Detroit Blight Analysis - Capstone Project

Introduction

This report is part of the Capstone Project in the Data Science at Scale course, and his objective is the creation of models for prediction algorithms in order to foreseen the possibility of a building be a "blighted building". This prediction is based on public data sets as: criminal incidents, permit violations and others.

Data

Detroit Open Data https://data.detroitmi.gov/ (https://data.detroitmi.gov/). Among the data available the data sets used were:

- Detroit-blight-violations.csv : Each record is a blight violation incident.
- detroit-demolition-permits.tsv: Each record represents a permit for a demolition.
- detroit-311.csv: Each record represents a 311 call, typically a complaint
- detroit-crime.csv: Each record represents a criminal incident.

Those files are composed of incidents and those incidents have an location available in form of latitude/longitude.

Loading data

```
d311 <- read.csv("./data/detroit-311.csv")
dBV <- read.csv("./data/detroit-blight-violations.csv")
dC <- read.csv("./data/detroit-crime.csv")
dDP <- read.delim("./data/detroit-demolition-permits.tsv")
```

Loading libraries

```
# Loading Libraries
options(digits = 7)
library(ggmap)
```

```
## Loading required package: ggplot2
```

```r
library(plyr)
library(sp)
library(caret)
```

```
## Loading required package: lattice
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tree)
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

Auxiliary functions

Some auxiliary functions were developed with the intention to facilitate the handling of data. Those functions are:

```r
# Parsing GPS coordinates define regular expression pattern
gpsParsing <- function(addr, p="\\(.*\\)"){
  r <- regexpr(p, addr)
  out <- rep(NA, length(r))
  out[r != -1] <- regmatches(addr, r)
  out <- gsub("[()]", "", out)
  lats <- unlist(lapply(out, function(x) as.numeric(strsplit(x, split=",")[[1]][1])))
  lngs <- unlist(lapply(out, function(x) as.numeric(strsplit(x, split=",")[[1]][2])))
  list(lat=lats, lng=lngs)
}

# Finds building ID based on Latitude/Logitude
findIds <- function(x = c(0.0, 0.0), coordID, bycolumn="id", digits = 4){
    x <- round(x, digits = digits)
    return (coordID[which(coordID[,"lat"] == x[1] & coordID[,"lng"] == x[2]),"id"])
}

# Sum of the total violations of neighborhood
neighbors_total <- function(x=c(0.0, 0.0), df){
    epsilon <- 0.0001
    return(sum(df[which(abs(df[,"latR"] - x[1]) < epsilon & abs(df[,"lngR"] - x[2]) <
epsilon), "freq"]))
}
```

Data cleansing

Analyzing the data sets which may be concluded is that many records not possessed latitude and longitude and the records that possessed latitude and longitude they were in a single variable. To correct these points the following measure was adopted:

- It Was used the function gpsParsing using the address to get the gps coordinates.

On that way correcting the latitude and longitude problem.

```
## Setting global configuratyions for cleaning
options(digits = 8)
epsilon <- 0.0001

## dBV dataset
dBVcoords <- gpsParsing(dBV$ViolationAddress)
dBV$lat <- dBVcoords$lat
dBV$lng <- dBVcoords$lng
dBV$FineAmt <- as.numeric(gsub("\\$","", as.character(dBV$FineAmt)), na.rm=FALSE)
dBV$FineAmt[is.na(dBV$FineAmt)] <- 0
dBV$AdminFee <- as.numeric(gsub("\\$","", as.character(dBV$AdminFee)), na.rm=FALSE)
dBV$AdminFee[is.na(dBV$AdminFee)] <- 0
dBV$StateFee <- as.numeric(gsub("\\$","", as.character(dBV$StateFee)), na.rm=FALSE)
dBV$StateFee[is.na(dBV$StateFee)] <- 0
dBV$LateFee <- as.numeric(gsub("\\$","", as.character(dBV$LateFee)), na.rm=FALSE)
dBV$LateFee[is.na(dBV$LateFee)] <- 0
dBV$CleanUpCost <- as.numeric(gsub("\\$","", as.character(dBV$CleanUpCost)), na.rm=FAL
SE)
dBV$CleanUpCost[is.na(dBV$CleanUpCost)] <- 0
dBV$JudgmentAmt <- as.numeric(gsub("\\$","", as.character(dBV$JudgmentAmt)), na.rm=FAL
SE)
dBV$JudgmentAmt[is.na(dBV$JudgmentAmt)] <- 0
dBV$totalfee <- dBV$FineAmt + dBV$AdminFee + dBV$StateFee + dBV$LateFee + dBV$CleanUpC
ost + dBV$JudgmentAmt

## dC dataset
dC1 <- names(dC)
dC1 <- tolower(dC1)
dC1[15:16] <- c("lng","lat")
names(dC) <- dC1
rm(dC1)

## dDP dataset
dDPcoords<- gpsParsing(dDP$site_location)
dDP$lat <- dDPcoords$lat
dDP$lng <- dDPcoords$lng

# Restore default options
options(digits = 7)
```

Below can be observed an inclusion of the columns latR and lngR that will store the rounded lat and long.

```
dBV$latR <- round(dBV$lat, digits=4)
dBV$lngR <- round(dBV$lng, digits=4)
dC$latR <- round(dC$lat, digits=4)
dC$lngR <- round(dC$lng, digits=4)
d311$latR <- round(d311$lat, digits=4)
d311$lngR <- round(d311$lng, digits=4)
dDP$latR <- round(dDP$lat, digits=4)
dDP$lngR <- round(dDP$lng, digits=4)
```

Data vizualization

On this section there will be presented one visualization of the information acquired on this analysis.

```
## Loading Detroit map
dMap <- get_googlemap(center = c(lon=-83.119128,lat=42.384713),maptype = "roadmap", si
ze=c(640,640),zoom = 11)
```

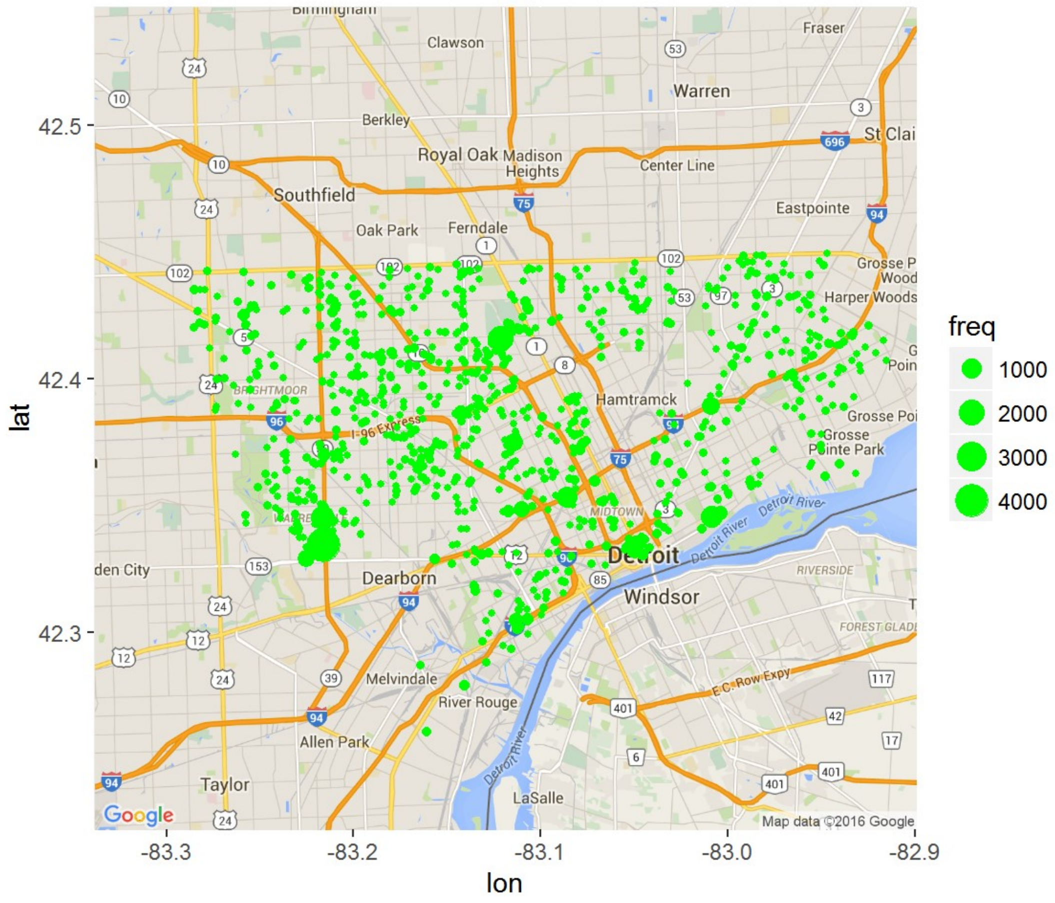
```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=42.384713,-83.1
19128&zoom=11&size=640x640&scale=2&maptype=roadmap&sensor=false
```

Ploting Blight Violation frequency

Here is presented the frequency calculation on the blight violation data set.

```
## Calculating frequencies of locations of dBV dataset
dBVloc <- ddply(dBV, .(dBV$lat, dBV$lng), nrow)
names(dBVloc) <- c("lat", "lng", "freq")
dBVloc <- dBVloc[order(-dBVloc$freq),]

## Presenting the information aquire above
dBVplot <- ggmap(dMap) + geom_point(aes(x=lng,y=lat,size=freq),data=dBVloc[2:1000,], c
olor = ("green"))
dBVplot
```

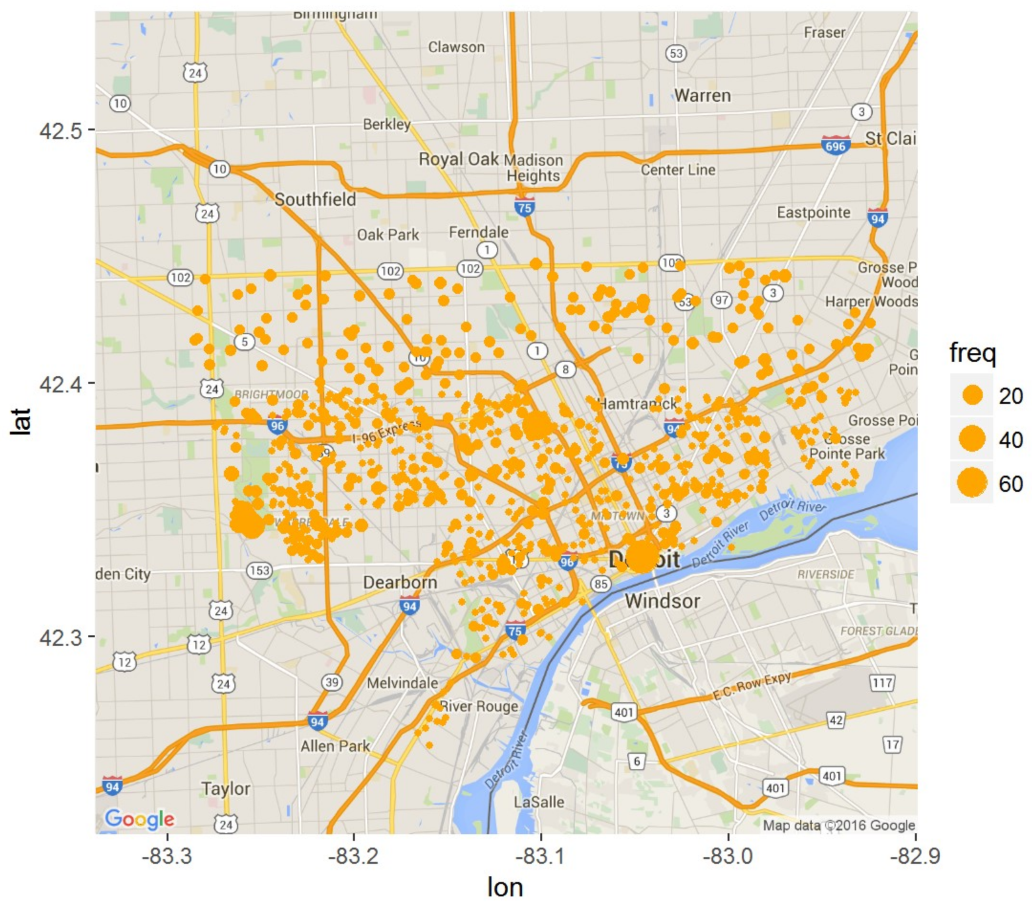


Ploting 311 call frequency

Here is presented the frequency calculation on the 311 call data set.

```
## Calculating frequencies of locations of d311 dataset
d311loc <- ddply(d311, .(d311$lat, d311$lng), nrow)
names(d311loc) <- c("lat", "lng", "freq")
d311loc <- d311loc[order(-d311loc$freq),]

## Presenting the information aquire above
d311plot <- ggmap(dMap) + geom_point(aes(x=lng,y=lat,size=freq),data=d311loc[2:100
0,], color = ("orange"))
d311plot
```

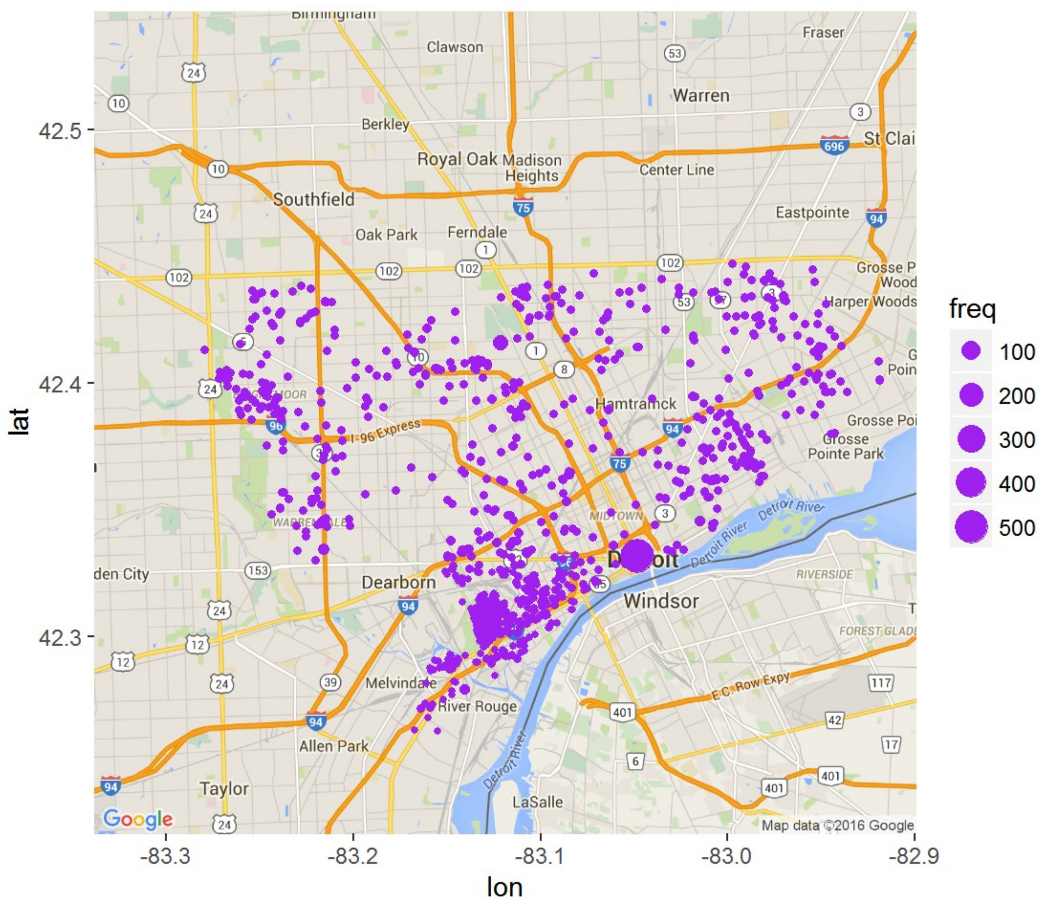


Ploting demolition permits frequency

Here is presented the frequency calculation on the demolition permits data set.

```
## Calculating frequencies of locations of dDP dataset
dDPloc <- ddply(dDP, .(dDP$lat, dDP$lng), nrow)
names(dDPloc) <- c("lat", "lng", "freq")
dDPloc <- dDPloc[order(-dDPloc$freq),]

## Presenting the information aquire above
dDPplot <- ggmap(dMap) + geom_point(aes(x=lng,y=lat,size=freq),data=dDPloc[2:1000,], c
olor = ("purple"))
dDPplot
```



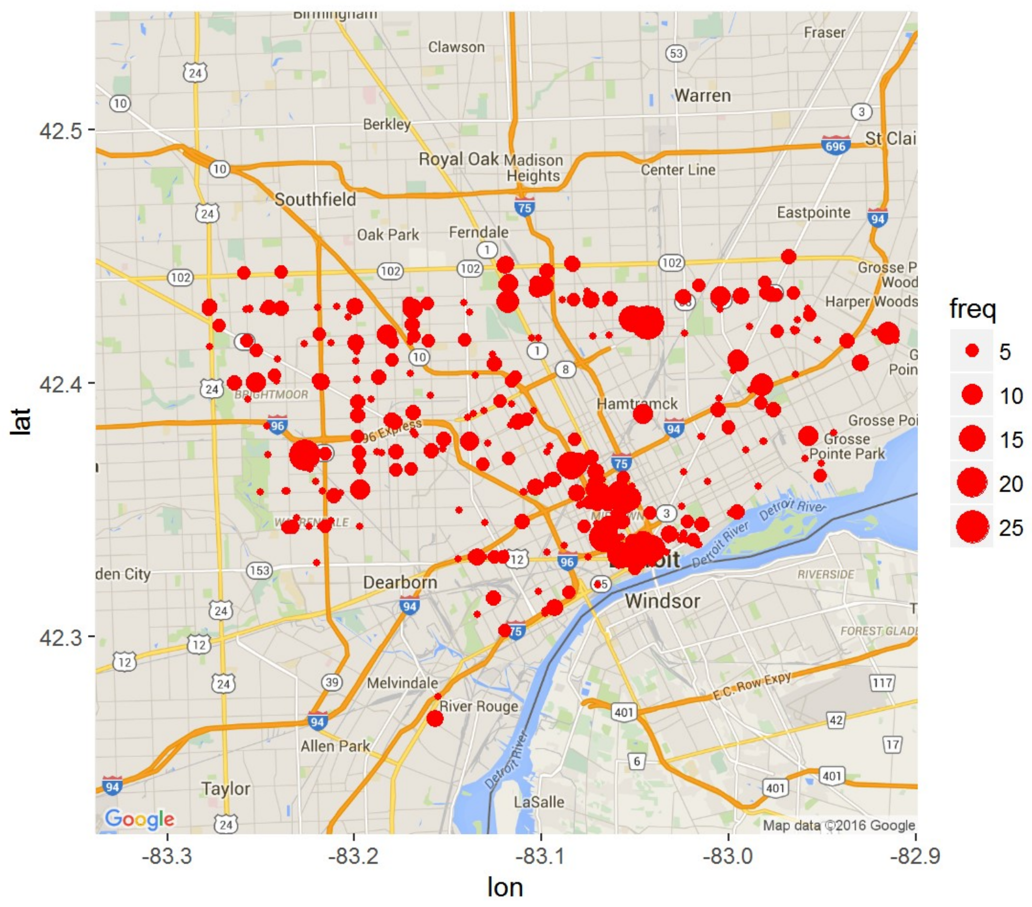
Ploting crime frequency

Here is presented the frequency calculation on the crime data set.

```
## Calculating frequencies of locations of dDP dataset
dDPloc <- ddply(dDP, .(dDP$lat, dDP$lng), nrow)
names(dDPloc) <- c("lat", "lng", "freq")
dDPloc <- dDPloc[order(-dDPloc$freq),]

## Presenting the information aquire above
## Calculating frequencies of locations of dC dataset
dCloc <- ddply(dC, .(dC$lat, dC$lng), nrow)
names(dCloc) <- c("lat", "lng", "freq")
dCloc <- dCloc[order(-dCloc$freq),]

## Presenting the information aquire above
dCplot <- ggmap(dMap) + geom_point(aes(x=lng,y=lat,size=freq),data=dCloc[2:1000,], col
or = ("red"))
dCplot
```



Model creation

This section is focused on the model creation, here will be presented the method that was used on the on the development of the model used on this report.

Merge of lat lon on the data sets and calculation of frequencies of the lat/lon entries

How can be observed below here all the lat/lon on all data sets are merged forming a unique set of lat/lon. And after the creation of the merged set the frequency was calculated in order to exemplify how many times a set of lat/lon appear on the data set.

```
## Creating lat lng subset
latList <- c(d311$lat, dBV$lat, dC$lat, dDP$lat)
lngList <- c(d311$lng, dBV$lng, dC$lng, dDP$lng)
coordList <- data.frame(lat = round(latList, digits =4), lng = round(lngList, digits =
4))

## Calculating frequency of coordList
coordFreq <- ddply(coordList, .(lat = coordList$lat, lng = coordList$lng), nrow)
names(coordFreq) <- c("lat", "lng", "freq")
coordFreq <- coordFreq[-coordFreq$freq,]
```

Cleaning merged set

Here how can be observed below the set was cleaned by the use of the unique() that as the name says make unique entries on the set. The NA entries were also removed, and was added the column ID. That as the name says is a unique ID for each entire.

```
## Unique locations
coordID <- unique(coordList)

## Removing NA
coordID <- coordID[!is.na(coordID$lat),]

## Adding ID collumn
coordID$id <- 1:dim(coordID)[[1]]
```

On this step a column ID is created on all main data sets and populated using the findID function created above.

```
## using function findID to populate id collumn
dBV$id <- as.numeric(apply(matrix(c(dBV$latR, dBV$lngR), ncol=2), 1, findIds, coordI
D))
dC$id <- as.numeric(apply(matrix(c(dC$latR, dC$lngR), ncol=2), 1, findIds, coordID))
d311$id <- as.numeric(apply(matrix(c(d311$latR, d311$lngR), ncol=2), 1, findIds, coord
ID))
dDP$id <- as.numeric(apply(matrix(c(dDP$latR, dDP$lngR), ncol=2), 1, findIds, coordI
D))
```

Filling the model collumns

This step is focused on the labeling of the entries as blight or non-blight.

```r
# Adding the number of blight incidents
coordID$blight <- FALSE
blightID <- dDP[which(dDP$PERMIT_DESCRIPTION != ""),"id"]
blightID <- unique(blightID)
coordID[which(coordID$id %in% blightID), "blight"] <- TRUE
```

Creating of the column of the number of blight incidents.

```r
## Adding the number of blight incidents
coordID$nB <- 0
dBVids <-  plyr::count(dBV$id)
names(dBVids) <- c("id","freq")
coordID$nB[1:(nrow(dBVids) -1)] <- dBVids[c(-nrow(dBV)), "freq"]
```

Adding column of the number neighbor blight violation

```r
## Adding collumn neighbor on coordID
nbviols_df <- plyr::count(dBV, vars = c("latR","lngR"))
mainData <- coordID
coordID$neighbor <- as.numeric(apply(as.matrix(coordID[,c("lat","lng")]), 1, neighbors
_total, df=nbviols_df), na.rm=FALSE)
```

Creating column that will store the number of crime violations listed above.

```r
## adding crime collumns on coordID
crimesBuild <- c("ARSON", "DAMAGE TO PROPERTY", "ENVIRONMENT","RUNAWAY")
otherCrimesRelated <- c("AGGRAVATED ASSAULT", "DRUNKENNESS", "EMBEZZLEMENT", "HOMICID
E","JUSTIFIABLE HOMICIDE","LARCENY","NEGLIGENT HOMICIDE","OTHER BURGLARY","OUIL DISPOS
E OF VEHICLE TO AVOID FORFEITURE","STOLEN VEHICLE","VAGRANCY (OTHER)", "ASSAULT","BURG
LARY","DANGEROUS DRUGS", "HEALTH-SAFETY", "IMMIGRATION", "KIDNAPING","LIQUOR", "WEAPON
S OFFENSES", "STOLEN PROPERTY", "ROBBERY")
dC1 <- plyr::count(dC[which(dC$category %in% crimesBuild),], vars = c("latR","lngR"))
coordID$crimesBuild <- as.numeric(apply(as.matrix(coordID[,c("lat","lng")]), 1, neighb
ors_total, df=dC1), na.rm=FALSE)
coordID$otherCrimesRelated <-  as.numeric(apply(as.matrix(coordID[,c("lat","lng")]),
1, neighbors_total, df=dC1), na.rm=FALSE)
```

Adding column for retain the number of 311 calls on the lat/long entire.

```r
# adding d311 calls types collumn on coordID
typeCall <- c("Traffic Sign Issue", "Traffic Signal Issue","Street Light Pole Down","T
est (internal use only, public issue)")
d311call <- plyr::count(d311[-which(d311$issue_type %in% typeCall),], vars = c("lat
R","lngR"))
coordID$ncalls <-  as.numeric(apply(as.matrix(coordID[,c("lat","lng")]), 1, neighbors_
total, df=d311call), na.rm=FALSE)
```

Finally, adding the value of the total amount of fees received by a lat/lon entrie.

```
## adding total violation fee collumn on coordID
dBV1 <- dBV[, c("latR","lngR","totalfee")]
names(dBV1) <- c("latR","lngR","freq")
coordID$fee <- as.numeric(apply(as.matrix(coordID[,c("lat","lng")]), 1, neighbors_tota
l, df=dBV1), na.rm=FALSE)
```

Regression and Prediction Models

This section is aimed on present the three regression and prediction models used to test the model created above.

Before demonstrate the regression and predictions models utilized, the creation of the test and train sets will be shown, below:

```
## creating train and test subsets
n <- nrow(coordID)
coordSample <- sample(n, size = n * 0.8)
train <- coordID[coordSample,]
test <- coordID[-coordSample,]

# Randomly shuffle the data
train <- train[sample(nrow(train)),]

# Create 5 equally size folds
fiveFolds <- cut(seq(1,nrow(train)),breaks=5,labels=FALSE)
```

The regression and prediction models used on this report are:

- Liner discriminant analysis using 5 fold

```
## LDA
error <- 1:5
for(i in 1:5){
    validIndex <- which(fiveFolds==i,arr.ind=TRUE)
    validData <- train[validIndex, ]
    trainData <- train[-validIndex, ]
    mod_fit <- lda(blight ~ log(1+nB) + log(1+neighbor)+ log(1 + crimesBuild) + log(1
+ otherCrimesRelated) + log(1+ ncalls) + log(1 + fee),data=trainData)
    mod_probs <- predict(mod_fit, validData)
    mod_preds <- mod_probs$class
    error[i] <- mean(mod_preds != validData$blight)
}
mean(error)
```

```
## [1] 0.02153113
```

- Generalized Binominal Linear Model using 5 fold.

```
## binomial
error <- 1:5
for(i in 1:5){
    validIndex <- which(fiveFolds==i,arr.ind=TRUE)
    validData <- train[validIndex, ]
    trainData <- train[-validIndex, ]
    mod_fit <- glm(blight ~ log(1+nB) + log(1+neighbor)+ log(1 + crimesBuild) + log(1
+ otherCrimesRelated) + log(1+ ncalls) + log(1 + fee), data=trainData, family = binomi
al)
    mod_probs <- predict(mod_fit, newdata = validData, type="response")
    mod_preds <- rep(FALSE, length(mod_probs))
    mod_preds[mod_probs > 0.5] <- TRUE
    error[i] <- mean(mod_preds != validData$blight)
}
mean(error)
```

```
## [1] 0.02152508
```

- Tree model

```
## Tree
ttrain <- train
ttrain$blight <- factor(ttrain$blight)
tree <- tree(blight ~ nB + neighbor + crimesBuild +otherCrimesRelated + ncalls + fee,
data = ttrain)
treePred <- predict(tree, test, type = "class")
mean(treePred != test$blight)
```

```
## [1] 0.02162764
```

Conclusion

So what can be concluded is that average error found on the prediction of blight building using this model is around 21,6%. Furthermore what can also be concluded is that this is not a precise model but can be used as reference for further development.