Team M&M
Megan Mets and Mounika Narayanan

STAT 101C Final Project

ABSTRACT

For this Kaggle Competition, we were given a training and testing dataset that included data from shootings by police officers to be used in predicting whether or not a fatality occured. The training dataset came with 17 predictors to explore including: id, date, number of subjects, fatal, subject armed, subject race, subject gender, subject age, nature of stop, number of shots, number of officers, officer race, officer gender, department, full narrative, city, and notes. The testing dataset included the same predictors although fatal was not included as this is what we aimed to predict. Since Kaggle allowed three submissions a day to check how our model is performing, we worked intensively over the past few weeks creating predictors, fitting different models, and checking the misclassification rate to arrive at our final submission model.

DATA PREPARATION

Before beginning our analysis, we cleaned and transformed the existing predictors and removed any predictors that we felt were unnecessary in predicting fatality. In addition to this process, our team created new predictors that we felt would help improve our results.

We first began by recoding the "Fatal" variable in the training data. Since we knew that the testing data had changed all of the unknown values to no, we began by changing "U" to "N" and "F" to "Y". The "N" represents that there was not a fatality while the "Y" represents that there was a fatality. We then created a cleaning function to apply to the training and testing data in order to clean some of the existing predictors, create new predictors, and remove some unnecessary predictors. When we examined the training data, we noticed that the "Notes" column included helpful information that we could use in our model, but it was unhelpful in the form it was given in the original data. We then proceeded to change all of the information in the Notes column to lowercase and then used the grepl() function in R to extract whether the row included keywords that could be used as a new predictor. The first predictor we created was the "hit" predictor that stated whether a police encounter included a hit or not. If the row included the words "no hit," the "hit" column was given a 0. If the row included the words "hit" or "shot," the "hit" column was given a 1. Otherwise, the row stated "U." A similar process was used in creating our "newfatal" and "weapon" columns. The next step in our function was to clean the "OfficerGender" and "OfficerRace" columns. Since there were multiple entries in these columns joined together by a semicolon, we wanted to clean the column to show the most

represented officer gender and race in the encounter. We felt that the model would be able to interpret one entry better. We also had to recode "OfficerRace" to the six races that were included in the data key: White/non-hispanic, Black/non-hispanic, Asian, Latino, Other, and Unknown. Next, we changed the "Date" column to be understood as a date in R. From this we extracted the month and the year in order to create the new columns: Month, season, and Year. We defined the 4 seasons as: spring (March to May), summer (June to August), fall (September to November), and winter (December to February). Through research done on this topic, we felt that the year, month, and season would be helpful variables in predicting fatality. Next, we did some more recoding with the "SubjectAge" and "NumberOfOfficers" columns. We recoded subject age as a factor with 7 levels: 0-19, 20-29, 30-39, 40-49, 50-59, 60+, U and number of officers to a factor with two levels: 1-3 and 4+. We removed the "Notes" column because we already extracted the necessary information from the column and we also removed "NumberOfShots" because 72% of the column was NA and "NatureOfStop" because 64% was NA. In all of the remaining columns, we changed the NA's to "U" because many modeling techniques do not work well with a lot of NA's.

BEST MODEL

Through this process we tested several different models. The first model we tried was the tree model using library(tree), which gave us a misclassification rate of around 28%. We thought this model would do well because almost all our predictors were categorical and trees generally generate flexible models. In pursuit of finding a better model to improve our score, we tried logistic regression with interaction of our race and gender variables for officer and subject; however, the model did not perform well. We also tried a neural network and lasso regression, both of which did not improve our score.

The model we ended up using was the boost model which uses library(gbm). We used the following predictors in our model: SubjectArmed, SubjectRace, SubjectGender, SubjectAge, OfficerRace, OfficerGender, Year, hit, newfatal, weapon, and NumberOfOfficers. This model gave us a mean squared error (or misclassification rate) of 21.667%.

WHY BOOSTING MODEL WORKS

Boosting worked the best because it grows trees that are dependent on previous trees. Thus, this algorithm improves weaker models by "boosting" them to become stronger.

Additionally, it worked well because we have a lot of predictors in our data, creating a large feature space. The gbm function performs well with data that has large feature spaces because it can therefore choose the important features and their relative weights. The tree model worked well because it was able to utilize our large feature space to create a flexible model for predictions, but it probably did not work as well as boosting since it did not allow us to train different trees or improve different trees to create a more refined

model. The linear regression model, lasso, and neural networks did not work that well because we have mostly categorical predictors, and these models perform better with more numeric predictors. The additional columns hit, newfatal, and weapon definitely helped our models by providing more information that was extracted from the Notes column to help us better predict fatality.