Suoyi Yang
Christine Lee
Meng Lin
6/15/2018

STATS 101C Final Report

Data Cleaning

Before we could start building our models, we made several modification to the variables in both the training and testing files in order to clean the data. In 'Nature of Stops', there were hundreds of different levels and categories in both the training and testing dataset. In order to clean the variable, we took all the variable's data in training and testing and grouped them into 20 main categories: Aggravated Assault, Assault, Burglary, Call for Service, Car Related, Crime, Disturbance, Homicide, Hostage or Major Emergency, Larceny, Mental Health, Narcotics, Off Duty, Other, Robbery, Shots Fired, Suspect, Suspicious Entity, Warrant, and Weapon Possession. In 'Officer Gender', several observations in the training and testing data had different combinations and permutations of M and F depending on how many officers were at the scene of the crime. To clean this variable, we decided to only have two levels: M (male) and F (female). We grouped all original levels that started with M into level M and all levels that started with F into level F. We also tried to have three levels: M, F, and Mixed, with levels that contained only M's sorted into M, all levels with only F's sorted into F, and levels with a combination of M's and F's sorted into Mixed. However, when we ended up using 'Officer Gender' as a predictor in our model, it played a more significant role when it was cleaned using the first method, and thus we decided to use our original cleaning method. We cleaned 'Officer Race' in a similar way after testing our cleaning methods in our models. Since there were so many NA's in the training and testing data, we decided to impute the NAs. For numerical variables, we set the NAs to the average value, and for categorical, we set the NAs according to the most common level.

Description of Model

We experimented with several different models before settling with a tree-based model. We first tried the logistical model and attempted to find the best subset of predictors by using the 'summary' function with 'glm' to find the most significant predictors, as well as trying the 'bestglm' function. In both models, 'Number of Shots' was deemed a significant predictor of 'Fatal'. We built several models using different cleaning methods and occasionally 'Subject Race' and 'Number of Officers' would also be included in the best subset of predictors. However, when we used these models to predict the test cases, our classification rate was not great and our kaggle score was consistently around low .70s and high .60s. We then tried to use a tree model, and our predictions improved significantly. We used using a combination of logic and results of the best subset predictors from our attempts with 'glm' and 'bestglm' in order to decide which variables we should start off with in our tree model. We ended up using 'Subject Armed', 'Subject Race', 'Number of Shots', 'Officer Race', 'Officer Gender', and 'Subject Gender' in our tree. In the end, only 'Subject Race' was used in our tree and gave us the best/smallest misclassification rate in our training dataset.

However, when we used this model on our testing dataset, we still got a low kaggle score, likely due to several NAs in both 'Number of Shots' (991) and 'Subject Race' (454) of the testing data. Since there are only 1400 test cases to begin with, this means that the majority of the cases/observations in the testing data are being predicted using values generated by imputing NAs, which are probably not necessarily accurate. Although we initially omitted 'Date', 'Notes', and 'Full Narrative' from our model because we believed that those variables would be too different from one observation to the next to be useful in training, we later realized that 'Notes' and 'Full Narrative' contained useful information about fatalities. Thus we decided to first determine the fatalities of some test cases using only 'Narratives' and 'Notes'. We used 'gprepl' and indexing to subset test cases with key words("fatal", "non-fatal", "pronounced dead", "death", etc.) and set their fatalities accordingly. Then we used our tree model from before to predict the remaining test cases whose 'Fatal' are still unpredicted due to missing or unuseful 'Notes' and 'Full Narratives'. As a result, This gave us a much better kaggle score (in the .80s). We also tried the tree-model method using 'gini' as well as random forest, but they only ended up decreasing our correct classification rate.

Best Model MSE/Classification Rate

```
Classification tree:
tree(formula = Fatal ~ SubjectArmed + SubjectRace + SubjectGender +
    SubjectRace + NumberOfShots + OfficerRace + OfficerGender,
    data = train)
Variables actually used in tree construction:
[1] "SubjectRace"
Number of terminal nodes:  2
Residual mean deviance:  1.291 = 2462 / 1906
Misclassification error rate: 0.3695 = 705 / 1908
```

For our tree model, the best misclassification error rate was 0.3695 for our training data. For our testing data, our best classification rate was 0.83095 for 30% of the testing data and 0.78979 for the remaining 70% of the testing data.

Why the Model Works Well

Our model works well because we tried a variety of models using several cleaning methods before deciding on a final set of predictors and a final model based on lowest misclassification error rate in the training data (which in turn led to our highest correct classification rate in our testing data). Imputing the NAs was very helpful since there were so many NAs in both the training and testing data that even with a great model, we would have ended up with many NAs in the prediction of 'Fatal' in the testing data. However, because so many of the data in the variables we used to predict the testing cases with ('Number of Shots' and 'Subject Race') were generated by us and not actually part of the original data, solely using those two predictors and the tree model would not have produced great submission predictions. Thus using keywords from test cases with useful information in 'Notes' and 'Full Narrative' to predict 'Fatal' for those cases, and then predicting the remaining test cases with our model helped to eliminate some of those uncertainties from using imputed data.