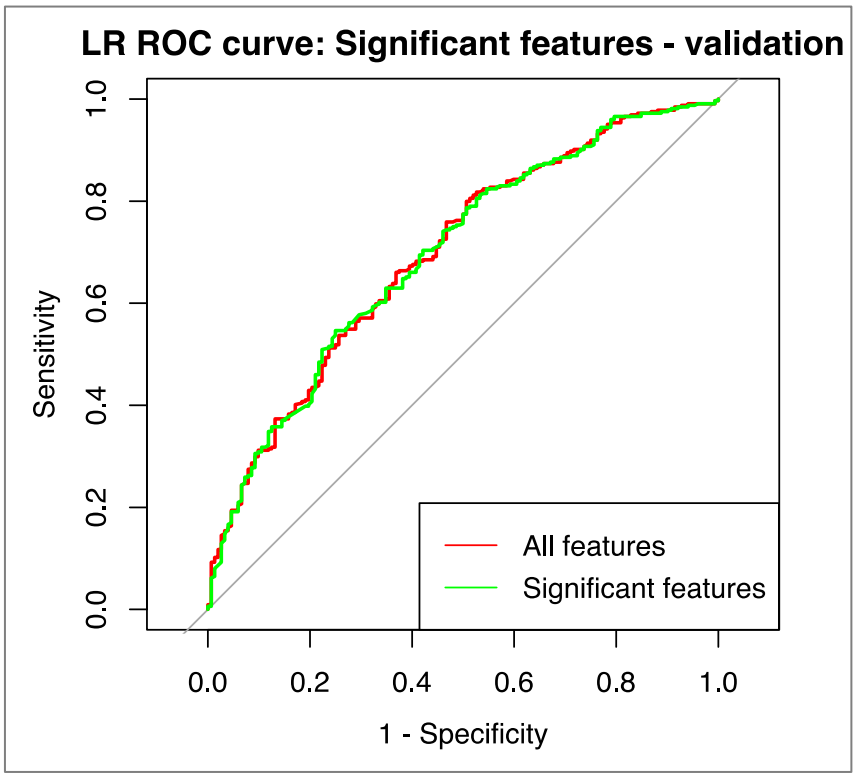

Figure 15 Logistic Regression ROC Curve - Significant Features - Validation

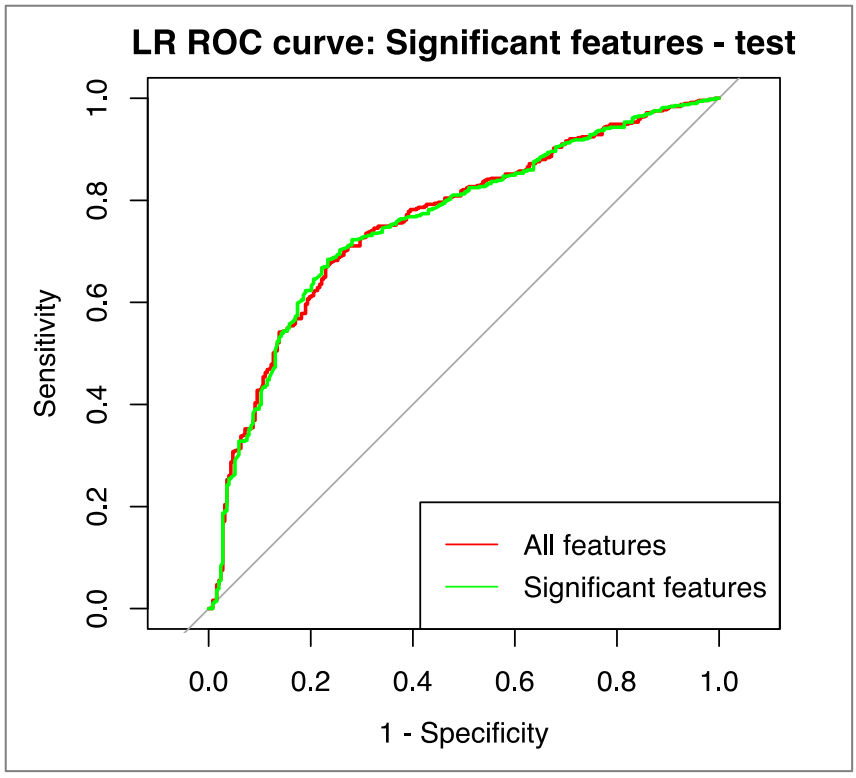

Figure 16 Logistic Regression ROC Curve - Significant Features - Test

Neural Networks & KNN

Finally, I used the Upsampled train data with near zero variance removed to create two new predictive models using Neural Networks and KNN. Using optimal tuning methods, the final Neural Networks model used size 10 and decay 0 and the final KNN model used k equals 3. The Neural Networks model achieved a higher performance on both validation and test sets.

Model results – Training set

Model	ROC	Sens	Spec	Accuracy	Kappa
Neural Network	0.7643289	0.67268	0.7424817	0.7075574	0.4151455
KNN	0.7516493	0.6019915	0.7463557	0.674185	0.3483525

Model results – Validation and Test correct predictions

Model	Validation	Test
Neural Network	80.88%	65.59%
KNN	74.16%	61.56%

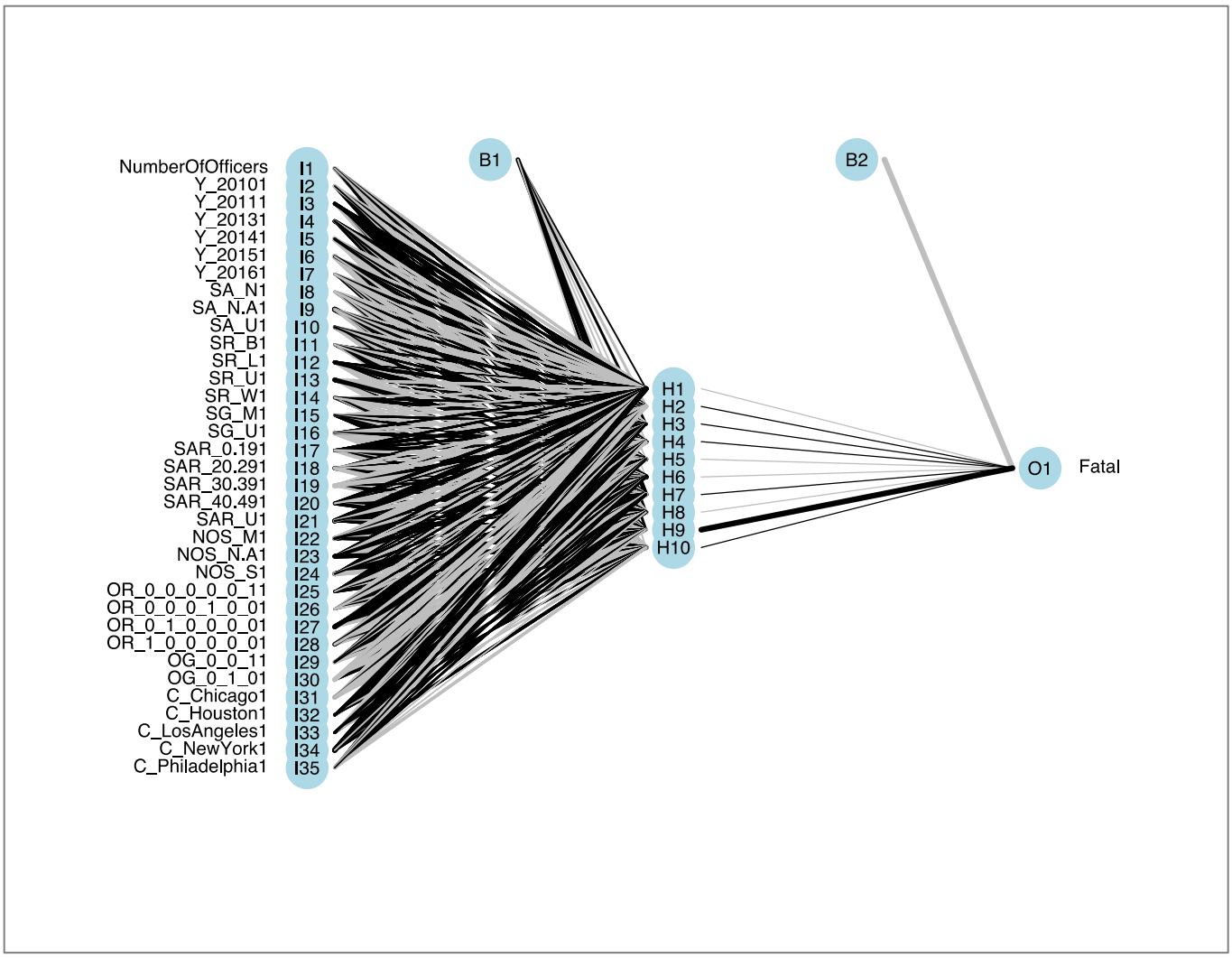

Figure 17 Neural Networks model - visualisation

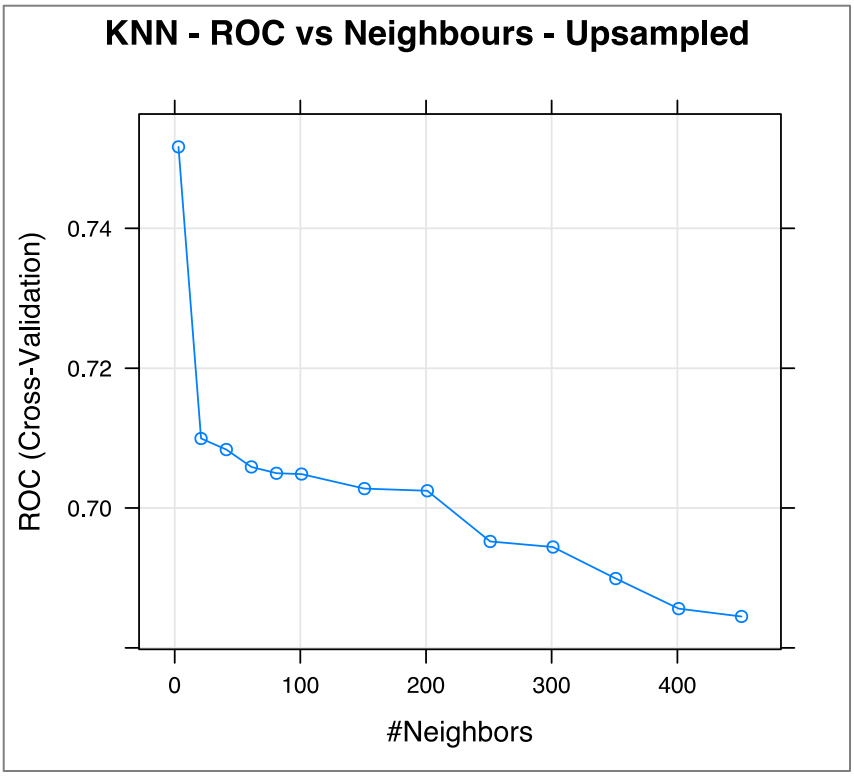

Figure 18 KNN Optimum Nearest Neighbours for ROC

Compare model performances

Looking at the performance across all models the Random Forest model using the Upsampled training data achieved the best results across the training, test and validation sets. The overall best result was achieved on the validation set with 98% AUC and a 95% prediction accuracy. It was interesting to see the Number of Officers as the strong predictor in both the Random Forest and Logistic regression model, giving strong evidence to show the increase in officers increases the likelihood of the incident being fatal. This could be as the number of officers called to a scene increased with the severity of the incident or may be as officers need a need to protect their colleagues, it would be interesting to look into this further.

To develop this project, I would be interested to look further into the relationships between the cities and the subject being Fatal, as well as the time of the incident. I chose to remove the time feature for this project, only looking at the year the incident took place. However, this feature could potentially be utilised for a time series project that looks at what factors effect the fatal to non-fatal field over time. Looking briefly into this I found a report that depicts the police gun fire policies that have been implemented each city. Looking at this data I found that the average percentage of policies implemented in the cities documented was 39.8%. This number increases to 87.5% in Philadelphia, which logistic regression showed the subject to be less likely to be fatal in that area. Development in this area could be used to evidence the need for police gun fire polices.

Use of Force Policy	Requires De-Escalation	Has Use of Force Continuum	Bans Chokeholds and Strangleholds	Requires Warning Before Shooting	Restricts Shooting at Moving Vehicles	Requires Exhaust All Other Means Before Shooting	Duty to Intervene	Requires Comprehensive Reporting
New York								
Newark								
Norfolk								
North Las Vegas								
Oakland								
Oklahoma City								
Omaha								
Orlando								
Philadelphia								
Phoenix								
Pittsburgh								
Plano								
Portland								
Raleigh								
Reno								
Riverside								

Figure 19 Screenshot from the report, showing police gun crime policies implemented

Conclusion

I have utilised and experimented with many of the Classification Machine Learning techniques learnt across the module. Whilst the pre-processing stage of the project was a challenge, I felt the project benefited overall from decisions made at this stage. With more time I would look at further classification techniques such as SVM and extend my analysis using Neutral Networks, as this achieved promising results. While this project was undertaken to practice Classification Machine Learning approaches, the results highlight the importance of data analysis for public policy.