Crime In San Fransisco

Alison Kingman, James LaManana, Lizzie Kumar

12/15/2015

Introduction

For our final project we entered a Kaggle data science competition on San Francisco Crime Classification. The competition's dataset provides nearly 12 years of crime reports from across all of San Francisco. The goal is to predict the category of a crime that occurred, given its time and location. More specifically, the competition requires that we submit a table of probabilities of each category that a crime could possibly be a member of. The prompt also encourages participants to explore the data set visually as well.

Importance

The Data

This dataset contains incidents derived from the San Francisco Police Department's Crime Incident Reporting system. The data ranges from 1/1/2003 to 5/13/2015. The training set and test set rotate every week, meaning week 1,3,5,7... belong to the test set, week 2,4,6,8 ... belong to the training set. Each of the training and test data sets both consist of well over 800,000 crimes. The fields in the given data set are as follows:

- Dates - timestamp of when the crime occurred

- Category - Type of crime that was committed. This is the response variable that we are attempting to predict. In total there are 39 different types of crimes in the category data field (train only).

- Descript - Represents a detailed description of the crime incident (training data only).

- DayOfWeek - The day of the week when the crime occurred.

- PdDistrict -Name of the Police Department District where the crime occurred.

- Resolution - How the crime incident was resolved (training data only)

- Address - the approximate street address of the crime incident

- X - Longitude

- Y - Latitude

Visualization

To visualize the data we created a few static charts, as well as a shiny application. The shiny application plots the locations of the top 5 most frequent crimes on a map of San Francisco. In addition, the application has a slider to select a specific year from the data set, and two checkbox groups: one to select the specific type of crime out pf the top five to be displayed and the other is a setting to look at crimes that happened either during the day or night. In addition to the geographic plot of the crime, the Shiny app also includes a bar chart that responds to the same inputs as before but displays the relative frequency of the crimes selected. This bar chart aims to make it easier to compare the amount of occurrences of the types of crimes currently plotted on the map. The visualization can be found via the following url `http://rstudio.campus.pomona.edu:3838/jrl31925/CrimeShiny/`

The goal of this analysis is to predict the type of crime given a location and time. This shiny app is a nice visualization of the data because it shows the data based on those two criteria. There is a day and night setting as well as a slider for the year component, and the data is plotted on a map of San Francisco, to display the geographic aspect of the data.

```
> library(shiny)
> #ui
> shinyUI(pageWithSidebar(
+    headerPanel("San Francisco Crime"),
+    sidebarPanel(
+      sliderInput('time.year', 'Year of crime occurance', sep='', value=2009, min=200
+      checkboxGroupInput("category", "Type of Crime Committed",
+                         c("Assault" = "ASSAULT",
```

```
+                           "Non Criminal" = "NON-CRIMINAL",
+                           "Drug" = "DRUG/NARCOTIC",
+                           "Vehicle Theft" = "VEHICLE THEFT",
+                           "Larceny" = "LARCENY/THEFT"), selected = c("Assault","Drug
+      checkboxGroupInput("daynight", "Crime Committed During the Day or Night",
+                         c("Day"="Day",
+                           "Night" = "Night"), selected = c("Night"))
+
+    ),
+    mainPanel(
+
+      plotOutput('newPlot'),
+      plotOutput('newPlot2')
+
+
+    )
+ ))


> library(shiny)
> library(XML)
> library(RCurl)
> library(mosaic)
> library(lubridate)
> library(dplyr)
> library(ggplot2)
> library(ggmap)
> #server.R
>
> map<-get_map(location=c(top =37.82, bottom = 37.7, right = -122.35,left = -122.52),
> crime.train <- read.csv("train.csv")
> crime.train2<-crime.train %>% mutate(Time = ymd_hms(Dates)) %>% mutate(Year = year(
+ mutate(DayNight = ifelse(hour(Time)>=18|hour(Time)<=6,"Night", "Day")) %>% mutate(c
> counts <- summarise(group_by(crime.train2, Category), Counts=length(Category))
> counts <- counts[order(-counts$Counts),]
> top5 <- crime.train2[crime.train2$Category %in% counts$Category[c(1,3:6)],]
> shinyServer(
+   function(input,output){
+     output$otime.year = renderPrint({input$time.year})
+     output$ocategory = renderPrint({input$category})
+     output$odate = renderPrint({input$date})
```

```
+       output$odaynight = renderPrint({input$daynight})
+
+       #data for map plot
+       dataInput <- reactive({
+         top5[(top5$Year %in% input$time.year
+               & top5$DayNight %in% input$daynight
+               & top5$Category %in% input$category),]
+       })
+
+       #plotting map
+       output$newPlot = renderPlot({
+         ggmap(map) +
+           geom_point(data=dataInput(), aes(x=X, y=Y, color=as.factor(Category)), alph
+           guides(color = guide_legend(override.aes = list(alpha=1.0, size=6.0),
+                                       title="Type of Crime"))
+       })
+
+       #Data for Bar Chart
+       dataInput2 <- reactive({
+         crime.train2[(crime.train2$Year %in% input$time.year
+                       & crime.train2$DayNight %in% input$daynight
+                       & crime.train2$Category %in% input$category),]
+       })
+
+       #get number of counts and add percentage
+       counts <- reactive({
+         dataInput2() %>% group_by(Category) %>%
+           summarise(Counts = length(Category)) %>%
+           mutate(percentage = Counts/nrow(crime.train2[(crime.train2$Year %in% input$
+                                               & crime.train2$DayNight %in%
+       })
+
+
+       #plot bar chart
+       output$newPlot2 = renderPlot({
+         ggplot(counts())+
+           geom_bar(stat = 'identity',aes( x = Category, y = percentage, fill = Catego
+       })
+
+     }
```

+)

Method

We chose a conditional inference tree for the classification of the data. This classification method work well for this analysis for a few reasons: first, like a standard CART algorithm, this method handles both continuous and categorical variables easily. Second, this model makes it much simpler for us to evaluate the model based on the Kaggle guidelines. Each crime has been labeled with one true class. For each incident, we must submit a set of predicted probabilities (one for every class). The conditional inference tree output includes a set of probabilities for each prediction, thus making it much easier to evaluate our model according to the Kaggle rules.

Conditional inference trees follow the same general algorithm as CART trees.

1. Choose variable to split on

2. Find best split point to partition on.

3. Repeat for the two partitions. Stop on stopping condition.

In CART trees, the variable splitting can be done by optimizing some parameter, such as the Gini index. In conditional inference trees however, we choose the variable to split on by using conditional inference through permutation test.

To determine the best variable to split on, we choose the null hypothesis $H_0 = \cap_{j=1}^{m} H_0^j : P(Y|X_j) = P(Y)$ where Y is the response variable and the $X_j s$ are the explanatory variable. That is, the null hypothesis is that the probability of making a prediction given an explanatory variable is exactly the same as making that prediction without any knowledge of the explanatory variable. Through a permutation test, we can calculate p-values for each of the explanatory variables. If all p-values are chosen above a certain threshold (for example, 0.05), then we do not reject the null hypothesis and we terminate the node. If we reject the null hypothesis, we choose the explanatory variable with the lowest p-value to split on.

Implementation

Before running the model, we first needed to modify our explanatory variables. We extracted the hour, month and year and made them factor vari-

ables. We also removed all data before 2005 because the amount of vehicle thefts, suddenly dropped after this year. This unnatural drop off in occurrences was likely to due a reporting change or some other outside factor. Finally, we filtered out unneeded columns such as resolution and description. In our analysis, we included hour, month, year and district as categorical variables and latitude and longitude as continuous variables.

```
> require(lubridate)
> require(dplyr)
> data= read.csv("../../train.csv")
> #Extract  hour, month and year from timestamp
> data = data %>%
+   mutate(Time = ymd_hms(Dates)) %>%
+   mutate(hour = as.factor(hour(Time)),
+          month = as.factor(month(Time)),
+          year=year(Time))
> #remove description, resolution address and date columns
> data = data[,-c(1,3,6,7,10)]
> #remove data before recorded before 2006
> data = data %>%
+   filter(year >2005)
> data$year = as.factor(data$year)
```

The Conditional Inference Tree model is computationally intense. Since our data has about 800,000 observations and 39 categories of crime, training the conditional inference tree with it takes a long time. Therefore, we made a few more modifications to cut down on our run time. We choose to keep all of the categories that had more than 10,000 representations in the training data and classified everything else as other, giving us a total of 13 categories. We decided to train the model on 50,000 randomly sampled observations from the training data, which took approximately 1.2 days.

```
> require(partykit)
> set.seed(100)
> trainsize = 50000
> trainind = sample(seq(1:nrow(data)), trainsize)
> train= data[trainind,]
> test = data[-trainind,]
> t = ctree(Category~., data = train)
```

Results

When implementing the model, we modified the categories of the training data. We tested the remaining data twice: once with the modified 13 categories and once with the original 39 categories. Note that in both cases, we used the same training set. We present the results for each in the form of a test error and logloss calculation. The log loss is defined as

$$logloss = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} \log(p_{ij})$$

N is the total number of observations and M is the total number of categories. $y_{ij} = 1$ if observation i is in category j and 0 otherwise. p_{ij} is the predicted probability that observation i is in category j. The logloss is the score that Kaggle teams compete to minimize.

First, we examine the results of test data with the modified categories. The test error is 70.6%. This error rate is higher than we would like .

```
> pred = predict(t, test)
> tab = table(pred, test$Category)
> error = (nrow(test) - sum(diag(tab)))/nrow(test)
> error

[1] 0.7059887
```

Next, we find the logloss of the test data. to be 2.31. On the Kaggle leaderboard, the logloss scores between 2 and 34 with low scores being preferable. About 5.7% of the teams have a score less than 2.31.

```
> probpred = predict(t, test, type="prob")
> N = nrow(test)
> logprobs = rep(1,N)
> for (i in seq(1,N)){
+    class = test$Category[i]
+    prob = probpred[i, class]
+    prob = max(min(prob,1-10^(-15)), 10^(-15))
+    logprobs[i] = log(prob)
+ }
> logloss = -sum(logprobs)/N
> logloss
```

```
[1] 2.316695
```

We repeat the calculation above for the test data with the original categories. The test error is 97.4% and the logloss is 10.4. About 79.3% of the teams on the kaggle leaderboard have a score lower than 10.4

Limitations and Conclusions

In our results, we have very poor test error rates. We examine possible reasons for this. First, if we examine the visualizations of the data from the shiny application, we see that most of the crimes tend to occur in the same area of Northeast San Francisco. Also, when we visualize the data by the time of day, there also does not seem to be a significant difference between daytime and nighttime crimes. Since the explanatory variables for all observations are so similar, the conditional inference tree is not able to accurately classify a crime to one of the many categories. Another issue is that training a conditional inference tree is very computationally intensive. It took 1.2 days to train on 50,000 data points which is only about 6% of the given training data. It is possible that the tree might not have been able to accurately build the tree since most of the observations have been left out. However, when we build the tree using 1000, 10,000 and 50,000 observations, the errors did not seem to improve when increasing the training size. Therefore, the cause for our poor errors is most likely due to the similarity of the explanatory variables across categories rather than our inability to train with the full data set.

Overall, a conditional inference tree probably was not the best model for this data. The data had too many observations and too many classification categories. A data set with fewer observations and classification categories might have better results.