Data Mining Group Project

Kaggle Contest – SF Crime Rate

Team members: Wei Zheng, Amit Talapatra, Wendy Zhang

Background

We chose the San Francisco Crime Classification competition on Kaggle our group project. From 1934 to 1963 San Francisco was known for housing some of the world's most notorious criminals on the inescapable island of Alcatraz. In modern days, although there has been a drastic change in the environment and a decrease in crime overall, there is still no scarcity of crime in the city.

The dataset included 12 years of crime reports from 2003 to 2015 from across all of San Francisco's neighborhoods. The goal was to predict the category of the crime that occurred given the time, date, and location from the crime reports.

Approach

We first examined the dataset to determine relationships of the variables with the target variable (Category) in the training set. We noticed that the Description and Resolution variables from the training set were not present in the test set so these were removed.

To establish baseline scores we initially tried running the unmodified raw data in SAS Enterprise Miner with neural network and decision tree models. These both achieved misclassification rates of around 0.67 which received scores on the Kaggle leaderboard of around 2.9. As another baseline, we submitted randomly generated values, resulting in a score on the Kaggle leaderboard of around 32. The leaderboard uses the log loss value from the results of the model on the test set as its scoring metric.

The tools we used to build models included SAS Enterprise Miner, H2O, R packages, and Python packages. Methods used included: PCA (for latitude/longitude), feature engineering from existing data, adding temperature data, random forest models, deep learning models, gradient boosting, and extreme gradient boosting. Table 1 shows an abbreviated version of how we tracked our model development and compared the success of our methods.

Table 1. Models run and their corresponding Kaggle scores.

Description	Tool Used	Kaggle Score
PCA + XGBoost (nrounds 30)	R	2.29054
PCA + XGBoost (nrounds 20)	R	2.29127
PCA + XGBoost (nrounds 100)	R	2.29254
PCA + XGBoost (nrounds 15)	R	2.2968
Average of best PCA/XGBoost model results and best Random Forest	R	2.30794
PCA + XGBoost (nrounds 100)	R	2.32871
Gradient Boosting Machine	R with H2O	2.3584
Gradient Boosting Machine	R with H2O	2.4002
Gradient Boosting Machine (Kaggle Python Script)	Python	2.49126
Random Forest w/o Feature Engineering (trees = 50)	R with H2O	2.53586
Average of best Random Forest model results and a basic Deep Learning model	R with H2O	2.53586
Random Forest w/ Temperature Data (trees = 50)	R with H2O	2.57264
Deep Learning	R with H2O	2.58016
Deep Learning with Defaults	R with H2O	2.60956
Original dataset with no feature engineering	SAS EM	2.89
Datasets with intersections and non intersections identified	SAS EM	2.9
Added Temperature	SAS EM	2.91
Random Forest w Feature Engineering (trees = 50)	R with H2O	3.88958
Gradient Boosting Machine	R with H2O	23.98042
Gradient Boosting Machine (kfold validation, 20 trees)	R with H2O	24.6867
Gradient Boosting Machine (kfold validation, 50 trees)	R with H2O	25.76583
Random Forest (Kaggle Python Script redone in H2O)	R with H2O	26.12752
Submitted Incorrectly	R with H2O	26.1701
Gradient Boosting Machine	R with H2O	26.24419
Submitted Incorrectly	R with H2O	26.37746
Random Forest with Features and PCA	R with H2O	26.42121
Submitted Incorrectly	R with H2O	26.42691
Random Forest (Kaggle Python Script)	Python	26.54636
Random Values	R	Around 32

Featuring Engineering and Dimension Reduction

We broke data and time into year, month, weekday, and hour. We obtained temperature data from National Oceanic and Atmosphere Administration (NOAA). Temperate data was added to the datasets as an additional variable for model building and binned into three categories of high, low, or average then. We also labeled addresses based on the whether the address is an intersection.

We attempted to utilize zip codes to extract economic data from external sites to add on to existing datasets to more accurately classify crime categories. To do that we needed to tie latitude and longitude to zip codes. However, we were not able to find free services to include this feature. As an alternative, we performed principle component analysis (PCA) of the latitude and longitude coordinates to process this information.

Parameter Tuning and Cross Validation

To efficiently search the best combination of parameters, we set up a H2O cluster of 300 cores for grid search. For GBM, the main parameters we modified were number of trees, learning rate, and maximum depth. For Random Forest, the parameters we tuned were number of trees, randomly chosen subsets of the features, and maximum depth. For R XGboost library, we manually adjust the number of iterations that the model ran for. In addition to the above parameter tuning, we turned on k-fold validation to avoid over fitting and to help us choose the best model. The cross validation scores produced based on our training set helped us estimate what kinds of scores our test set would achieve on the Kaggle leaderboard. This let us improve our model without having to submit our results to Kaggle each time to estimate performance.

Model Selection

From the wide range of methods applied, we identified some models and approaches that clearly worked better than others. The highest score we achieved on the Kaggle leaderboard was 2.2905, placing us at a rank of 109. However, the method used here was largely based on a script already available on Kaggle which combined some feature engineering of the date and time variables, PCA of the latitude and longitude variables, and extreme gradient boosting for the model itself. To improve on the script found on Kaggle, we experimented with parameter tuning and found that adjusting the number of the iterations for the model improved our score.

The second best approach was using the gradient boosting machine (GBM) R package and using parameter tuning to see what parameters resulted in better performance. The best combination of parameters used was ntrees = 200, learn_rate = 0.1, and max_depth = 10. This achieved a score of 2.3584 of the competition leaderboard.

The third best approach we identified involved using random forest models through H2O. The best score achieved was 2.5726 using an ntrees value of 50. It is likely that further parameter tuning on this model could have resulted in a better score, but since two other approaches resulted in better scores, we focused our efforts on improving those models.

Key Takeaways

This competition provided a more complex problem than some of the exercises we have attempted thus far in this class. In our attempts to solve this problem, we learned several lessons that can be applied to improving our skillset when dealing with advanced data science problems. For example, we realized that feature engineering (particularly to break up the time and data variables, did not always help our score. Although our assumption was that breaking out the time of day and time of year would be essential to identify the type of crime, there were cases where this had a minimal effect or even hurt our score. Overall, appropriate model selection had the largest effect on our scores and, as we work on other problems, we will have to develop a stronger understanding of where to use particular types of models. This can also be applied to our understanding of model parameters. Given our limited experience with some of these models, much of our parameter tuning started with trial and error. By expanding our knowledge of some of these models, and the functions that their parameters serve, we will likely save time with future data science problems by choosing better starting parameters.