Final Project Report

Team: LQZ

2017/6/5

Team members: Qiang Fei, Lingjue Xie, Zheqin Li

DATA CLEANING

When cleaning the training dataset, we found that most NAs were in the response variable "elapsed_time" which would prevent us from predicting. Therefore, we applied na.omit() to all the NAs in the training dataset.

In the original testing dataset, all NAs were in the variable "dispatch sequence," which shows how many vehicles were dispatched in the same incident. Since there were only around 500 NAs, we assigned numbers one by one to the missing values. For example, if a single incident had fewer than or equal to 2 vehicles in total, we assigned 1 to the "dispatch sequence." If the incident had more than 2 vehicles in total, we assigned the mean of the closest observation from the same incident. The number of obervations with NA is small, so we don't expect their predictions to make a significant difference in the final predictions.

VARIABLES

variables we have looked at: year, First in District, Dispatch sequence, Dispatch status, Unit Type, PPE Level, fd, incident, Dispatch count, hour, ICHourDiv, District order.

variables we used in our final model: year, First in District, Dispatch sequence, Dispatch status, Unit Type, PPE Level, fd, Dispatch count, ICHourDiv.

When choosing the variables, we thought about some inference questions. Might the "elapsed_time" be increased from year to year? Might the "elapsed_time" be different in different district, at different hour during the day, based on different emergency level of the incident?

After considering a lot about the possibilities of different relation between the predictors and the response variable, we included predictors "year," "First in District," "Dispatch sequence," "Dispatch status," "Unit Type," "PPE Level" from the original dataset. Besides, we created new variables including "fd," "incident," "Dispatch count," "hour," "ICHourDiv," and "District.order," though not all of them appear in the final model.

- **"fd" and "incident"** – "fd" and "incident" comes from the variable "incident.ID" in the original dataset. "fd" is the first part and "incident" is the second part of the original "incident.ID." The variable "fd" has only two levels and we treat it as a categorical variable, while the "incident" is a large number that we suspected if it would have some influence on the "elapsed_time." After testing for multiple rounds, we finally decided to include "fd" and discard "incident."
- **"hour"** – We thought that the hour at which the incident occured might affect the "elapsed_time" due to multiple reasons such as traffic and working state of firemen. Therefore, we took the hour part from the variable "Incident.Creation.Time..GMT," which shows the exact time of the incident, and built the new variable "hour." In other words, "hour" shows which hour of the day the incident happened, ranging from 0 to 23.
- **"ICHourDiv"** – When we look at the mean elapsed time for each hour, we noticed that the value for the 9th hour (9am) was around 900, significantly different from the mean elapsed time of other hours which fall between betweeen 400 and 700. Therefore, to cut the number of variables, we divided "hour"

into four parts: 0-5, 6-11, 12-17, 18-23 but have hour 9 seperated as the fifth category in the variable "ICHourDiv."

- **"Dispatch count"** – We also thought that the total number of vehicles sent out in one incident indictes the severity of an incident, and thus might affect "elapsed_time." Therefore, we created the variable "Dispatch count," which is the count of vehicles used in the same incident.
- **"District.order"** – We ordered the variable "First.in.district" through their mean elapsed time and wondered if it could be treated as a numerical variable. However, after rounds of test, we thought that it worked better as a categorical variable and thus did not use this variable.

MODEL

We used extreme gradient boosting (package: xgboost) to build our model, as we are interested in predictions only. Because xgboost function does not run categorical variables properly, we used dummyVars() function to transform the factors so that each level of each factor corresponds to a column in train_used with levels or 0 and 1.

In order to test the parameters and variables included, we split our training_final data by 80% and 20% to test and did a cross-validation on them to find the best "max_depth" and "nrounds" of xgboost(). The range of each parameters we have tried are listed below:

max_depth: 2-7 (Our final choice: 5)

nrounds: 5-20 (Our final choice: 12)

eta(learning rate): 0.2-0.4 (Our final choice: 0.3)

After we found the optimal parameters according to cross validation, we tried models around these values. When we decided the models to be submitted, we also referred to the similarity of five number summary between predictions and the "elapsed_time" from the whole training dataset. Our final model is the optimal combination of parameters and variables that we have found.

BEST MSE

Our lowest MSE is 1373993.70961 (on kaggle). It is from the model we built using xgboost. We tested different numbers on the parameters and finally chose eta = 0.3, max.depth=5, and nrounds=12.

EVALUATION

First, Extreme gradient boosting is a good choice due to its own algorithm. It starts with parts of variation that is easier to be explained and then gradually improves.

Second, we plotted the importance of predictors using the function xgb.plot.importance() to see which variables have more effects. For example, after we built our own variable "dispatch count," the plot shows that it is quite significant and we decided to put it in our final model.

Last but not least, we based our choices of best parameters of xgboost() on cross-validation, which provided insight on the parameters that might lead to a better model.

The model might be improved if we had found a good way to show correlation within the same incidents in the model, as we noticed that the "elapsed_time" should be highly correlated for the same incident.