Final San Francisco Crime

Pedro Cornejo

02/14/2020

Introduccion.

The data provided about San Francisco city crime classification come from Kaggle's competition. The goal is to create a model able to predict crime in this city using the data between 2014 and 2015, which has 65534 observations and 9 features.–

During the process of designing the model, the maximum accuracy is achieved when data sets are filter over districts and high rated crimes. The performance is more accurate because the data is focalized in a specific area and crimes, which is known as bucketing data set to improve performance and accuracy in data set where classification is the goal.–

The hypothesis is that Crime is a function of time and its location. The data visualization and correlation analysis will verify that this hypothesis is supported, and then using the statistical models and tunning them look for the highest accuracy results.

```r
# Loading libraries needed in the analysis.

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project
```

```
## Loading required package: tidyverse

## -- Attaching packages ---------------------------------------------------------- tidyver

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ---------------------------------------------------------- tidyverse_con
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org"
if(!require(tidyr)) install.packages("tidyr", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Loading required package: lattice
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project
```

```
## Loading required package: lubridate

##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```r
if(!require(kableExtra)) install.packages("kableExtra", repos = "http://cran.us.r-proje
```

```
## Loading required package: kableExtra

##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```r
if(!require(gplots)) install.packages("gplots", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```r
if(!require(RColorBrewer)) install.packages("RColorBrewer", repos = "http://cran.us.r-p
```

```
## Loading required package: RColorBrewer
```

```r
if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.o
```

```
## Loading required package: corrplot

## corrplot 0.84 loaded
```

```r
if(!require(here)) install.packages("here", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: here
```

```
## here() starts at /Users/PedroCornejo/projects/Harvardx-data-science/Learning/SFLearni
```

```
##
## Attaching package: 'here'
```

```
## The following object is masked from 'package:lubridate':
##
##     here
```

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-proje
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(tidyverse)
library(caret)
library(lubridate)
library(kableExtra)
library(data.table)
library(e1071)
library(RColorBrewer)
library(gplots)
library(corrplot)
library(here)
```

Getting the data.

The data is dowloaded from: https://www.kaggle.com/c/sf-crime/download/RiTVAa9kf1hu9l7TmtUX%2Fversions%2FNRHocVFvjrC3Q7lrLMcf%2Ffiles%2Ftest.csv.zip, and https://www.kaggle.com/c/sf-crime/download/RiTVAa9kf1hu9l7TmtUX%2Fversions%2FNRHocVFvjrC3Q7lrLMcf%2Ffiles%2Ftrain.csv.zip. Kaggle website requires to login to load the data.–

The train and test sets are zipped, and they should be loaded in a folder named "data", which is a subfolder where the R.project is located. The function here() allows running the code in any OS and computer, openning the R.project file the file will pull the data in any system configuration without using path or system file functions. – Barrett,2019 explained that using set wd, path, and rm ls functions lack accuracy and create extra issues. The author states "If the first line of your R script is: setwd("c:....“). I will come into your office and SET YOUR COMPUTER ON FIRE. If the first line of your R script is: rm(list = ls()). I will come into your office and SET YOUR COMPUTER ON FIRE."–

The author explains that any project should load up on any computer, and it must "just work". Therefore, the Here function will help to load projects in any operating system and computer, just saving the data folder where the R.project file is located. This can be proved using the github: https://github.com/UNI1982, where this file project is saved, and runs in any computer, even in a laptop from work. It just works.

```
#################################
# Data Sets acquisition
#################################
 # San Francisco datasets:
# https://www.kaggle.com/c/sf-crime/download/RiTVAa9kf1hu9l7TmtUX%2Fversions%2FNRHocVF
# https://www.kaggle.com/c/sf-crime/download/RiTVAa9kf1hu9l7TmtUX%2Fversions%2FNRHocVF
# kaggle requires to login to download the files



# Getting data load requires to use of a data folder located in
# the same main folder were Rproject is located

here()
```

```
## [1] "/Users/PedroCornejo/projects/Harvardx-data-science/Learning/SFLearning"
```

```
getwd()
```

```
## [1] "/Users/PedroCornejo/projects/Harvardx-data-science/Learning/SFLearning"
```

```
test <- read.csv(here("data","SanFrancisco_test.csv"), stringsAsFactors = FALSE)
train<- read.csv(here("data","SanFrancisco_train.csv"),stringsAsFactors = FALSE)
str(test)
```

```
## 'data.frame':    65534 obs. of  7 variables:
##  $ Id        : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ Dates     : chr  "2015-05-10 23:59:00" "2015-05-10 23:51:00" "2015-05-10 23:50:00"
##  $ DayOfWeek : chr  "Sunday" "Sunday" "Sunday" "Sunday" ...
##  $ PdDistrict: chr  "BAYVIEW" "BAYVIEW" "NORTHERN" "INGLESIDE" ...
```

```
##  $ Address   : chr  "2000 Block of THOMAS AV" "3RD ST / REVERE AV" "2000 Block of GOU
##  $ X         : num  -122 -122 -122 -122 -122 ...
##  $ Y         : num  37.7 37.7 37.8 37.7 37.7 ...
```

```r
str(train)
```

```
## 'data.frame':    65534 obs. of  9 variables:
##  $ Dates     : chr  "2015-05-13 23:53:00" "2015-05-13 23:53:00" "2015-05-13 23:33:00"
##  $ Category  : chr  "WARRANTS" "OTHER OFFENSES" "OTHER OFFENSES" "LARCENY/THEFT" ...
##  $ Descript  : chr  "WARRANT ARREST" "TRAFFIC VIOLATION ARREST" "TRAFFIC VIOLATION AR
##  $ DayOfWeek : chr  "Wednesday" "Wednesday" "Wednesday" "Wednesday" ...
##  $ PdDistrict: chr  "NORTHERN" "NORTHERN" "NORTHERN" "NORTHERN" ...
##  $ Resolution: chr  "ARREST, BOOKED" "ARREST, BOOKED" "ARREST, BOOKED" "NONE" ...
##  $ Address   : chr  "OAK ST / LAGUNA ST" "OAK ST / LAGUNA ST" "VANNESS AV / GREENWICH
##  $ X         : num  -122 -122 -122 -122 -122 ...
##  $ Y         : num  37.8 37.8 37.8 37.8 37.8 ...
```

```r
dim(train)
```

```
## [1] 65534     9
```

```r
dim(test)
```

```
## [1] 65534     7
```

Train and Test Set up.

The str function shows two data sets with different dimensions. The test-set has only one possible factor to analyze, the PdDistrict, with the predictors' time and location features. On the other hand, the train set has the factors Category, the PdDistrict, which are the type of crimes and the district of occurrence, and the predictors' time and location respectively. The provided train-set gives a better opportunity to create a model that predicts the type of crime, the category column, related to PdDistrict, time, and location in San Francisco city.–

There are two solutions, one is to sample and split the train set to get the training and testing data for the model. The second option is to join the provided train and test datasets, and then, create the training, testing and validate data for the model will be created. The latter will be the approach in this analysis to have bigger data sets. To join the data sets, first, it is necessary to ensure that both train and test sets are not sharing data to avoid repetitive or missing data, and then create the training and testing sets for the project.

```r
# str and im functions help to know and check the data loaded to have
# an idea # how to set it up for the future model.
# Spliting the train set into subsets for cross validation of the model
#  Checking if both datasets are unique
main_data <- full_join(train,test, by = NULL, type = "full", match = "all")
```

```
## Joining, by = c("Dates", "DayOfWeek", "PdDistrict", "Address", "X", "Y")
```

```r
dim(main_data)
```

```
## [1] 131068     10
```

```r
# Successfully jointed, the dimension of the main_data is the  sum of
# dim(train) + dim(test)

# Spliting the train set into subsets for cross validation of the model
# Set.seed(1) for R version before the 3.6 version
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
sampling_Trainset <- createDataPartition(y = main_data$Category,
                                         p = 0.20, list = FALSE)
```

```
## Warning in createDataPartition(y = main_data$Category, p = 0.2, list = FALSE):
## Some classes have a single record ( TREA ) and these will be selected for the
## sample
```

```r
# creating the data sets shows TREA , which is denominated as Traspassing in the
# FBI crime definition. This is a unique data that will no affect our analysis,
#and it is included.
training <- main_data[-sampling_Trainset, ]
testing <- main_data[sampling_Trainset,]
summary(training)
```

```
##     Dates             Category           Descript          DayOfWeek
##  Length:104839      Length:104839      Length:104839      Length:104839
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   PdDistrict         Resolution          Address                 X
##  Length:104839      Length:104839      Length:104839      Min.   :-122.5
##  Class :character   Class :character   Class :character   1st Qu.:-122.4
##  Mode  :character   Mode  :character   Mode  :character   Median :-122.4
##                                                           Mean   :-122.4
##                                                           3rd Qu.:-122.4
##                                                           Max.   :-122.4
##
##        Y               Id
##  Min.   :37.71   Min.   :    1
##  1st Qu.:37.76   1st Qu.:16442
```

```
##  Median :37.78   Median :32804
##  Mean   :37.77   Mean   :32789
##  3rd Qu.:37.79   3rd Qu.:49185
##  Max.   :37.82   Max.   :65533
##                  NA's   :52412
```

```r
# The Id column has NAs that weres added in the Join process , we do not need them

training$Id <- NULL
testing$Id <- NULL
```

Cleaning up the data.

The category column is a character variable that requires to be a factor to be part of the model. Model prediction predicts a factor depending on the numerical features, predictors. Therefore, the category column requires to be a factor, the predictors need to numerical.

```r
# Category as a factor to be able to validate the model

training$Category <-  as.factor(training$Category)
# Factors to be predicted
levels(training$Category)
```

```
##  [1] "ARSON"                       "ASSAULT"
##  [3] "BAD CHECKS"                  "BRIBERY"
##  [5] "BURGLARY"                    "DISORDERLY CONDUCT"
##  [7] "DRIVING UNDER THE INFLUENCE" "DRUG/NARCOTIC"
##  [9] "DRUNKENNESS"                 "EMBEZZLEMENT"
## [11] "EXTORTION"                   "FAMILY OFFENSES"
## [13] "FORGERY/COUNTERFEITING"      "FRAUD"
## [15] "GAMBLING"                    "KIDNAPPING"
## [17] "LARCENY/THEFT"               "LIQUOR LAWS"
## [19] "LOITERING"                   "MISSING PERSON"
## [21] "NON-CRIMINAL"                "OTHER OFFENSES"
## [23] "PROSTITUTION"                "ROBBERY"
## [25] "RUNAWAY"                     "SECONDARY CODES"
## [27] "SEX OFFENSES FORCIBLE"       "SEX OFFENSES NON FORCIBLE"
## [29] "STOLEN PROPERTY"             "SUICIDE"
## [31] "SUSPICIOUS OCC"              "TRESPASS"
## [33] "VANDALISM"                   "VEHICLE THEFT"
## [35] "WARRANTS"                    "WEAPON LAWS"
```

```r
# Checking any NA on data sets
anyNA(training)
```

```
## [1] TRUE
```

```r
# There are 52427 NAs in Category, meaning there is no records on type of crime,
# so these data are not providing any information, so they can be eliminated

training <- training%>%filter(Category !="NA.")%>% droplevels()
levels(training$Category)
```

```
##  [1] "ARSON"                       "ASSAULT"
##  [3] "BAD CHECKS"                  "BRIBERY"
##  [5] "BURGLARY"                    "DISORDERLY CONDUCT"
##  [7] "DRIVING UNDER THE INFLUENCE" "DRUG/NARCOTIC"
##  [9] "DRUNKENNESS"                 "EMBEZZLEMENT"
## [11] "EXTORTION"                   "FAMILY OFFENSES"
## [13] "FORGERY/COUNTERFEITING"      "FRAUD"
## [15] "GAMBLING"                    "KIDNAPPING"
## [17] "LARCENY/THEFT"               "LIQUOR LAWS"
## [19] "LOITERING"                   "MISSING PERSON"
## [21] "NON-CRIMINAL"                "OTHER OFFENSES"
## [23] "PROSTITUTION"                "ROBBERY"
## [25] "RUNAWAY"                     "SECONDARY CODES"
## [27] "SEX OFFENSES FORCIBLE"       "SEX OFFENSES NON FORCIBLE"
## [29] "STOLEN PROPERTY"             "SUICIDE"
## [31] "SUSPICIOUS OCC"              "TRESPASS"
## [33] "VANDALISM"                   "VEHICLE THEFT"
## [35] "WARRANTS"                    "WEAPON LAWS"
```

```r
anyNA(training)
```

```
## [1] FALSE
```

```r
summary(training)
```

```
##     Dates                          Category        Descript
##  Length:52412       LARCENY/THEFT :14352   Length:52412
##  Class :character   OTHER OFFENSES: 6820   Class :character
##  Mode  :character   NON-CRIMINAL  : 6560   Mode  :character
##                     ASSAULT       : 4475
##                     VEHICLE THEFT : 2602
##                     VANDALISM     : 2519
##                     (Other)       :15084
##   DayOfWeek          PdDistrict         Resolution          Address
##  Length:52412       Length:52412       Length:52412       Length:52412
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
```

```
##
##        X                 Y
##  Min.   :-122.5   Min.   :37.71
##  1st Qu.:-122.4   1st Qu.:37.76
##  Median :-122.4   Median :37.78
##  Mean   :-122.4   Mean   :37.77
##  3rd Qu.:-122.4   3rd Qu.:37.79
##  Max.   :-122.4   Max.   :37.82
##
```

```r
# Now training is cleaned

# Cleaning testing data set
testing$Category <-  as.factor(testing$Category)
# Factors to be predicted
levels(testing$Category)
```

```
##  [1] "ARSON"                       "ASSAULT"
##  [3] "BAD CHECKS"                  "BRIBERY"
##  [5] "BURGLARY"                    "DISORDERLY CONDUCT"
##  [7] "DRIVING UNDER THE INFLUENCE" "DRUG/NARCOTIC"
##  [9] "DRUNKENNESS"                 "EMBEZZLEMENT"
## [11] "EXTORTION"                   "FAMILY OFFENSES"
## [13] "FORGERY/COUNTERFEITING"      "FRAUD"
## [15] "GAMBLING"                    "KIDNAPPING"
## [17] "LARCENY/THEFT"               "LIQUOR LAWS"
## [19] "LOITERING"                   "MISSING PERSON"
## [21] "NON-CRIMINAL"                "OTHER OFFENSES"
## [23] "PROSTITUTION"                "ROBBERY"
## [25] "RUNAWAY"                     "SECONDARY CODES"
## [27] "SEX OFFENSES FORCIBLE"       "SEX OFFENSES NON FORCIBLE"
## [29] "STOLEN PROPERTY"             "SUICIDE"
## [31] "SUSPICIOUS OCC"              "TREA"
## [33] "TRESPASS"                    "VANDALISM"
## [35] "VEHICLE THEFT"               "WARRANTS"
## [37] "WEAPON LAWS"
```

```r
# Checking any NA on data sets
anyNA(testing)
```

```
## [1] TRUE
```

```r
# There are 52427 NAs in Category, meaning there is no records on type of crime,
# so these data are not providing any information, so they can be eliminated

testing <- testing%>%filter(Category !="NA.")%>% droplevels()
levels(testing$Category)
```

```
##  [1] "ARSON"                       "ASSAULT"
##  [3] "BAD CHECKS"                  "BRIBERY"
##  [5] "BURGLARY"                    "DISORDERLY CONDUCT"
##  [7] "DRIVING UNDER THE INFLUENCE" "DRUG/NARCOTIC"
##  [9] "DRUNKENNESS"                 "EMBEZZLEMENT"
## [11] "EXTORTION"                   "FAMILY OFFENSES"
## [13] "FORGERY/COUNTERFEITING"      "FRAUD"
## [15] "GAMBLING"                    "KIDNAPPING"
## [17] "LARCENY/THEFT"               "LIQUOR LAWS"
## [19] "LOITERING"                   "MISSING PERSON"
## [21] "NON-CRIMINAL"                "OTHER OFFENSES"
## [23] "PROSTITUTION"                "ROBBERY"
## [25] "RUNAWAY"                     "SECONDARY CODES"
## [27] "SEX OFFENSES FORCIBLE"       "SEX OFFENSES NON FORCIBLE"
## [29] "STOLEN PROPERTY"             "SUICIDE"
## [31] "SUSPICIOUS OCC"              "TREA"
## [33] "TRESPASS"                    "VANDALISM"
## [35] "VEHICLE THEFT"               "WARRANTS"
## [37] "WEAPON LAWS"
```

```r
anyNA(testing)
```

```
## [1] FALSE
```

```r
summary(testing)
```

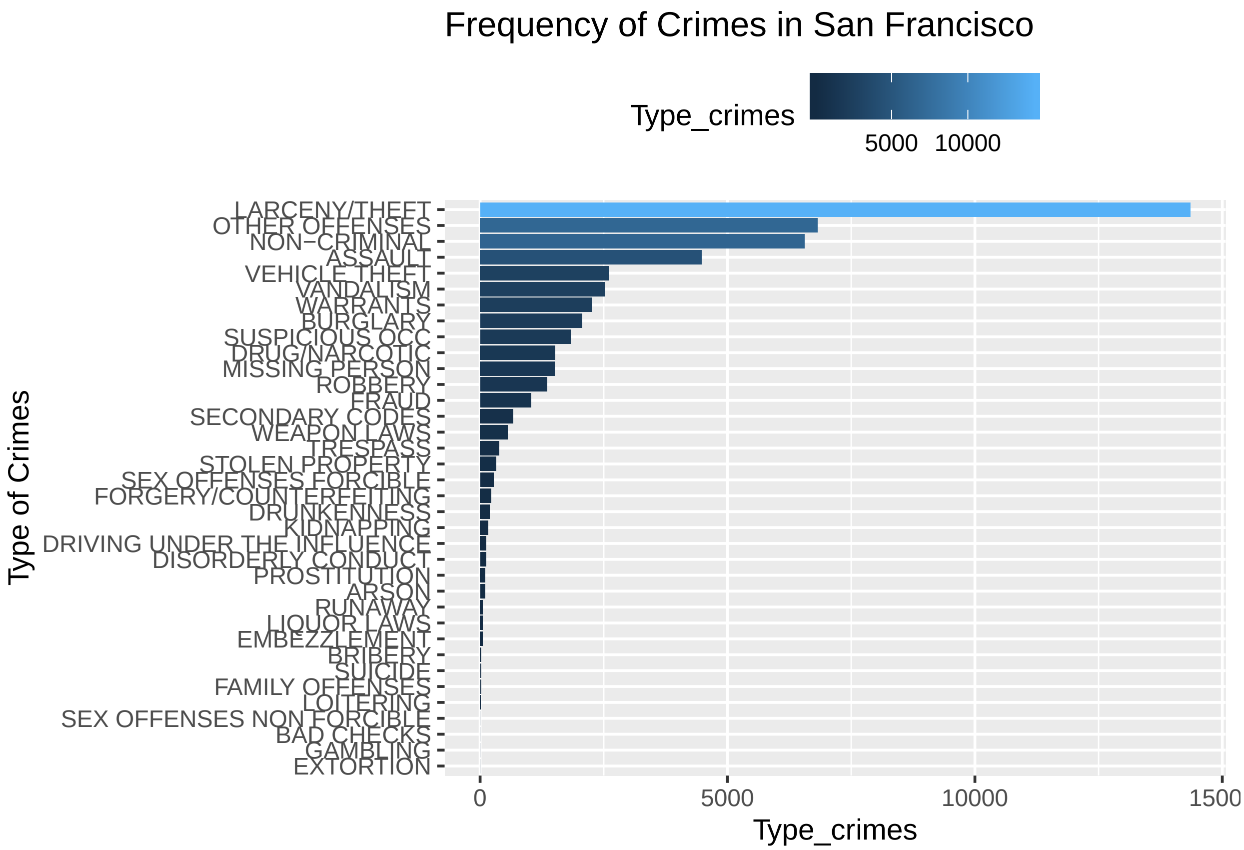
```
##     Dates                        Category       Descript          DayOfWeek
##  Length:13122       LARCENY/THEFT :3589   Length:13122       Length:13122
##  Class :character   OTHER OFFENSES:1705   Class :character   Class :character
##  Mode  :character   NON-CRIMINAL  :1640   Mode  :character   Mode  :character
##                     ASSAULT       :1119
##                     VEHICLE THEFT : 651
##                     VANDALISM     : 630
##                     (Other)       :3788
##   PdDistrict         Resolution          Address                  X
##  Length:13122       Length:13122       Length:13122       Min.   :-122.5
##  Class :character   Class :character   Class :character   1st Qu.:-122.4
##  Mode  :character   Mode  :character   Mode  :character   Median :-122.4
##                                                           Mean   :-122.4
##                                                           3rd Qu.:-122.4
##                                                           Max.   :-122.4
##
##        Y
##  Min.   :37.71
##  1st Qu.:37.76
```

```
##  Median :37.78
##  Mean   :37.77
##  3rd Qu.:37.79
##  Max.   :37.81
##
```

```
# Data sets are cleaned and ready to be analysed
```

**** Distribution of type Crime. On the graphic, the five more relevant of the 36 crimes are: are Larceny, other offenses, non-criminal, assault, and vehicle theft**. There are crimes in the very low ranges that perhaps are not influential in any district. The heatmaps will clarify if only high crimes on all districts have significant influence over the districts and locations.**

```
############ Data Visualization#######################


Crimes_freq <- training %>% group_by(Category) %>%
               summarise(Type_crimes = n()) %>%
               arrange(desc(Type_crimes)) %>% ungroup()

# Using http://r-statistics.co/ggplot2-Tutorial-With-R.html as plotting guide:

Crimes_freq%>% ggplot(aes(reorder(Category,Type_crimes), y = Type_crimes,
                          fill =Type_crimes)) + geom_col() +
              coord_flip() +theme(legend.position = "top") +
              labs( x = 'Type of Crimes',
                    title = 'Frequency of Crimes in San Francisco')
```




```r
nrow(Crimes_freq)
```

```
## [1] 36
```

```r
# 36 crimes that our graphic shows larceny, Assault , Vehiclue theft as the more
# frequent type of crimea in all the districts.
```

**** Setting up databases**. Now, it is required to transform our predictors into numerical variables. The time and location are not simple variables. Time is a cyclical variable that changes every 24hour, 12 months, and 365 days.– The location is expressed on degrees because the data are expressed in spherical coordinates that it is used to calculate the longitude and latitude of any position on the planet. Therefore, both variables Location and time require a transformation to normalize and ensure the same scale.**

Location. R multipurpose kit,2015 gives the relationship between latitude and longitude to cartesian coordinates, which is the numerical distance at any point of the planet. "x = radius * cos(latitude) * cos(longitude) y = radius * cos(latitude) * sin(longitude) z = radius * sin(latitude)" The normalization of the varibles x,y,z, the vector(x,y,z),is dividing by the radius. The main goal of this tranformation is to normalized the data to ensure the same scale distance variables.

```
#LOCATION:
# Longitud lattidue are in spherical coordinates, which have degrees as scale.
# Therefore, those variable need to be in cartesian coordenates to avoid degrees,
# and standarized the data sets.

training$Location_x <- cos(training$Y)*cos(training$X)
training$Location_y <- cos(training$Y)* sin(training$X)
training$Location_z <- sin(training$Y)
```

Time Abramson,2019 presents an easy way to understand cyclical variables. This cycle is a unitary circle from 0 to 360, the period depends on the variable to be analyzed. For instance, in one month the cycle is every 12 month and a new cycle happens.–

Cycical concepts are important because the distance between Januaries of different years is zero, computers are not able to know the difference. Therefore, it is necessary to tell the computer that a new cycle happens. Abramson established that any point in the circle, which is normalized because it is a unitary circle, is represented by the sin and cos.–

Therefore, the W position in the circle is:(sin(2piW/f), cos(2piW/f)). The "f" it is the frequency or the new cycle. For a month, it will be sin(2piMonth/12) and its os(2pi*Month/12). This concept is applied for the day, month and year on the training and testing datasets.

```
# TIME

# The dates as POSIXct and characters as factors.
training$Dates <- ymd_hms(training$Dates)
training$Years <- year(training$Dates)
training$Months <- month(training$Dates)
training$Days <- day(training$Dates)
training$Hours <- hour(training$Dates)
training$PdDistrict <- as.factor(training$PdDistrict)
training$DayOfWeek <- wday(training$Dates)

# Time data is cyclical, so it is more accurate to use radial time
# approach to set up time data
training$Years_sin <- sin(2*pi*training$Years/365)
training$Years_cos <- cos(2*pi*training$Years/365)
# For cyclical months
training$Months_sin <- sin(2*pi*training$Months/12)
training$Months_cos <- cos(2*pi*training$Months/12)

# For cyclical hours and days
training$Days_sin <-  sin(2*pi*training$Days/30)
training$Days_cos <- cos(2*pi*training$Days/30)
training$Hours_sin <- sin(2*pi*training$Hours/24)
```

```r
training$Hours_cos <- cos(2*pi*training$Hours/24)


# The test-set must have in same paramenters to have a correct validation.

testing$Dates <- ymd_hms(testing$Dates)
testing$Years <- year(testing$Dates)
testing$Months <- month(testing$Dates)
testing$Days <- day(testing$Dates)
testing$Hours <- hour(testing$Dates)
testing$PdDistrict <- as.factor(testing$PdDistrict)
testing$DayOfWeek <- wday(testing$Dates)
testing$Years_sin <- sin(2*pi*testing$Years/365)
testing$Years_cos <- cos(2*pi*testing$Years/365)
testing$Months_sin <- sin(2*pi*testing$Months/12)
testing$Months_cos <- cos(2*pi*testing$Months/12)
testing$Days_sin <-  sin(2*pi*testing$Days/30)
testing$Days_cos <- cos(2*pi*testing$Days/30)
testing$Hours_sin <- sin(2*pi*testing$Hours/24)
testing$Hours_cos <- cos(2*pi*testing$Hours/24)
testing$Location_x <- cos(testing$Y)*cos(testing$X)
testing$Location_y <- cos(testing$Y)* sin(testing$X)
testing$Location_z <- sin(testing$Y)


# ckeking if both data sets have the same columns

ifelse(all(sort(names(training)) %in% sort(names(testing))),
       "Identical data sets", "No ready")
```

```
## [1] "Identical data sets"
```

```r
ifelse(all(sapply(training, class) %in% sapply(testing, class)),
       "Same Classes", "No ready")
```

```
## [1] "Same Classes"
```

```r
# The datasets can be leaner. Address is not required because longitude
# and latidue are provided.
# descript and resolution are not part of the hypothesis.
# Elimating data that were transformed

training[c("Descript","Resolution", "Address","DayOfWeek","X", "Y",
           "Years","Days","Hours")] <- list(NULL)
testing[c("Descript","Resolution", "Address","DayOfWeek","X", "Y",
          "Years","Days","Hours")] <- list(NULL)
```

```r
# Ensuring no NAs and normalization
summary(training)
```

```
##       Dates                                         Category          PdDistrict
##  Min.   :2014-06-29 18:20:00   LARCENY/THEFT :14352   SOUTHERN :10123
##  1st Qu.:2014-09-18 19:00:00   OTHER OFFENSES: 6820   NORTHERN : 6602
##  Median :2014-12-08 14:00:00   NON-CRIMINAL  : 6560   CENTRAL  : 6458
##  Mean   :2014-12-06 07:47:44   ASSAULT       : 4475   MISSION  : 6335
##  3rd Qu.:2015-02-21 13:33:30   VEHICLE THEFT : 2602   BAYVIEW  : 4694
##  Max.   :2015-05-13 23:53:00   VANDALISM     : 2519   INGLESIDE: 4567
##                                (Other)       :15084   (Other)  :13633
##    Location_x        Location_y          Location_z            Months
##  Min.   :-0.9996   Min.   :-0.155270   Min.   :0.008919   Min.   : 1.000
##  1st Qu.:-0.9939   1st Qu.:-0.115058   1st Qu.:0.056203   1st Qu.: 3.000
##  Median :-0.9915   Median :-0.105196   Median :0.076235   Median : 7.000
##  Mean   :-0.9921   Mean   :-0.098576   Mean   :0.069546   Mean   : 6.664
##  3rd Qu.:-0.9901   3rd Qu.:-0.088162   3rd Qu.:0.086001   3rd Qu.:10.000
##  Max.   :-0.9817   Max.   :-0.008445   Max.   :0.120518   Max.   :12.000
##
##    Years_sin         Years_cos          Months_sin          Months_cos
##  Min.   :-0.1287   Min.   :-0.9937   Min.   :-1.00000   Min.   :-1.000
##  1st Qu.:-0.1287   1st Qu.:-0.9937   1st Qu.:-0.86603   1st Qu.:-0.500
##  Median :-0.1117   Median :-0.9937   Median : 0.00000   Median : 0.000
##  Mean   :-0.1188   Mean   :-0.9929   Mean   :-0.04104   Mean   : 0.143
##  3rd Qu.:-0.1117   3rd Qu.:-0.9917   3rd Qu.: 0.86603   3rd Qu.: 0.866
##  Max.   :-0.1117   Max.   :-0.9917   Max.   : 1.00000   Max.   : 1.000
##
##     Days_sin          Days_cos            Hours_sin          Hours_cos
##  Min.   :-0.9945   Min.   :-1.000000   Min.   :-1.0000   Min.   :-1.00000
##  1st Qu.:-0.7431   1st Qu.:-0.669131   1st Qu.:-0.8660   1st Qu.:-0.70711
##  Median : 0.0000   Median :-0.104528   Median :-0.5000   Median :-0.25882
##  Mean   : 0.0280   Mean   : 0.003002   Mean   :-0.2578   Mean   :-0.06215
##  3rd Qu.: 0.7431   3rd Qu.: 0.669131   3rd Qu.: 0.2588   3rd Qu.: 0.70711
##  Max.   : 0.9945   Max.   : 1.000000   Max.   : 1.0000   Max.   : 1.00000
##
```

```r
summary(testing)
```

```
##       Dates                                         Category         PdDistrict
##  Min.   :2014-06-29 18:28:00   LARCENY/THEFT :3589   SOUTHERN :2580
##  1st Qu.:2014-09-17 16:05:30   OTHER OFFENSES:1705   NORTHERN :1668
##  Median :2014-12-08 20:24:30   NON-CRIMINAL  :1640   CENTRAL  :1619
##  Mean   :2014-12-05 22:11:54   ASSAULT       :1119   MISSION  :1587
##  3rd Qu.:2015-02-21 14:33:45   VEHICLE THEFT : 651   BAYVIEW  :1170
##  Max.   :2015-05-13 23:53:00   VANDALISM     : 630   INGLESIDE:1169
```

```
##                                (Other)       :3788   (Other)  :3329
##      Months         Years_sin         Years_cos         Months_sin
##  Min.   : 1.000   Min.   :-0.1287   Min.   :-0.9937   Min.   :-1.00000
##  1st Qu.: 3.000   1st Qu.:-0.1287   1st Qu.:-0.9937   1st Qu.:-0.86603
##  Median : 7.000   Median :-0.1117   Median :-0.9937   Median : 0.00000
##  Mean   : 6.595   Mean   :-0.1189   Mean   :-0.9929   Mean   :-0.03565
##  3rd Qu.:10.000   3rd Qu.:-0.1117   3rd Qu.:-0.9917   3rd Qu.: 0.86603
##  Max.   :12.000   Max.   :-0.1117   Max.   :-0.9917   Max.   : 1.00000
##
##    Months_cos         Days_sin           Days_cos            Hours_sin
##  Min.   :-1.0000   Min.   :-0.99452   Min.   :-1.000000   Min.   :-1.0000
##  1st Qu.:-0.5000   1st Qu.:-0.74314   1st Qu.:-0.669131   1st Qu.:-0.8660
##  Median : 0.0000   Median : 0.00000   Median :-0.104528   Median :-0.5000
##  Mean   : 0.1309   Mean   : 0.02185   Mean   : 0.006718   Mean   :-0.2687
##  3rd Qu.: 0.8660   3rd Qu.: 0.74314   3rd Qu.: 0.669131   3rd Qu.: 0.2588
##  Max.   : 1.0000   Max.   : 0.99452   Max.   : 1.000000   Max.   : 1.0000
##
##    Hours_cos           Location_x        Location_y           Location_z
##  Min.   :-1.00000   Min.   :-0.9996   Min.   :-0.154958   Min.   :0.009042
##  1st Qu.:-0.70711   1st Qu.:-0.9938   1st Qu.:-0.115058   1st Qu.:0.056828
##  Median :-0.25882   Median :-0.9915   Median :-0.104981   Median :0.076235
##  Mean   :-0.06429   Mean   :-0.9921   Mean   :-0.098665   Mean   :0.069609
##  3rd Qu.: 0.70711   3rd Qu.:-0.9901   3rd Qu.:-0.088210   3rd Qu.:0.086050
##  Max.   : 1.00000   Max.   :-0.9817   Max.   :-0.008445   Max.   :0.110334
##
```

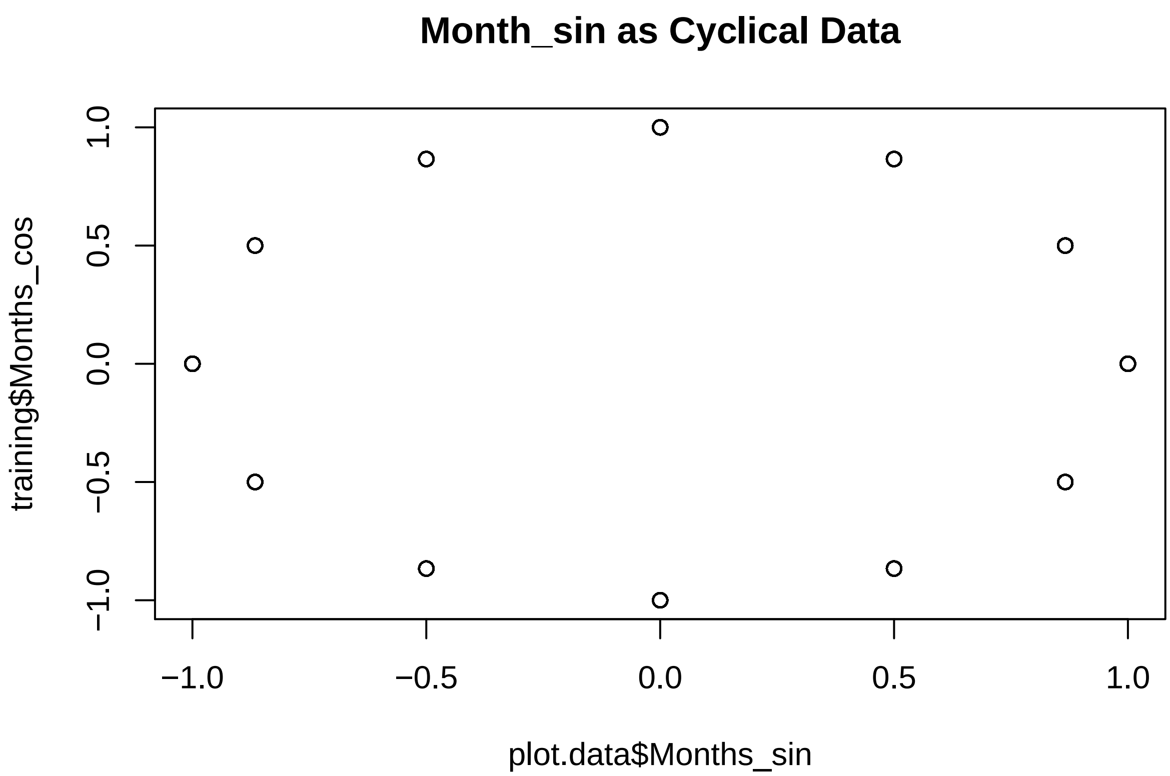
```r
# The datasets are normalized and ready to be used because
# The minimum and maximum do not have a big gaps.
```

Data Visualization to ensure normalization

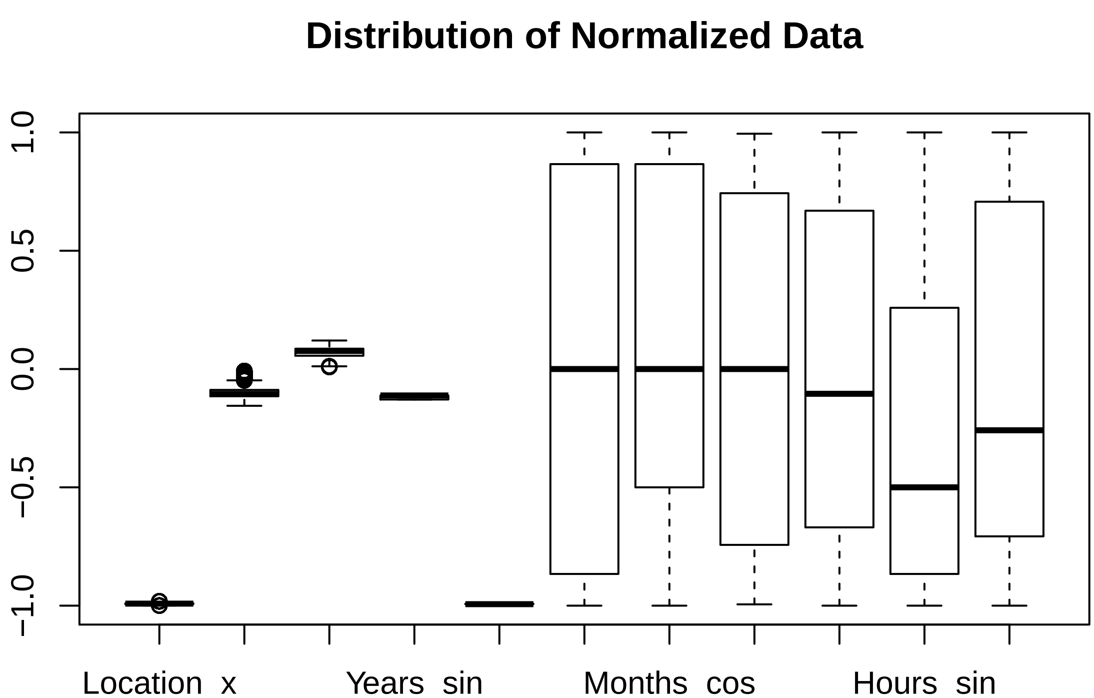
The plot and boxplot show the cyclical variable behavior and the numerical normalized data set, respectively. There are not outliers and the cyclical variables can tell the model when new data starts a new cycle.–

The plot() shows a circle for the month because this is a cyclical data. The boxplot shows the data range and behaviors of the data. The graphics ensure that our data sets are normalized and ready to be used.

```r
#### CHECKING NORMALIZATION ######

# Checking if the data for time is cyclical
plot.data <- training
plot(plot.data$Months_sin,training$Months_cos, main="Month_sin as Cyclical Data")
```




```r
# Checking data normalization
boxplot(plot.data[ , c(4:6,8:15)], main="Distribution of Normalized Data")
```




```r
# Graphics show a range of -1.0 to 1.0. Data is normalized and ready to be used.
```

Correlation of crimes per district

First checking the graphic of correlation, it is noticeable that the color blue is all over the graphic, which is according to the range a high 0.73 value. Therefore, the hypothesis of crimes related to the districts and time is realistically measurable. The correlation with the other variables is similar. It is not present in this paper to not exaggerate the length of the presentation.

```
###### Creating Correlation and Heatmaps of Crimes per Dictrict #######

# Guide gplots heatmap.2() features:  page 26 and 31 of #https://cran.r-project.org/we

CategoryPdDistrict_data <- training %>%
                           group_by(Category, PdDistrict)%>%
                           summarise(District_crimes = n())
CaPD <- CategoryPdDistrict_data %>%
        group_by_at(vars(-District_crimes)) %>%
        mutate(row_id=1:n()) %>% ungroup() %>%
        spread(key=PdDistrict, value=District_crimes) %>%
        select(-row_id)
head(CaPD)
```

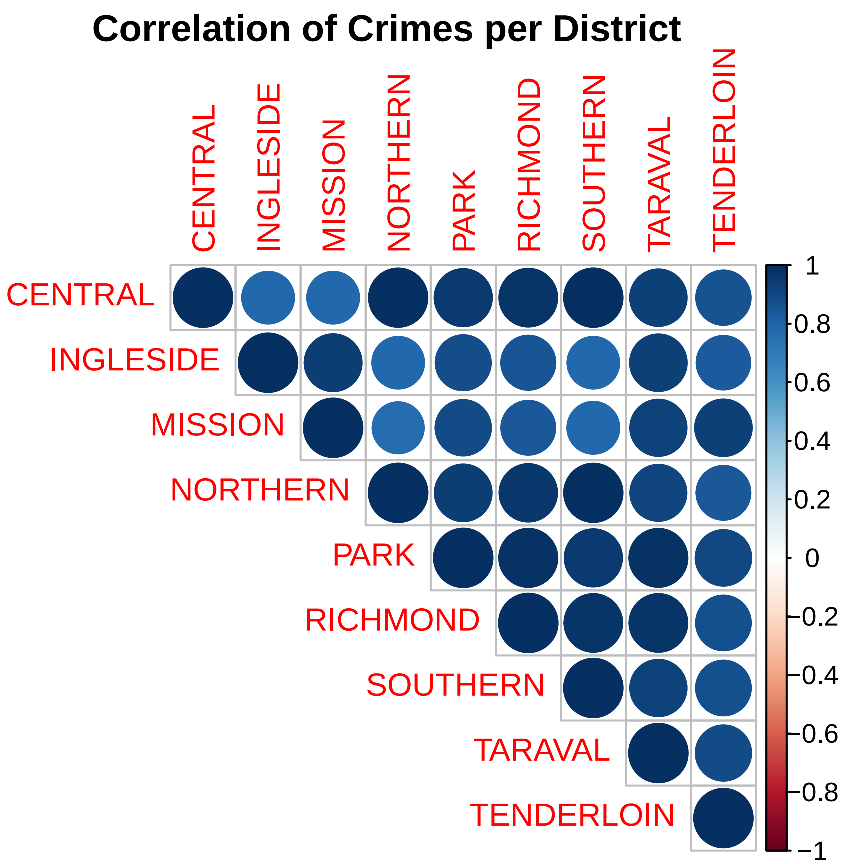
```
## # A tibble: 6 x 11
##   Category BAYVIEW CENTRAL INGLESIDE MISSION NORTHERN  PARK RICHMOND SOUTHERN
##   <fct>      <int>   <int>     <int>   <int>    <int> <int>    <int>    <int>
## 1 ARSON         20      11         6      14       16     3        7       13
## 2 ASSAULT      549     476       487     681      518   182      147      709
## 3 BAD CHE~       1       2        NA       2        1    NA        1        1
## 4 BRIBERY        6      NA         3       7        3    NA       NA        1
## 5 BURGLARY     179     272       213     240      288   144      122      302
## 6 DISORDE~       9      18         3      27       12     8        3       25
## # ... with 2 more variables: TARAVAL <int>, TENDERLOIN <int>
```

```
# There are NAs over the districts. These NAs is because is very low crime or
# no data were collected. So, it necessary to give a zero value instead
CaPD[is.na(CaPD)] <- 0

# CaPD needs to be a data frame to avoid the warning Tibble depreceated
CaPD <- as.data.frame(CaPD)
# Preparing the matrix
row.names(CaPD) <- CaPD$Category

Matrix1_CaPd <-data.matrix(CaPD[,-1])
Matrix_CaPd <- Matrix1_CaPd[,-1]
head(Matrix_CaPd)
```

```
##                      CENTRAL INGLESIDE MISSION NORTHERN PARK RICHMOND SOUTHERN
```

```
## ARSON                         11          6       14        16     3        7        13
## ASSAULT                      476        487      681       518   182      147       709
## BAD CHECKS                     2          0        2         1     0        1         1
## BRIBERY                        0          3        7         3     0        0         1
## BURGLARY                     272        213      240       288   144      122       302
## DISORDERLY CONDUCT            18          3       27        12     8        3        25
##                     TARAVAL TENDERLOIN
## ARSON                     7          4
## ASSAULT                 308        418
## BAD CHECKS                0          0
## BRIBERY                   1          1
## BURGLARY                222         78
## DISORDERLY CONDUCT        4         12
```

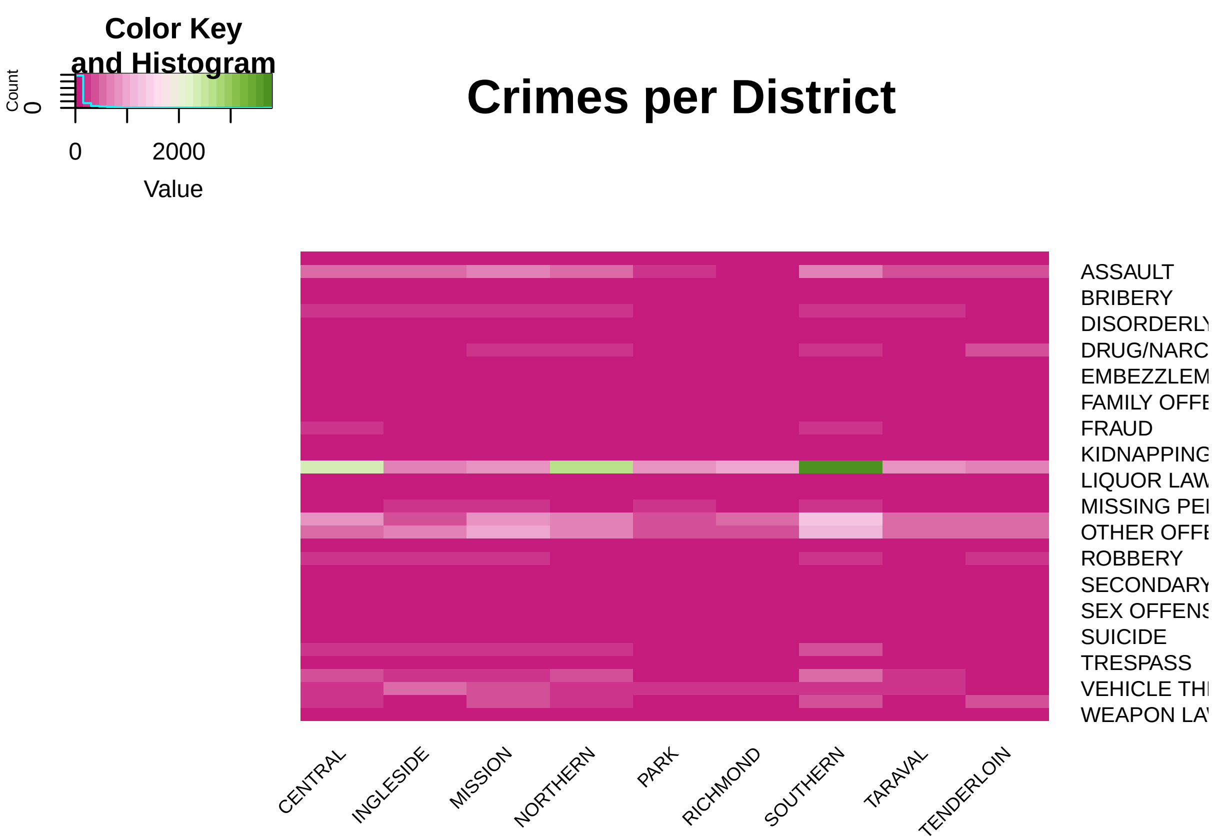
```r
# Checking the correlation between variables to ensure that
# the hypothesis is on the correct direction.
m_cor <- cor(Matrix_CaPd)
corrplot(m_cor, type = "upper",mar=c(0,0,1,0),
         main="Correlation of Crimes per District")
```




```r
# The minimum correlation is 0.73, which corroborate how  the Type
# of crime depend on the district.
```

**** Heatmaps Crime per district**. The heatmap shows a prominent increase of Larceny/theft and Assault as the main crimes over the districts and time. The most dangerous is the Southern district and the least dangerous Richmond and Park districts, which shows a unique pink color overall. The heatmaps are telling us that the hypothesis is in the correct direction because there is a high dependency on crimes on the Districts.**

```r
# The heat will tell the district with high and low crime
coul2 <- colorRampPalette(brewer.pal(8, "PiYG"))(25)
CrimeDistrict_heatmap3 <- heatmap.2(Matrix_CaPd,
                          Colv=FALSE,
                          srtCol=45,
                          Rowv=FALSE,
                          dendrogram="none",
                          density.info="histogram",
                          trace="none",
                          col = coul2,
                          cexRow=0.85,cexCol=0.75,
                          main = " Crimes per District")
```



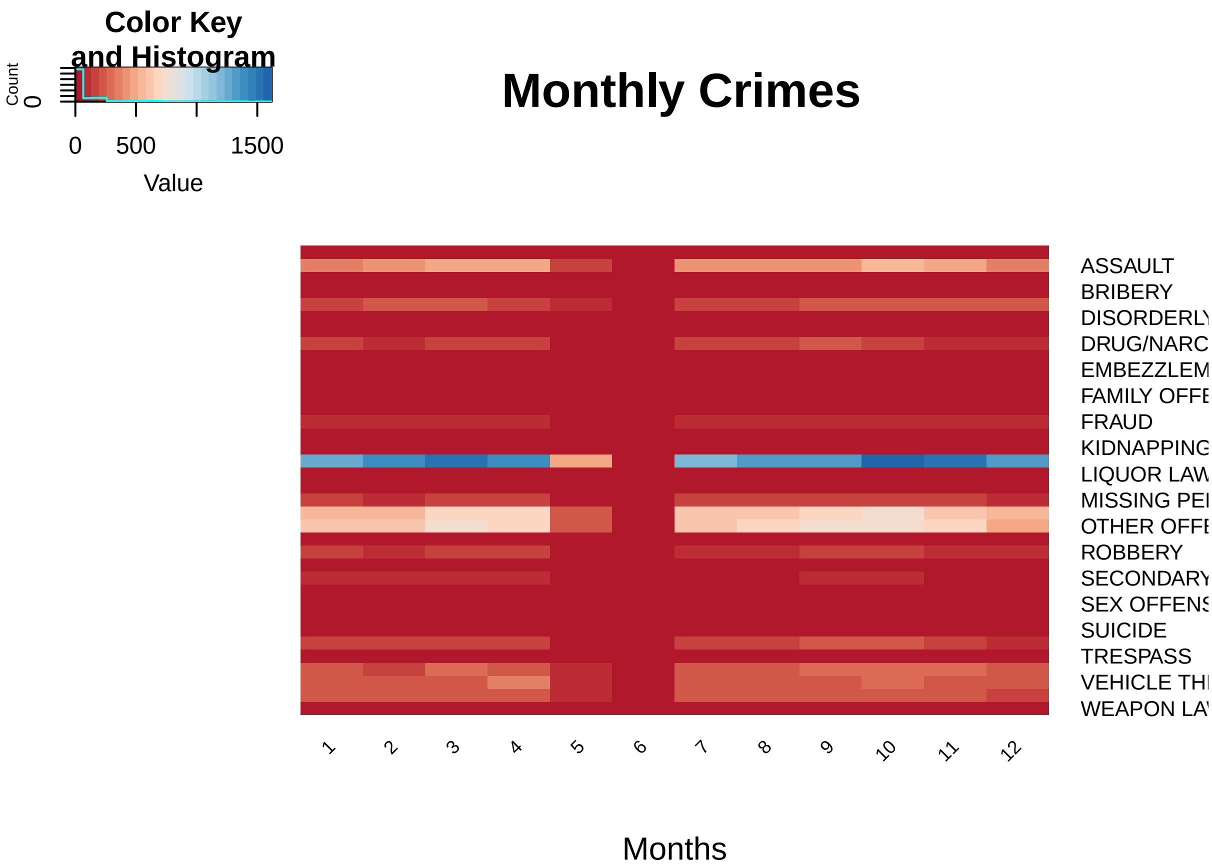
** Heatmaps Type of Crimes Monthly**. This heatmap shows how crime increase in

certain months. Some high violent crimes are high during the whole year, but at the end of the year crimes increase substantially.

```r
##### ** Heatmaps Type of Crimes Monthly ################
# Now correlation of category with Time. Months was choosen for
# simplicity, but it coul be Hours, Years,

CategoryTime_data <- training %>% group_by(Category, Months)%>%
                     summarise(Monthly_crimes = n())

CaMonths <- CategoryTime_data %>% group_by_at(vars(-Monthly_crimes)) %>%
            mutate(row_id=1:n()) %>% ungroup() %>%
            spread(key=Months, value=Monthly_crimes) %>%
            select(-row_id)
CaMonths[is.na(CaMonths)] <- 0
CaMonths <- as.data.frame(CaMonths)
row.names(CaMonths) <- CaMonths$Category
Matrix1_CaMonths <- data.matrix(CaMonths)
Matrix_CaMonths <- Matrix1_CaMonths[,-1]
head(Matrix_CaMonths)
```

```
##                      1   2   3   4   5  6   7   8   9  10  11  12
## ARSON               20  11   8  12   5  0   4   7   6  11   9   8
## ASSAULT            360 411 462 459 149 18 435 398 450 524 464 345
## BAD CHECKS           0   1   2   0   0  0   0   0   2   2   0   1
## BRIBERY              1   4   4   2   1  0   1   3   2   0   3   1
## BURGLARY           161 197 233 191  79  2 170 176 196 214 235 206
## DISORDERLY CONDUCT  21  14  10  12   2  1  15   8  10  13  10   5
```

```r
coul3 <- colorRampPalette(brewer.pal(8, "RdBu"))(25)
heatmap_CategoryPdDistrict <- heatmap.2(Matrix_CaMonths,
                                        Colv=FALSE,
                                        srtCol=45,
                                        Rowv=FALSE,
                                        dendrogram="none",
                                        density.info="histogram",
                                        trace="none",
                                        col = coul3,
                                        cexRow=0.85,cexCol=0.75,
                                        xlab = "Months",
                                        main = " Monthly Crimes")
```



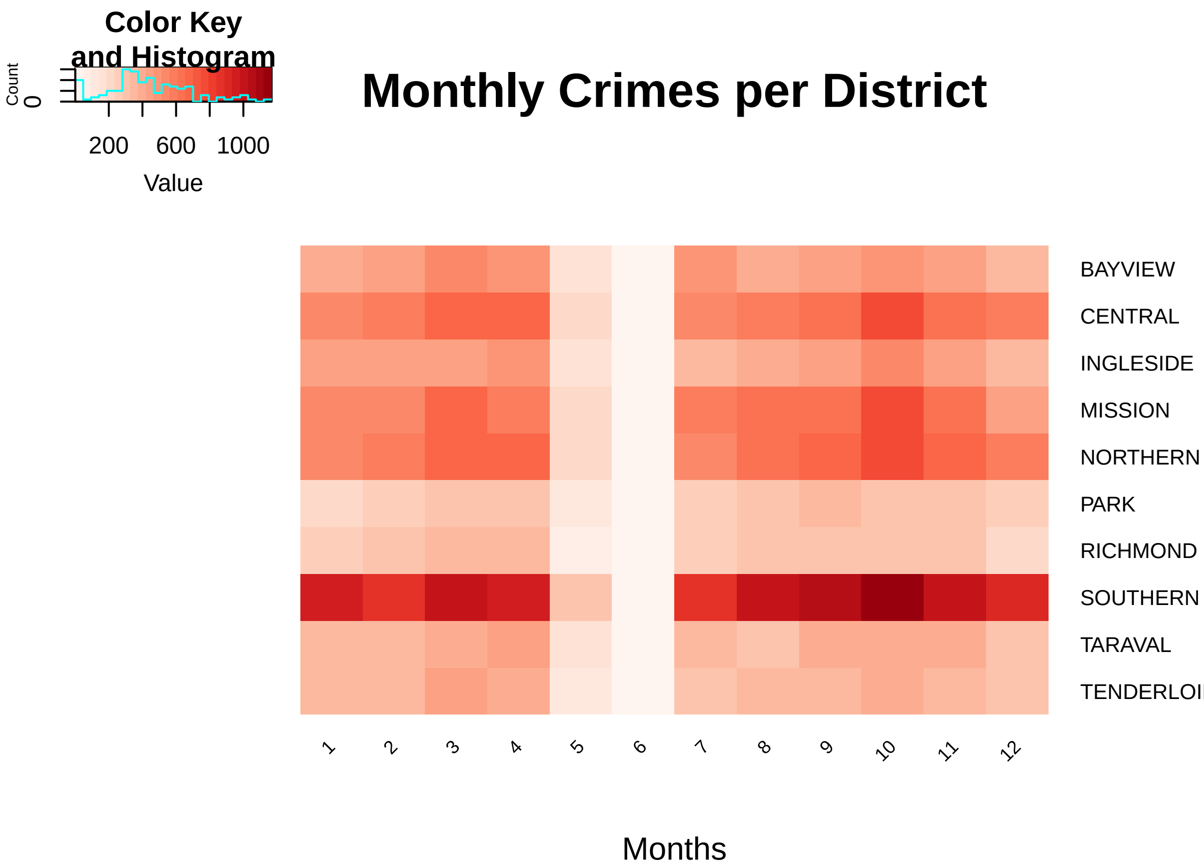
** Heatmaps " Monthly Crimes per District"**. The crime changes during the year, and it is higher in certain districts. Putting together all the heatmaps the type of crime has a correlation on the districts and time of execution. The heatmaps are showing that the hypothesis is supported by the data provided. Meaning, it is correct but only using the same scenario of predictors. The crimes depend directly on the Districts, time, and location, and PdDistricts show a different relationship with the type of crime and the time of execution.

```
##### Heatmaps " Monthly Crimes per District" #############
# Now correlation of PdDistrict with Time. The correlation of the district
# with time and category with time allow to understand the intrisic
# correlation of category with time and districts.

PdDistricTime_data <- training %>% group_by(PdDistrict, Months) %>%
                      summarise(Monthly_crimes = n())

PdMonths <- PdDistricTime_data %>% group_by_at(vars(-Monthly_crimes)) %>%
            mutate(row_id=1:n()) %>% ungroup() %>%
            spread(key=Months, value=Monthly_crimes) %>%
            select(-row_id)

PdMonths[is.na(PdMonths)] <- 0
PdMonths <- as.data.frame(PdMonths)
row.names(PdMonths) <- PdMonths$PdDistrict
```

```r
Matrix1_PdMonths <- data.matrix(PdMonths)
Matrix_PdMonths<- Matrix1_PdMonths[,-1]
head(Matrix_PdMonths)
```

```
##             1   2   3   4   5  6   7   8   9  10  11  12
## BAYVIEW   415 455 525 479 163 10 477 419 450 485 450 366
## CENTRAL   560 594 663 670 213 10 538 569 639 773 639 590
## INGLESIDE 427 452 466 477 173 14 354 416 466 543 442 337
## MISSION   545 527 673 590 229 17 596 625 650 775 650 458
## NORTHERN  536 606 681 689 204 14 540 650 664 752 684 582
## PARK      226 278 300 290 104  3 264 298 346 305 299 240
```

```r
coul4 <- colorRampPalette(brewer.pal(8, "Reds"))(25)
heatmap_CategoryPdDistrict <- heatmap.2(Matrix_PdMonths,
                                        Colv=FALSE,
                                        Rowv=FALSE,
                                        srtCol=45,
                                        dendrogram="none",
                                        density.info="histogram",
                                        trace="none",
                                        col = coul4,
                                        cexRow=0.85,cexCol=0.75,
                                        xlab = "Months",
                                        main = "Monthly Crimes per District")
```




```
# The heatmaps are telling that the type ofcrimes depends directly on
# the Districts and on the time of execution.
```

The correlation map and heatmaps demonstrate that it is better to segment the data by the type of crime over the districts to ensure high accuracy. Now, in order to ensure the correct segmentation, the FBI crime characterization codes are used to have a more realistic model. Using the FBI's UCR codes will reduce the concern of bucketing incorrectly. As Developersgoogle,2019 explains splitting data must be done with caution because some buckets could have many points, while others few or none.

```
# the data required to be segmented over the crime type. Therefore, the FBI crime
#  characterization will give the correct segemnation of crimes.

# Grouping according with FBI codes:
#  https://ucr.fbi.gov/nibrs/2011/resources/nibrs-offense-codes/view

FBI_groupA <- c("ARSON","ASSAULT", "BRIBERY",      "BURGLARY","FRAUD","DRUG.NARCOTIC","EM
                "EXTORTION","SECONDARY.CODES",
                "FRAUD","FORGERY.COUNTERFEITING","GAMBLING",
                "KIDNAPPING","LARCENY.THEFT","MISSING.PERSON",
                "PROSTITUTION","ROBBERY","VANDALISM",
                "SEX.OFFENSES.FORCIBLE","SEX.OFFENSES.NON.FORCIBLE",
                "STOLEN.PROPERTY", "VEHICLE.THEFT","WEAPON.LAWS")
```

```r
FBI_groupB <- c("BAD.CHECKS","DISORDERLY.CONDUCT","DRIVING.UNDER.THE.INFLUENCE",
                "DRUNKENNESS","FAMILY.OFFENSES","LIQUOR.LAWS",
                "LOITERING","NON.CRIMINAL","OTHER.OFFENSES","RUNAWAY",
                "SUICIDE","SUSPICIOUS.OCC","TRESPASS","WARRANTS")

FBI_violent <- c("ASSAULT","DRUG.NARCOTIC","KIDNAPPING","ROBBERY","DRUG.NARCOTIC")
FBI_property <- c(FBI_groupB,FBI_groupA[!FBI_groupA %in%
                c("ASSAULT","DRUG.NARCOTIC", "KIDNAPPING",
                  "ROBBERY","DRUG.NARCOTIC")])

# Bucketing Category , crime types to be more specific in the prediction
Crime_groupA <-   training %>% filter(Category %in% FBI_groupA) %>%
                  droplevels()
Crime_groupA_test <- testing %>% filter(Category %in% FBI_groupA) %>%
                    droplevels()

Crime_groupB <-   training %>% filter(Category %in% FBI_groupB) %>%
                  droplevels()
Crime_groupB_test <- testing %>% filter(Category %in% FBI_groupB) %>%
                  droplevels()

Crime_violent <-   training %>% filter(Category %in% FBI_violent) %>%
                  droplevels()
Crime_violent_test <- testing %>% filter(Category %in% FBI_violent) %>%
                  droplevels()

Crime_property<-   training %>% filter(Category %in% FBI_property) %>%
                  droplevels()
Crime_property_test <- testing %>% filter(Category %in% FBI_property) %>%
                  droplevels()
```

Modeling

The modeling is under Lda, SVM, and Random Forest models. The data is normalized over distances. Therefore, the models can be used because classification models as SVM require numerical normalized distances to ensure accuracy. Irizarry, R.2019, shows us that a model starts Y ~ sum(sum of predictors) and the train() will get the directly the accuracy of the model over the training model that it is used.–

LDA.–

Starting with the model Lda modeling, the accuracy of San Francisco city without data grouping or bucketing is very low, the data set is spread over a too big area. Then, one more Lda modeling is used over the group of violent crime in the entire San Francisco city. The accuracy improves radically.– The last Lda model uses a more specific data set, the violent crimes per district. This model produces the highest accuracy and it will be improved with SVM and Random Forest tunning.

```
#################### Hypothesis ##########################################



# Hypothesis:Crimes (Category) depends on the week + hour +month +
#                                            year + location (X+Y)
# Let's start looking for the best model to use. LDA is apply to  the whole
# data set to demostrate that it is require to split the Category variable into
# small fractions

Hypothesis <- Category ~ Years_sin + Years_cos + Months_sin+ Months_cos +
              Hours_sin+ Hours_cos+
              Days_sin+ Days_cos+
              Location_x+ Location_y+ Location_z

#################### LDA Modelling ##########################################

######## Entire San Francisco City ##############

# First Using the entire datasets

model.lda.SF<- train(Hypothesis, method = "lda", data = training)

# Cross-validation
predictionSF <- predict(model.lda.SF, newdata = testing)
prediction_ldaSF <- factor(predictionSF, levels = levels(testing$Category))

# Accuracy
Accuracy_ldaSF <- confusionMatrix(prediction_ldaSF, testing$Category)$
                  overall["Accuracy"]

# Creating the table that will store all the Accuracies results to compare results
Accuracy_results.SF <- data_frame(method = " LDA on San Francisco City",
                                  Accuracy = Accuracy_ldaSF)

Accuracy_results.SF
```

```
## # A tibble: 1 x 2
##   method                       Accuracy
##   <chr>                           <dbl>
## 1 " LDA on San Francisco City"    0.286
```

```r
# accuracy is 14%


######## Using Violent data ################

# LDA over Violent crimes , which is the smallest group of crimes

model.lda.Violent <- train(Hypothesis, method = "lda", data = Crime_violent)

# Cross-validation
predictionViolent <- predict(model.lda.Violent , newdata = Crime_violent_test)
prediction_ldaViolent <- factor(predictionViolent,
                                levels = levels(Crime_violent_test$Category))

# Accuracy
Accuracy_ldaViolent <- confusionMatrix(prediction_ldaViolent,
                                Crime_violent_test$Category)$overall["Accuracy"]

Accuracy_results.SF <- bind_rows(Accuracy_results.SF,
                          data_frame(method=" LDA on Violent Crimes in SF city",
                          Accuracy = Accuracy_ldaViolent ))

Accuracy_results.SF
```

```
## # A tibble: 2 x 2
##   method                              Accuracy
##   <chr>                                  <dbl>
## 1 " LDA on San Francisco City"           0.286
## 2 " LDA on Violent Crimes in SF city"    0.746
```

```r
######### Using High crimes rates on  San Francisco's districts#########

# The highest accurary is predicting crimes over specific Districts

Crimes_district <-  Crime_violent%>% filter(PdDistrict %in%
                                    c("BAYVIEW","SOUTHERN"))%>%droplevels()

Crimes_district_test <- Crime_violent_test %>%
                        filter(PdDistrict %in%
                        c("BAYVIEW","SOUTHERN"))%>%droplevels()

model.lda.District <- train(Hypothesis, method = "lda",
                            data = Crimes_district)

# Cross-validation
```

```r
predictionDistrict<- predict(model.lda.District,
                             newdata = Crimes_district_test)
prediction_ldaDistrict <- factor(predictionDistrict,
                          levels = levels(Crimes_district_test$Category))

# Accuracy
Accuracy_ldaDistrict <- confusionMatrix(prediction_ldaDistrict,
                          Crimes_district_test$Category)$overall["Accuracy"]

Accuracy_results.SF.Districts <- data_frame(method=
                                    "LDA on Violent Crime per District",
                                    Accuracy = Accuracy_ldaDistrict )
Accuracy_results.SF.Districts
```

```
## # A tibble: 1 x 2
##   method                            Accuracy
##   <chr>                                <dbl>
## 1 LDA on Violent Crime per District    0.748
```

SVM Modeling

Shiyuan and Peng 2019 have a detail explanation of why the SVM is a Euclidean model. The predictors must be numerical and with the same dimensions because numerical distance are calculated as part of the statistical analysis. That is why, to include the categorical PdDistrict PdDistrict would require to transform it into numerical data, using dummy variables. About the transformation of categorical data, Trochim,2006 explains that the best way is to use dummy variables function, which essentially gives a Boolean number to each categorical data to be transformed to numeric data. For instance A, B. C categorical data will be 001, 010, 011, this process will allow us to used categorical data without introducing new characteristics to the original data, and SVM will process the model without errors. This is not part of this study so it is not included in the hypothesis.– Predictors requirement of being numerical create the necessity of changing the location data from degrees dimensions to cartesian dimension, and the time dimension to a circular unitary data so the process of SVM will faster and increase accuracy.– The SVM modeling is applied over the data sets with the highest accuracy found in the LDA models, the Violent crimes in San Francisco city and the data set on violent crimes per district. After tunning the SVM model the accuracy achieve is 77%

```r
#################### SVM Modeling ##############################################


# Svm method tuning Group A
grid <- expand.grid(C = c(0,0.01,0.05,0.1,0.25,0.5,0.75,1,1.25,1.5,1.75,2,5))
svm_model <- svm(Hypothesis, data = Crimes_district, method ="radial",
                 trControl = trctrl, preProcess = c("center", "scale"),
```

```r
                 tuneGrid = grid,tuneLength=10)

# better results gives cost= 1
svm_model <- svm(Hypothesis, data = Crimes_district, method ="radial",
                 trControl = trctrl, preProcess = c("center", "scale"),
                 cost= 1,tuneGrid = grid,tuneLength=10)

# Cross validation svm_model2
svm_prediction <- predict(svm_model, newdata = Crimes_district_test)
svm_prediction.District <- factor(svm_prediction,
                              levels = levels(Crimes_district_test$Category))

Accuracy_svm <- confusionMatrix(data = svm_prediction.District,
                                reference = Crimes_district_test$Category)$
                                overall["Accuracy"]

Accuracy_results.SF.Districts <- bind_rows(Accuracy_results.SF.Districts,
                              data_frame(method=
                              "SVM on Violent Crimes per District",
                              Accuracy = Accuracy_svm ))
Accuracy_results.SF.Districts
```

```
## # A tibble: 2 x 2
##   method                              Accuracy
##   <chr>                                  <dbl>
## 1 LDA on Violent Crime per District      0.748
## 2 SVM on Violent Crimes per District     0.748
```

```r
############### SVM Tunning #########################

District.crimes <-Crimes_district %>%
                  filter(Category %in%
                  c("ASSAULT","ROBBERY"))%>%droplevels()

District.crimes.test <- Crimes_district_test %>%
                        filter(Category %in%
                        c("ASSAULT", "ROBBERY"))%>%droplevels()

trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
svm_model.district.crimes <- svm(Hypothesis, data =District.crimes,
                                 method ="linear",
                                 trControl = trctrl,
                                 preProcess = c("center", "scale"),
                                 tuneGrid = grid,tuneLength=10)
```

```r
svm_district.crimes <- predict(svm_model.district.crimes,
                               newdata = District.crimes.test)
svm_district.crimes.type <- factor(svm_district.crimes,
                            levels = levels(District.crimes.test$Category))

Accuracy_district_type <- confusionMatrix(data = svm_district.crimes.type,
                              reference = District.crimes.test$Category)$
                              overall["Accuracy"]

Accuracy_results.SF.Districts <- bind_rows(Accuracy_results.SF.Districts,
                              data_frame(method=
                              "SVM tunned on Violent Crimes per District",
                               Accuracy = Accuracy_district_type ))
Accuracy_results.SF.Districts
```

```
## # A tibble: 3 x 2
##   method                                      Accuracy
##   <chr>                                          <dbl>
## 1 LDA on Violent Crime per District              0.748
## 2 SVM on Violent Crimes per District             0.748
## 3 SVM tunned on Violent Crimes per District      0.774
```

##** Random Forest** The SVM model gave high accuracy results, but Random Forest could improve the results because it is a model that is accurate when classification is required. Predicting crimes on districts as a function of time and location is a decision tree classification, and Random Forest perform was designed for that type of cases.– Random Forest will be applied over the same data sets as the SVM modeling to compare results. The accuracy is 78% after tunning the model, giving the expected predictions .

```r
##################### Random Forest Modelling #########################################



######### Using High crimes rates on  San Francisco's districts#########

rf_training.districts <-train(Hypothesis,data =District.crimes,method="rf")



rf_district_type<- predict(rf_training.districts,
                           newdata = District.crimes.test)
rf_district12_type <- factor(rf_district_type,
                        levels = levels(District.crimes.test$Category))

Accuracy_district_rf <- confusionMatrix(data = rf_district12_type,
                                        reference = District.crimes.test$
```

```r
                                                Category)$overall["Accuracy"]

Accuracy_results.SF.Districts <- bind_rows(Accuracy_results.SF.Districts,
                              data_frame(method=
                                    "RF on Violent crimes per District",
                                    Accuracy = Accuracy_district_rf))
Accuracy_results.SF.Districts
```

```
## # A tibble: 4 x 2
##   method                                    Accuracy
##   <chr>                                        <dbl>
## 1 LDA on Violent Crime per District            0.748
## 2 SVM on Violent Crimes per District           0.748
## 3 SVM tunned on Violent Crimes per District    0.774
## 4 RF on Violent crimes per District            0.749
```

```r
############## Random Foresr Tunnig ###########################

trControl <- trainControl(method = "cv",
                          number = 10,
                          search = "grid")
## mtry###

set.seed(1234)
tuneGrid <- expand.grid(.mtry = c(5:15))
rf_mtry <- train(Hypothesis,
                 data = District.crimes,
                 method = "rf",
                 metric = "Accuracy",
                 tuneGrid = tuneGrid,
                 trControl = trControl,
                 nodesize = 14,
                 ntree = 300)
print(rf_mtry)
```

```
## Random Forest
##
## 1620 samples
##   11 predictor
##    2 classes: 'ASSAULT', 'ROBBERY'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1458, 1457, 1458, 1458, 1458, 1458, ...
## Resampling results across tuning parameters:
```

```
##
##   mtry  Accuracy   Kappa
##    5    0.7586249  0.08045403
##    6    0.7567615  0.08543238
##    7    0.7561443  0.08871894
##    8    0.7555308  0.08461433
##    9    0.7542924  0.08728254
##   10    0.7518309  0.07845514
##   11    0.7567692  0.09345103
##   12    0.7536866  0.08682631
##   13    0.7561481  0.09564321
##   14    0.7518194  0.09030722
##   15    0.7561519  0.09361793
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 5.
```

```r
best_mtry <- rf_mtry$bestTune$mtry
best_mtry
```

```
## [1] 5
```

```r
## Max nodes##

store_maxnode <- list()
tuneGrid <- expand.grid(.mtry = best_mtry)
trControl <- trainControl(method = "cv",
                          number = 10,
                          search = "grid")
for (maxnodes in c(30: 40)) {
  set.seed(1234)
  rf_maxnode <- train(Hypothesis,
                      data = District.crimes,
                      method = "rf",
                      metric = "Accuracy",
                      tuneGrid = tuneGrid,
                      trControl = trControl,
                      nodesize = 14,
                      maxnodes = maxnodes,
                      ntree = 300)
  current_iteration <- toString(maxnodes)
  store_maxnode[[current_iteration]] <- rf_maxnode
}
results_mtry <- resamples(store_maxnode)
summary(results_mtry)
```

```
##
## Call:
## summary.resamples(object = results_mtry)
##
## Models: 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40
## Number of resamples: 10
##
## Accuracy
##         Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## 30 0.7592593 0.7666207 0.7716049 0.7716032 0.7765849 0.7839506    0
## 31 0.7654321 0.7666207 0.7746914 0.7728340 0.7777778 0.7791411    0
## 32 0.7654321 0.7666207 0.7716049 0.7722167 0.7762346 0.7839506    0
## 33 0.7654321 0.7705410 0.7746914 0.7734513 0.7777778 0.7791411    0
## 34 0.7530864 0.7666207 0.7716049 0.7703686 0.7765849 0.7777778    0
## 35 0.7654321 0.7705410 0.7746914 0.7734513 0.7777778 0.7791411    0
## 36 0.7654321 0.7705410 0.7746914 0.7740685 0.7777778 0.7839506    0
## 37 0.7530864 0.7705410 0.7716049 0.7722167 0.7777778 0.7839506    0
## 38 0.7592593 0.7643394 0.7685185 0.7691264 0.7762346 0.7791411    0
## 39 0.7530864 0.7643394 0.7692381 0.7678994 0.7716049 0.7777778    0
## 40 0.7469136 0.7643394 0.7716049 0.7691264 0.7777778 0.7791411    0
##
## Kappa
##           Min.      1st Qu.      Median        Mean    3rd Qu.       Max. NA's
## 30 -0.03539823 -0.021022695 0.008849558 0.005326369 0.02962211 0.04255319    0
## 31 -0.02395210 -0.007671799 0.029940120 0.019134171 0.02994012 0.06827564    0
## 32 -0.02395210 -0.012164801 0.011756534 0.014961560 0.03939992 0.06827564    0
## 33 -0.02395210 -0.009175871 0.023819617 0.014559655 0.02994012 0.06827564    0
## 34 -0.04651163 -0.009175871 0.011756534 0.008637343 0.02994012 0.05573822    0
## 35 -0.02395210  0.020759366 0.029940120 0.026045700 0.03930325 0.06827564    0
## 36 -0.02395210  0.020759366 0.031919212 0.030131334 0.04466018 0.07079646    0
## 37 -0.01694915  0.001453488 0.017699115 0.021098088 0.02994012 0.07079646    0
## 38 -0.03539823 -0.021120102 0.011756534 0.006230010 0.02687987 0.06827564    0
## 39 -0.02410445  0.004424779 0.019991619 0.015395269 0.02994012 0.04354540    0
## 40 -0.05730659 -0.002844506 0.023112118 0.012299249 0.02994012 0.06827564    0
```

```r
## ntrees ##

store_maxtrees <- list()
for (ntree in c(400, 450, 500, 550, 600, 800, 1000, 2000,2500, 2700,3000))
  {
  set.seed(5678)
  rf_maxtrees <- train(Hypothesis,
                       data = District.crimes,
                       method = "rf",
                       metric = "Accuracy",
```

```r
                         tuneGrid = tuneGrid,
                         trControl = trControl,
                         nodesize = 14,
                         maxnodes = 30,
                         ntree = ntree)
  key <- toString(ntree)
  store_maxtrees[[key]] <- rf_maxtrees
}
results_tree <- resamples(store_maxtrees)
summary(results_tree)
```

```
##
## Call:
## summary.resamples(object = results_tree)
##
## Models: 400, 450, 500, 550, 600, 800, 1000, 2000, 2500, 2700, 3000
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## 400  0.7530864 0.7677000 0.7770876 0.7728399 0.7788003 0.7839506    0
## 450  0.7530864 0.7677000 0.7747018 0.7716091 0.7777778 0.7839506    0
## 500  0.7530864 0.7677000 0.7747018 0.7716091 0.7777778 0.7839506    0
## 550  0.7530864 0.7630987 0.7753920 0.7716167 0.7814010 0.7839506    0
## 600  0.7530864 0.7677000 0.7747018 0.7722264 0.7777778 0.7839506    0
## 800  0.7530864 0.7677000 0.7747018 0.7722264 0.7777778 0.7839506    0
## 1000 0.7530864 0.7677000 0.7747018 0.7728436 0.7824074 0.7839506    0
## 2000 0.7530864 0.7677000 0.7753920 0.7728475 0.7814010 0.7839506    0
## 2500 0.7530864 0.7677000 0.7753920 0.7734648 0.7814010 0.7839506    0
## 2700 0.7530864 0.7719552 0.7763975 0.7740820 0.7777778 0.7839506    0
## 3000 0.7530864 0.7719552 0.7763975 0.7740820 0.7777778 0.7839506    0
##
## Kappa
##             Min.     1st Qu.     Median        Mean    3rd Qu.       Max. NA's
## 400  -0.04651163 0.004128295 0.02363944 0.013810517 0.03836738 0.04255319    0
## 450  -0.04651163 0.000000000 0.01699814 0.011537778 0.02990404 0.04255319    0
## 500  -0.04651163 0.000000000 0.01699814 0.008581530 0.02990404 0.04255319    0
## 550  -0.04651163 0.000000000 0.01109007 0.008668427 0.03934569 0.04255319    0
## 600  -0.04651163 0.000000000 0.01699814 0.009692870 0.02990404 0.04255319    0
## 800  -0.04651163 0.000000000 0.01699814 0.012659627 0.02990404 0.04255319    0
## 1000 -0.04651163 0.000000000 0.01699814 0.016745261 0.03936384 0.07079646    0
## 2000 -0.04651163 0.000000000 0.01699814 0.013772503 0.03939992 0.07059212    0
## 2500 -0.04651163 0.001453488 0.01699814 0.017893722 0.03939992 0.07059212    0
## 2700 -0.04651163 0.004128295 0.02979578 0.019048489 0.03939992 0.04584527    0
```

```
## 3000 -0.04651163 0.004128295 0.02979578 0.019048489 0.03939992 0.04584527     0
```

```
#### Random Forest with tunning  results ############

model.RF <- train(Hypothesis,
                  data = District.crimes,
                  method = "rf",
                  metric = "Accuracy",
                  tuneGrid = tuneGrid,
                  trControl = trControl,
                  nodesize = 14,
                  maxnodes = 30,
                  ntree = 2000)

RF_fit <- predict(model.RF, newdata = District.crimes.test)
RF_fit2 <- factor(RF_fit, levels = levels(District.crimes.test$Category))

Accuracy_RF_fit <- confusionMatrix(data = RF_fit2,
                                    reference = District.crimes.test$
                                    Category)$overall["Accuracy"]

Accuracy_results.SF.Districts <- bind_rows(Accuracy_results.SF.Districts,
                              data_frame(method=
                                "RF tunned on Violent crimes per District",
                                Accuracy = Accuracy_RF_fit))
Accuracy_results.SF.Districts
```

```
## # A tibble: 5 x 2
##   method                                       Accuracy
##   <chr>                                           <dbl>
## 1 LDA on Violent Crime per District               0.748
## 2 SVM on Violent Crimes per District              0.748
## 3 SVM tunned on Violent Crimes per District       0.774
## 4 RF on Violent crimes per District               0.749
## 5 RF tunned on Violent crimes per District        0.779
```

```
Accuracy_results.SF
```

```
## # A tibble: 2 x 2
##   method                                 Accuracy
##   <chr>                                     <dbl>
## 1 " LDA on San Francisco City"              0.286
## 2 " LDA on Violent Crimes in SF city"       0.746
```

Conclusion

The project has required multiple modeling tasks to obtain the best accuracy results. Grouping the data sets was necessary to reduce the area of application and increases the quantity of data per district. Working by districts and the type of crime to predict is the most accurate form of modeling. That is why the models went from 24% to 78% of accuracy.– The analysis performed shows how to looking for the correct model and tuning the models and recognize what is the best way to approach unknown results. Splitting the project into steps to initiate prediction behavior and using multiple models allows to find better results.– This analysis could be improved including more data, for instance incorporating Zip Codes of the PdDistrict predictor and the police code for the resolutions. Including those data will allow the SVM model, Euclidean model, to incorporate them and improve accuracy. We can hypothesize more accuracy because PDdistricts and Zipcodes will give specific addresses as gas stations or banks that are more sensitive to violent crime.

**** REFERENCES****

Abramson, J.2019. Unit Circle - Sine and Cosine Functions. Retrieved from: https://math.libretexts.org/Bookshelves/Precalculus/Book%3A_Precalculus_(OpenStax)/05%3A_Trigonometric_Functions/5.03%3A_Unit_Circle_-_Sine_and_Cosine_Functions

Irizarry, R.2019. Introduction to Data Science.Retrieved from file:///Users/PedroCornejo/Documents/Data%20Science/datasciencebook.pdf

Barett,M. Why should I use the here package when I'm already using projects?2019. Retrieved from: https://malco.io/2018/11/05/why-should-i-use-the-here-package-when-i-m-already-using-projects/

Developersgoogle,2019. Bucketing. Retrieved from https://developers.google.com/machine-learning/data-prep/transform/bucketing

FBI.2019.NIBRS Offense Codes. retrieved from: https://ucr.fbi.gov/nibrs/2011/resources/nibrs-offense-codes/view

Prabhakaran,S.2017. How to make any plot in ggplot2?. Retrieved from: http://r-statistics.co/ggplot2-Tutorial-With-R.html

R mutipurose kit.2015. Polar, Spherical and Geographic Coordinates.2015. Retrieved from: https://vvvv.org/blog/polar-spherical-and-geographic-coordinates

R-project-gplots.2019. Package gplots. Retrived from: https://cran.r-project.org/web/packages/gplots/gplots.pdf

Shiyuan and Peng 2019.Preprocessing of categorical predictors in SVM, KNN and KDC. Retrieved from https://stats.libretexts.org/Bookshelves/Advanced_Statistics_Computing/RTG%3A_Classification_Methods/4%3A_Numerical_Experiments_and_Real_Data_Analysis/Preprocessing_of_categorical_predictors_in_SVM%2C_KNN_and_KDC_(contributed_by_Xi_Cheng)

Trochim,2006.Dummy Variables. Retrieved from https://socialresearchmethods.net/kb/dummyvar.php

“‘