Stats 101C Final Report

Team Stats Trek

Li, Xiao, leo1995@ucla.edu
Huo, Jiahao, jhuo831alex@g.ucla.edu
Wang, Jia Qi, jiaqiwang@g.ucla.edu

Data Cleaning and Feature Engineering

i) Dealing with NAs

First of all, we removed all the NAs in the elapsed time since it is our reponse variable. For other NAs in PPE.Level and Dispatch.Sequence, the package we used (xgboost) has a build in feature to imputate them.

```
library(xgboost)
library(caret)
library(Metrics)
library(dplyr)
library(mlr)
library(Hmisc)
library(checkmate)
library(ggplot2)
library(mice)
full<-read.csv("~/Desktop/Project/lafdtraining updated.csv")
full<-full[which(!is.na(full$elapsed_time)),]
md.pattern(full)
```

```
##         row.id incident.ID year First.in.District Emergency.Dispatch.Code
## 2315060      1           1    1                 1                       1
##    2359      1           1    1                 1                       1
##      11      1           1    1                 1                       1
##              0           0    0                 0                       0
##         Dispatch.Status Unit.Type Incident.Creation.Time..GMT.
## 2315060               1         1                            1
##    2359               1         1                            1
##      11               1         1                            1
##                       0         0                            0
##         elapsed_time PPE.Level Dispatch.Sequence
## 2315060            1         1                 1    0
##    2359            1         1                 0    1
##      11            1         0                 1    1
##                    0        11              2359 2370
```

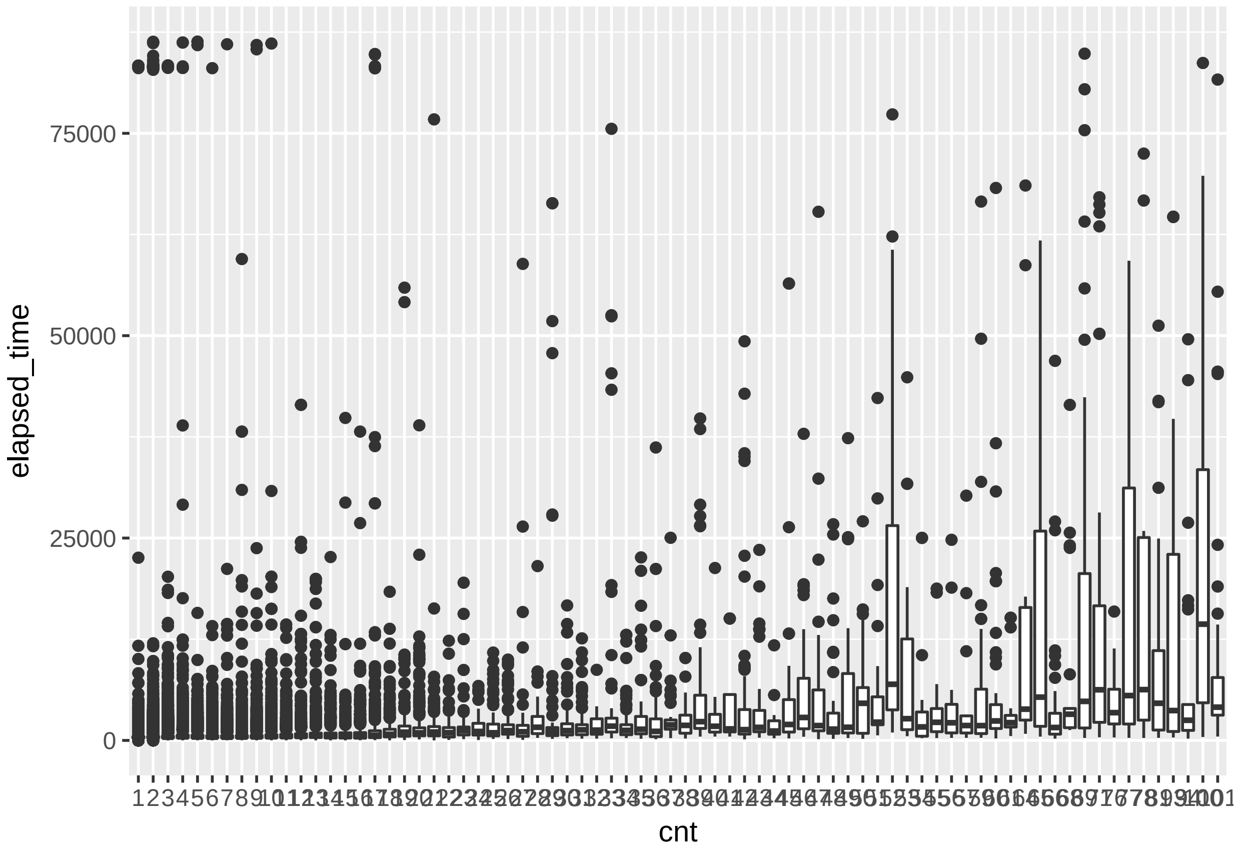
```
full$fd<-substring(full$incident.ID,1,7)
full$incident<-substring(full$incident.ID,9,length(full$incident.ID))
full$incident<-as.numeric(full$incident)
full$Incident.Creation.Time..GMT.<-as.numeric(full$Incident.Creation.Time..GMT.)
full$creation<-cut(full$Incident.Creation.Time..GMT.,
                     breaks = c(0,6*60*60,12*60*60,18*60*60,
                                max(full$Incident.Creation.Time..GMT.)),
                     labels = c(1:4))
full$fd<-as.factor(full$fd)
```

```
full$year<-as.factor(full$year)
full$First.in.District<-as.factor(full$First.in.District)
full$row.id<-NULL
full$Emergency.Dispatch.Code<-NULL
full$Incident.Creation.Time..GMT. <- NULL
```

ii) Original Data

Next, we separated incident.ID into 2 variables, fd (fire department; factor) and incident (incident numebr; numeric), using substring. For Incident.Creation.Time, we factorized it into 4 levels (00:00-6:00; 6:00-12:00, 12:00-18:00, 18:00-24:00). We have also tried to factorized it into 24 levels for better accuracy, but it occurs the problem of overfitting. In addition to fd, we also factorized year and First.in.District. After we removed NAs in elapsed_time, Emergency.Dispatch Code has only one level of factor. Therefore, we removed Emergency.Dispatch Code, incident.ID, row.id and Incident.Creation.Time. In addition to the above data manipulation, we engineered a new feature, count, using incident.ID. It is the number of vehicles dispatched in the same incident. We considered this an important feature. In the boxplot, we can see that more vehicles dispatched generally result in longer elapsed time.

```
c<-full %>% group_by(incident.ID) %>% summarise(cnt=n())
full<-merge(full,c,by.x="incident.ID")
set.seed(123)
percent1=sample(nrow(full), nrow(full)*0.25, replace = F)
try<-full[percent1,]
try$cnt <- as.factor(try$cnt)
ggplot(try, aes(x=cnt,y=elapsed_time))+geom_boxplot()
```



```r
full$incident.ID<-NULL
```

iii) External Data

We also utilized external data from LA Times to classify the First.in.District variable into division and battalion (http://boundaries.latimes.com/set/lafd-first-in-districts/). Division has 3 levels and battalion has 17 levels. However, after careful analysis, we found these two variables are not significant, because the MSE of our model increased from 1370754.87050 to 2102766.06553 after adding these two variables. Therefore, they are both dropped from the predicators in our final model.

Algorithm and Model

The algorithm we used is eXtreme Gradient Boosting (package: xgboost). It is an implementation of gradient boosted decision trees designed for higher speed and better performance. There are 10 variables in our final model; 6 variables (year, First.in.District, Dispatch.Sequence, Dispatch.Status, Unit.Type, PPE.Level) are from the original data set and 4 variables (fd, incident, creation, cnt) are newly created. The parameters we used are eta=0.3, max.depth=3, nround=100 and objective=reg:linear.

```r
dmy <- dummyVars("~.", data=full)
trainTrsf <- data.frame(predict(dmy, newdata = full))
outcomeName <- c('elapsed_time')
predictors <- names(trainTrsf)[!names(trainTrsf) %in% outcomeName]

cv <- 10
trainSet <- trainTrsf
cvDivider <- floor(nrow(trainSet) / (cv+1))

smallestError <- 10000
for (depth in seq(1,10,1)) {
  for (rounds in seq(1,20,1)) {
    totalError <- c()
    indexCount <- 1
        for (i in 1:cv) {
          # assign chunk to data test
          dataTestIndex <- c((i * cvDivider):(i * cvDivider + cvDivider))
          dataTest <- trainSet[dataTestIndex,]
          # everything else to train
          dataTrain <- trainSet[-dataTestIndex,]

          bst <- xgboost(data = as.matrix(dataTrain[,predictors]),
                         label = dataTrain[,outcomeName],
                         max.depth=depth, nround=rounds,
                         objective = "reg:linear", verbose=0)
          gc()
          predictions <- predict(bst, as.matrix(dataTest[,predictors]), outputmargin=TRUE)

          err <- rmse(as.numeric(dataTest[,outcomeName]), as.numeric(predictions))
          totalError <- c(totalError, err)
          print(totalError[i])
        }
    if (mean(totalError) < smallestError) {
      smallestError = mean(totalError)
      print(paste(depth,rounds,smallestError))
    }
  }
}

bst <- xgboost(data = as.matrix(trainSet[,predictors]),
               label = trainSet[,outcomeName],
               eta = 0.3,
               max.depth=3,
               nround=100,
               objective = "reg:linear",
               verbose=0)
```

Model Analysis

i) Parameters Used

We created our model bst using xgboost function. There are a lot of parameters in this function, but tuning all of them requires expensive computation. Therefore, we narrowed down our selection to eta, max_depth and nround, since these three parameters are considered the most important ones to control overfitting and underfitting probelm. Eta controls the rate at which our model learns patterns in data and it shrinks the feature weights to make the boosting process more conservative. Therefore, lower eta value leads to slower computation and it makes the model more robust to overfitting. Max_depth controls the maximum depth of a tree. Larger the depth, more complex the model, and higher chance of overfitting. Nround represents the number of trees to grow. Similar to max_depth, higher nround leads to higher chance of overfitting.

ii) Parameter Tuning

We used hypertuning to tune eta, which gives a final value of 0.05. However, after we utilized this value as eta, our model becomes too conservative, leading to underfitting. Therefore, we resolved to the default value 0.3. Max_depth and nround are tuned using a loop and then the fitted model was cross validated using 10-folds. It took 6 hours to run the loop and the final result we got is max_depth=3, nround=20. However, looking into the predictions of the model using max-depth=3 and nrounds=20, we found that there are a lot of same numbers. This implies that the choice of 20 seems to be too conservative and results in underfitting. Thus, we tried other numbers for nround such as 100, 150, 500, and 1000. Nround = 100 gives a low MSE of 1370754.87050. At first, we thought that higher nround will give higher accuracy but we neglected the problem of overfitting. Using nround = 1000 increased our MSE from 1370754.87050 to 1451594.45882, a severe overfitting of the model. Then, we tried 150 and 500, but they do not yield a better result than nround=100. After careful tuning and comparison, we decided our parameters to be eta=0.3, max.depth=3, nround=100.

iii) Final Adjustment

After we created the prediction, we spotted 2 negative values with row.id 21386 and 848737. Since both the training and testing data are modified so that there should not be any negative elapsed value, we manually changed these two predictions into 0. This modification decreased our MSE by 4.62581.

```
testing <- read.csv("~/Desktop/Project/testing.csv")
testing$elapsed_time<-0
testing<-testing[which(!is.na(testing$elapsed_time)),]
testing$fd <- substring(testing$incident.ID,1,7)
```

```
testing$incident <- substring(testing$incident.ID,9,length(testing$incident.ID))
testing$incident<-as.numeric(testing$incident)
c <- testing %>% group_by(incident.ID) %>% summarise(cnt=n())
testing<-merge(testing,c,by.x="incident.ID")
testing$Incident.Creation.Time..GMT.<-as.numeric(testing$Incident.Creation.Time..GMT.)
testing$creation <- cut(testing$Incident.Creation.Time..GMT.,
                        breaks = c(0,6*60*60,12*60*60,18*60*60,
                                   max(testing$Incident.Creation.Time..GMT.)),
                        labels = c(1:4))
testing$year<-as.factor(testing$year)
testing$First.in.District<-as.factor(testing$First.in.District)
testing$fd<-as.factor(testing$fd)
testing$incident.ID<-NULL
testing$row.id<-NULL
testing$Emergency.Dispatch.Code<-NULL
testing$Incident.Creation.Time..GMT. <- NULL

fulltest<-rbind(full,testing)

dmytest <- dummyVars("~.", data=fulltest)
testTrsf <- data.frame(predict(dmytest, newdata = fulltest))
pred <- predict(bst, as.matrix(testTrsf[,predictors]), outputmargin=TRUE)
fulltest$pred<-pred
result<-fulltest[fulltest$elapsed_time==0,]
testing <- read.csv("~/Desktop/Project/testing.csv")
final_result<-data.frame(row.id=testing$row.id,prediction=result$pred)
write.csv(final_result,file="attempt11.csv",row.names=F)
```

```
attempt11 <- read.csv("~/Downloads/attempt11.csv")
negative <- attempt11[which(attempt11$prediction < 0),]
head(negative)
```

```
##        row.id prediction
## 433321  21386  -39.56905
## 433324 848737 -330.72598
```

```
negative$prediction <- 0
head(negative)
```

```
##        row.id prediction
## 433321  21386          0
## 433324 848737          0
```

Best MSE

Our model achieved a score of 1370750.24469 on Kaggle. The final score is 1408337.13401.