```r
#CIND820 Captstone Project 'ASSESSMENT OF TORONTO CRIME DATA THROUGH EXPLORATORY DATA ANALYSIS AND CLASSIFICATION METHODS'

#Load all the libraries, won't necessarily use them all but the are frequently used
library(ggplot2)
library(ggmap)
library(tidyverse)
library(rgdal)
library(readxl)
library(sf)
library(data.table)
library(plyr)
library(dplyr)
library(hablar)
library(ggthemes)
library(tidyr)
library(ggrepel)
library(lubridate)
library(mapproj)
library(RColorBrewer)
library(viridisLite)


#Load Toronto Major Crime Indicator dataset
crime<-read.csv(file ="D:/Ryerson Big Data/CIND820 Big Data Analytics Project/TorontoCrime/Major_Crime_Indicators.csv", header = T, na.

#check structure
str(crime)

#summary of data
summary(crime)

#summary of datatypes
table(sapply(crime, class))

#confirm number of complete records
sum(complete.cases(crime))

#check for occurrence dates before 2014 in the MCI
sum(crime$occurrenceyear < 2014)

#find records with the same ID and offense type
dupRow <- crime[which(duplicated(crime[,c('event_unique_id','offence')])==TRUE),]
nrow(dupRow)


#Start cleaning dataset
```

```
#feature reduction - remove redundant variables
MCI_cln <- crime[ -c(1:3,7,8,11,13:18,21, 22, 30) ]

#check occurrence year values
range(MCI_cln$occurrenceyear) #can see the column contains occurrences before 2014

#filter for occurrence dates between 2014 and 2021
MCI_cln<-filter(MCI_cln,occurrenceyear=='2014'| occurrenceyear=='2015' | occurrenceyear=='2016' | occurrenceyear=='2017' | occurrenceye

range(MCI_cln$occurrenceyear) #confirm range

#remove records with same ID and offense type, rename dataset
MCI_cln<-MCI_cln %>%
  distinct(event_unique_id, offence, .keep_all = TRUE)


#convert selected categorical columns to factor variables
MCI_cln$Division<-as.factor(MCI_cln$Division)
MCI_cln$premises_type<-as.factor(MCI_cln$premises_type)
MCI_cln$offence<-as.factor(MCI_cln$offence)
MCI_cln$occurrencemonth<-factor(MCI_cln$occurrencemonth, levels=c("January","February","March","April","May", "June", "July", "August",
MCI_cln$occurrencedayofweek = gsub(" ", "", MCI_cln$occurrencedayofweek)
MCI_cln$occurrencedayofweek<-factor(MCI_cln$occurrencedayofweek, levels=c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Satu
MCI_cln$MCI<-as.factor(MCI_cln$MCI)
MCI_cln$Neighbourhood<-as.factor(MCI_cln$Neighbourhood)

#add a new column "Weight" via the matched UCR columns in the MCI and Weights dataframes, then move after ucr
MCI_cln$weight <- Weights$Weighting[match(MCI_cln$ucr, Weights$UCR)]

MCI_cln <- MCI_cln %>% relocate(weight, .before = offence)

#Add a season column based on months then convert to factor
MCI_cln$season <- ifelse(MCI_cln$occurrencemonth %in% c('December','January','February'), "Winter",
                         ifelse (MCI_cln$occurrencemonth %in% c('September','October', 'November'), "Autumn",
                                 ifelse (MCI_cln$occurrencemonth %in% c('March','April','May'),
                                         "Spring", "Summer")))
#Convert Season to a factor
MCI_cln$season<-factor(MCI_cln$season, levels=c("Winter", "Spring", "Summer", "Autumn"))


#Figure 1: structure of cleaned Toronto MCI database
str(MCI_cln)


#Load Toronto Neighbourhood Profiles and check structure, entries, and variables.  Dataset was cumbersome and cleaned in another notebo

Nhoods<-read.csv(file ="D:/Ryerson Big Data/CIND820 Big Data Analytics Project/TorontoCrime/Neighbourhoods.csv", header = T, na.strings

#check the structure
str(Nhoods)

#check datatypes
table(sapply(Nhoods, class))

#check for missing values
sum(is.na(Nhoods))

#check for duplicates
sum(duplicated(Nhoods$X_id))

#Transpose database
Hoods_cln<-t(Nhoods)

#Confirm  the new file is a dataframe
str(Nhoods)
```

```r

exists("Hoods_cln")&&is.data.frame(get("Hoods_cln"))

#Convert to a data frame & confirm updated structure

Hoods_cln <- as.data.frame(Hoods_cln)

exists("Hoods_cln")&&is.data.frame(get("Hoods_cln"))


#read CSI weights file
Weights<-read_excel("D:\\Ryerson Big Data\\CIND820 Big Data Analytics Project\\TorontoCrime\\CSI_weights2020.xlsx")

str(Weights)


#read police divisions shape file
Patrols <- st_read(
  "D:/Ryerson Big Data/CIND820 Big Data Analytics Project/TorontoCrime/ShapeFiles/Police_Divisions.shp")

str(Patrols)


#Figure 2: Plot of Patrol Zones

ggplot() +
  geom_sf(data = Patrols, size = 1, color = "black", fill = "white") +
  ggtitle("Toronto Police Patrol Zones") +
  xlab("Longitude") +
  ylab("Latitude")+
  coord_sf()

#read neighbourhoods shape file
Neighbourhoods <- st_read(
  "D:/Ryerson Big Data/CIND820 Big Data Analytics Project/TorontoCrime/ShapeFiles/Neighbourhoods.shp")
str(Neighbourhoods)
head(Neighbourhoods)


#Figure 3: Plot of Toronto Neighbourhoods
ggplot() +
  geom_sf(data = Neighbourhoods, size = 1, color = "black", fill = "white") +
  ggtitle("Toronto Neighbourhoods") +
  xlab("Longitude") +
  ylab("Latitude")+
  coord_sf()



#EDA & Descriptive Statistics Toronto MCI Dataset

#Figure 4: incidents per year
IncYear<-count(MCI_cln$occurrenceyear)
setnames(IncYear, "x", "Year")
setnames(IncYear, "freq", "IncidentCounts")

#Calc avg yearly crime count between 2014 and 2021 for Toronto
AvgCrime<-(sum(IncYear$IncidentCounts))/8

write.table(IncYear, file = "IncYear.txt", sep = ",", quote = FALSE, row.names = F)

ggplot(data=MCI_cln, aes(MCI_cln$occurrenceyear)) +
  geom_histogram(binwidth = 1, color="black", fill="blue") + scale_x_continuous(breaks =  2014:2021) +
  labs(title = "Toronto Crime - Incidents per Year", x = "Year", y = "Frequency") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
```

```r
       panel.background = element_blank(), axis.line = element_line(colour = "black"))


#Figure 5: boxplot of incidents per year
BP_Yr <-MCI_cln %>% group_by(occurrenceyear) %>%
  dplyr::summarise(N = n())

ggplot(BP_Yr, aes(x="", N, y=N)) +
  geom_boxplot(width=0.6, outlier.size=3,outlier.colour="black", fill = 'blue') +
  stat_summary(
    aes(label=sprintf("%1.1f", ..y..)),
    geom="text",
    fun = function(y) boxplot.stats(y)$stats,
    position=position_nudge(x=0.33),
    size=3.5) +
  theme_bw() +
  stat_boxplot(geom = "errorbar", width = 0.5) +
  xlab("Year") + ylab("Count") +
  ggtitle("Range of Incident Counts per Year")



#Figure 6: incidents per month
IncMonth<-count(MCI_cln$occurrencemonth)
setnames(IncMonth, "x", "Month")
setnames(IncMonth, "freq", "IncidentCounts")

ggplot(MCI_cln, aes(x = occurrencemonth, fill=occurrencemonth)) +
  geom_bar(width=0.8, stat="count") + scale_fill_brewer(palette="Set3") +
  theme(legend.position="none") + scale_x_discrete()+
  ggtitle("Incidents per Month") + xlab("Month") + ylab("Count") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))

#Figure 7: boxplot of incidents per month
BP_M <-MCI_cln %>% group_by(occurrencemonth) %>%
  dplyr::summarise(N = n())


ggplot(BP_M, aes(x="", N, y=N)) +
  geom_boxplot(width=0.6, outlier.size=3,outlier.colour="black", fill = "aquamarine") +
  stat_summary(
    aes(label=sprintf("%1.1f", ..y..)),
    geom="text",
    fun = function(y) boxplot.stats(y)$stats,
    position=position_nudge(x=0.33),
    size=3.5) +
  theme_bw() +
  stat_boxplot(geom = "errorbar", width = 0.5) +
  xlab("Month") + ylab("Count") +
  ggtitle("Range of Incident Counts per Month")



#Figure 8: incidents per day
IncDay<-count(MCI_cln$occurrencedayofweek)
setnames(IncDay, "x", "Day")
setnames(IncDay, "freq", "IncidentCounts")

ggplot(MCI_cln, aes(x = occurrencedayofweek, fill=occurrencedayofweek)) +
  geom_bar(width=0.8, stat="count") + scale_fill_brewer(palette="Set3") +
  theme(legend.position="none") + scale_x_discrete()+
  ggtitle("Incidents per Day") + xlab("Day") + ylab("Count") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))
```

```r

#Figure 9: boxplot of incidents per day
BP_dow <-MCI_cln %>% group_by(occurrencedayofweek) %>%
  dplyr::summarise(N = n())


ggplot(BP_dow, aes(x="", N, y=N)) +
  geom_boxplot(width=0.6, outlier.size=3,outlier.colour="black", fill = "cadetblue") +
  stat_summary(
    aes(label=sprintf("%1.1f", ..y..)),
    geom="text",
    fun = function(y) boxplot.stats(y)$stats,
    position=position_nudge(x=0.33),
    size=3.5) +
  theme_bw() +
  stat_boxplot(geom = "errorbar", width = 0.5) +
  xlab("Day") + ylab("Count") +
  ggtitle("Range of Incident Counts per Day")



#Figure 10: incidents per season
IncSeason<-count(MCI_cln$season)
setnames(IncSeason, "x", "Season")
setnames(IncSeason, "freq", "IncidentCounts")

ggplot(MCI_cln, aes(x = season, fill=season)) +
  geom_bar(width=0.8, stat="count") + scale_fill_brewer(palette="Set3") +
  theme(legend.position="none") + scale_x_discrete()+
  ggtitle("Incidents per Season ") + xlab("Season") + ylab("Count") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))

#Figure 11: incidents per premises type
IncPrem<-count(MCI_cln$premises_type)
setnames(IncPrem, "x", "Premises")
setnames(IncPrem, "freq", "IncidentCounts")


ggplot(MCI_cln, aes(x = premises_type, fill=premises_type)) +
  geom_bar(width=0.8, stat="count") + scale_fill_brewer(palette="Set3") +
  theme(legend.position="none") + scale_x_discrete()+
  ggtitle("Incidents per Premises Type ") + xlab("Premises Type") + ylab("Count") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))


#Figure 12: incidents per hour
IncHour<-count(MCI_cln$occurrencehour)
setnames(IncHour, "x", "Hour")
setnames(IncHour, "freq", "IncidentCounts")


ggplot(MCI_cln, aes(x = occurrencehour, fill=as.factor(occurrencehour))) +
  geom_bar(width=0.8, stat="count", fill = plasma(24)) + theme(legend.position="none") +  scale_x_continuous(breaks =  0:23)+
  ggtitle("Incidents per Hour ") + xlab("Hour") + ylab("Count") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))


#Figure 13: Day/time trends
daytime <-MCI_cln %>% group_by(occurrencedayofweek,occurrencehour) %>%
  dplyr::summarise(N = n())  #sample code found here - https://stackoverflow.com/questions/22767893/count-number-of-rows-by-group-using

ggplot(daytime, aes(occurrencedayofweek, occurrencehour, fill = N)) +
```

```r
  geom_tile(size = 1, color = "white") +
  scale_fill_gradient2('N', low = "darkslategray3", high = "darkslategray4", midpoint = 1750) +
  scale_y_continuous(breaks =  0:23) +
  ggtitle("Toronto Crimes by Day and Time") + xlab("Day") + ylab("Hour") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))

#Figure 14: Updated Table of incidents per Neighbourhood with NSA removed
IncHood2<-count(MCI_RemNSA$Neighbourhood)
setnames(IncHood2, "x", "Neighbourhood")
setnames(IncHood2, "freq", "IncidentCounts")
IncHood2_TopInc<-head(IncHood2,10)  #some weirdness with ordering top = last 10 and last = top 10
IncHood2_LastInc<-head(IncHood2,10)

BP_IncHood <-MCI_RemNSA %>% group_by(Neighbourhood) %>%
  dplyr::summarise(N = n())


ggplot(BP_IncHood, aes(x="", N, y=N)) +
  geom_boxplot(width=0.6, outlier.size=3,outlier.colour="black", fill = "cadetblue") +
  stat_summary(
    aes(label=sprintf("%1.1f", ..y..)),
    geom="text",
    fun = function(y) boxplot.stats(y)$stats,
    position=position_nudge(x=0.33),
    size=3.5) +
  theme_bw() +
  stat_boxplot(geom = "errorbar", width = 0.5) +
  xlab("Neighbourhood") + ylab("Count") +
  ggtitle("Range of Incident Counts per Neighbourhood")


#Fig 15: incidents per neighbourhood & plot top 10 neighbourhood with most crime
IncHood<-count(MCI_cln$Neighbourhood)
setnames(IncHood, "x", "Neighbourhood")
setnames(IncHood, "freq", "IncidentCounts")

MCI_RemNSA<-MCI_cln[!grepl("NSA", MCI_cln$Neighbourhood),] #Unlabelled neighbourhoods listed as NSA, removed.
MCIhood <-MCI_RemNSA %>% group_by(Neighbourhood, MCI) %>%
  dplyr::summarise(N = n())
MCIhood <- MCIhood[order(MCIhood$N),]
MCIhood_top10<-tail(MCIhood, 10)

ggplot(aes(x = reorder(Neighbourhood, N), y = N), data = MCIhood_top10) +
  geom_bar(stat = 'identity', width = 0.6, fill = plasma(10)) +
  geom_text(aes(label = N), stat = 'identity', data = MCIhood_top10, hjust = -0.1, size = 3) +
  coord_flip() +
  xlab('Neighbourhoods') +
  ylab('Incident Counts') +
  ggtitle('Top 10 Toronto Neighbourhoods with the Most Crime') +
  theme_bw() +
  theme(plot.title = element_text(size = 14),
        axis.title = element_text(size = 12, face = "bold"))



#Figure 16: Neighbourhoods with the least crime
IncHood3 <-MCI_RemNSA %>% group_by(Neighbourhood) %>%  #code from Sundar (2020), Li (2017)
  dplyr::summarise(N = n())
IncHood3 <- IncHood3[order(IncHood3$N), ]  #so much drama with these files
IncHood3_Last10 <- head(IncHood3, 10)
IncHood3_Top10 <- tail(IncHood3, 10)

ggplot(aes(x = reorder(Neighbourhood, N), y = N), data = IncHood3_Top10) +
  geom_bar(stat = 'identity', width = 0.6, fill = plasma(10)) +
```

```r
  geom_text(aes(label = N), stat = 'identity', data = IncHood3_Top10, hjust = -0.1, size = 3) +
  coord_flip() +
  xlab('Neighbourhoods') +
  ylab('Incident Counts') +
  ggtitle('Top 10 Toronto Neighbourhoods with the Most Crime') +
  theme_bw() +
  theme(plot.title = element_text(size = 14),
        axis.title = element_text(size = 12, face = "bold"))

#Figure 17: Listing of incident types and counts
MCIcat <-MCI_cln %>% group_by(MCI) %>%
  dplyr::summarise(N = n())
MCIcat <- MCIcat[order(MCIcat$N), ]

ggplot(aes(x = reorder(MCI, N), y = N), data = MCIcat) +
  geom_bar(stat = 'identity', width = 0.5, fill = "blue") +
  geom_text(aes(label = N), stat = 'identity', data = MCIcat, hjust = -0.1, size = 3.5) +
  coord_flip() +
  xlab('Major Crime Indicators') +
  ylab('Occurrence Count') +
  ggtitle('Major Crime Indicators Toronto (2014 - 2021)') +
  theme_bw() +
  theme(plot.title = element_text(size = 14),
        axis.title = element_text(size = 12, face = "bold"))

#Figure 18:Listing of all offence types
OffCat <-MCI_cln %>% group_by(offence) %>%
  dplyr::summarise(N = n())
OffCat <- OffCat[order(OffCat$N),]

ggplot(aes(x = reorder(offence, N), y = N), data = OffCat) +
  geom_bar(stat = 'identity', width = 0.5, fill = "blue") +
  geom_text(aes(label = N), stat = 'identity', data = OffCat, hjust = -0.1, size = 3.5) +
  coord_flip() +
  xlab('Offence Types') +
  ylab('Count') +
  ggtitle('Toronto Criminal Offence Types (2014 - 2021)') +
  theme_bw() +
  theme(plot.title = element_text(size = 14),
        axis.title = element_text(size = 12, face = "bold"))

#Figure 18: Inset with Top 10 Offenses
OffCat_Top10 <- tail(OffCat, 10)

ggplot(aes(x = reorder(offence, N), y = N), data = OffCat_Top10) +
  geom_bar(stat = 'identity', width = 0.5, fill = "blue") +
  geom_text(aes(label = N), stat = 'identity', data = OffCat_Top10, hjust = -0.1, size = 3.5) +
  coord_flip() +
  xlab('Offence Types') +
  ylab('Count') +
  ggtitle('Toronto Criminal Offence Types - Top 10') +
  theme_bw() +
  theme(plot.title = element_text(size = 14),
        axis.title = element_text(size = 12, face = "bold"))

#Fig 19: MCI per Neighbourhood
MCI_RemNSA<-MCI_cln[!grepl("NSA", MCI_cln$Neighbourhood),]
MCIhood <-MCI_RemNSA %>% group_by(Neighbourhood, MCI) %>%
  dplyr::summarise(N = n())
MCIhood <- MCIhood[order(MCIhood$N),]
MCIhood_top10<-tail(MCIhood, 10)

ggplot(aes(x = reorder(Neighbourhood, N), y = N), data = MCIhood_top10) +
  geom_bar(stat = 'identity', width = 0.5, fill = "blue") +
  geom_text(aes(label = N), stat = 'identity', data = MCIhood_top10, hjust = -0.1, size = 3.5) +
```

```r
  coord_flip() +
  xlab('Offence Types') +
  ylab('Count') +
  ggtitle('Toronto Criminal Offence Types - Top 10') +
  theme_bw() +
  theme(plot.title = element_text(size = 14),
        axis.title = element_text(size = 12, face = "bold"))



#Figure 19: MCI counts per Neighbourhood
MCI_N <-MCI_RemNSA %>% group_by(MCI, Neighbourhood) %>%
  dplyr::summarise(N = n())
MCI_N <- MCI_N[order(MCI_N$N),]
MCI_N_Top10<-tail(MCI_N, 20)

ggplot(MCI_N_Top10, aes(x = Neighbourhood, y=N, fill = MCI)) +
  geom_bar(stat = 'identity',  width = 0.8) +
  xlab('Neighbourhood') +
  ylab('MCI Count') +
  ggtitle('Top MCI Categories by Neighbourhood (2014 - 2021)') + theme_bw() +
  theme(plot.title = element_text(size = 14),
        axis.title = element_text(size = 12, face = "bold"),
        axis.text.x = element_text(angle = 70, hjust = 1, vjust = 1))


#Figure 20: MCI counts per Premises
MCI_P <-MCI_cln %>% group_by(MCI,premises_type) %>%
  dplyr::summarise(N = n())

ggplot(MCI_P, aes(premises_type, MCI, fill = N)) +
  geom_tile(size = 1, color = "white") +
  scale_fill_gradient2('N', low = "cadetblue", mid = "white", high = "darkslategray", midpoint = 25000) +
  ggtitle("Toronto MCI Categories by Premises Type (2014 - 2021)") + xlab("Premises Type") + ylab("MCI") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))


#Figure 21: MCI counts per Year
MCI_Y <-MCI_cln %>% group_by(occurrenceyear,MCI) %>%
  dplyr::summarise(N = n())

ggplot(MCI_Y, aes(occurrenceyear, MCI, fill = N)) +
  geom_tile(size = 1, color = "white") +
  scale_fill_gradient2('N', low = "darkslategray4", mid = "yellow", high = "darkslategray", midpoint = 12000) +
  ggtitle("Toronto MCI Categories by Year (2014 - 2021)") + xlab("Year") + ylab("MCI") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black")) +
  scale_x_continuous(breaks =  2014:2021)


#Figure 22: MCI counts per Month
MCI_M <-MCI_cln %>% group_by(occurrencemonth,MCI) %>%
  dplyr::summarise(N = n())

ggplot(MCI_M, aes(occurrencemonth, MCI, fill = N)) +
  geom_tile(size = 1, color = "white") +
  scale_fill_gradient2('N', low = "darkslategray4", mid = "yellow", high = "darkslategray", midpoint = 7000) +
  ggtitle("Toronto MCI Categories by Month (2014 - 2021)") + xlab("Month") + ylab("MCI") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))


#Figure 23: MCI counts per Day
MCI_D <-MCI_cln %>% group_by(occurrencedayofweek,MCI) %>%
```

```r
  dplyr::summarise(N = n())

ggplot(MCI_D, aes(occurrencedayofweek, MCI, fill = N)) +
  geom_tile(size = 1, color = "white") +
  scale_fill_gradient2('N', low = "cyan2", mid = "white", high = "cyan4", midpoint = 11000) +
  ggtitle("Toronto MCI Categories by Day (2014 - 2021)") + xlab("Day") + ylab("MCI") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))



#Figure 24: MCI counts per Hour
MCI_H <-MCI_cln %>% group_by(occurrencehour,MCI) %>%
  dplyr::summarise(N = n())

ggplot(MCI_H, aes(occurrencehour, MCI, fill = N)) +
  geom_tile(size = 1, color = "white") +
  scale_fill_gradient2('N', low = "cyan2", mid = "white", high = "cyan4", midpoint = 5000) +
  ggtitle("Toronto MCI Categories by Hour (2014 - 2021)") + xlab("Hour") + ylab("MCI") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))


#Combine the neighbourhood crime counts from MCI dataset with the Neighbourhoods dataframe
#start by adding a new column for avg crime counts per year and populate with AvgCrime (avg for Toronto)
IncHood2$AvgCount<-(IncHood2$IncidentCounts)/8
IncHood2$Number<-MCI_RemNSA$Hood_ID[match(IncHood2$Neighbourhood,MCI_RemNSA$Neighbourhood)]

#add the avg yearly crime counts per neighbourhood
Nhoods$AvgCrime<-IncHood2$AvgCount[match(Nhoods$Neighbourhood.Number, IncHood2$Number)]
#add the avg crime count for Toronto
Nhoods[1,25] = AvgCrime

#generate the average crime per 100k population
Nhoods$Avg100k<-(Nhoods$AvgCrime / Nhoods$Population)*100000

#Add column to designate neighbourhoods as high/low crime
#start by dividing each column by the city average, 1 = TO, >1 = high crime, <1 = low crime;
Nhoods$Ratio<-(Nhoods$Avg100k / 1190.5923)

Nhoods$Ratio[Nhoods$Ratio > 1] <- 1 #high crime area
Nhoods$Ratio[Nhoods$Ratio < 1] <- 0   #low crime

#Convert column to factor
Nhoods$Ratio<-as.factor(Nhoods$Ratio)

HoodOff <-MCI_RemNSA %>% group_by(offence, Neighbourhood) %>%
  dplyr::summarise(N = n())

HoodOff$Wt<-MCI_cln$weight[match(HoodOff$offence,MCI_cln$offence)]
HoodOff$ucr<-MCI_cln$ucr_code[match(HoodOff$offence,MCI_cln$offence)]

#filter for ucr less than 1700
HoodViolent<-filter(HoodOff,ucr < 1700)
HoodViolent$Total<- HoodViolent$N * HoodViolent$Wt


```

kmault / CIND820_Capstone Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main

CIND820_Capstone / CIND820_Assignment3_FeatureSelection_InitialResults.R

kmault Add files via upload History

1 contributor

233 lines (161 sloc) | 6.78 KB

```
#Feature Selection, Initial Results & Code

#Figure 27; final review of dataset, removal of redundant or non-predictive variables
str(MCI_db)

#FEATURE SELECTION

#Check for correlation among variables
set.seed(123)
correlationMatrix <- cor(MCI_db[,1:10])

#Figure 28
print(correlationMatrix)

#Figure 29; Display correlation visually
corrplot(correlationMatrix, method="circle")

#Feature Selection method 1: Boruta
#Boruta method of show variable importance; http://r-statistics.co/Variable-Selection-and-Importance-With-R.html;
#https://www.machinelearningplus.com/machine-learning/feature-selection/
#https://www.analyticsvidhya.com/blog/2016/03/select-important-variables-boruta-package/
#https://towardsdatascience.com/boruta-explained-the-way-i-wish-someone-explained-it-to-me-4489d70e154a
#libraries: Boruta, mlbench, randomforest

set.seed(123)
boruta.train <- Boruta(as.factor(class)~., data = MCI_db, doTrace = 2)
print(boruta.train)

# Get significant variable
boruta_signif <- getSelectedAttributes(boruta.train, withTentative = TRUE)
print(boruta_signif)

# Variable Importance Scores
imps <- attStats(boruta.train)
imps2 = imps[imps$decision != 'Rejected', c('meanImp', 'decision')]
head(imps2[order(-imps2$meanImp), ])  # descending sort

# Figure 30: Plot variable importance
plot(boruta.train, cex.axis=.7, las=2, xlab="", main="Variable Importance")



#Feature Selection Method 2: Stepwise forward and backward elimination; caret and random forest libraries
#https://www.machinelearningplus.com/machine-learning/feature-selection/#4stepwiseforwardandbackwardselection
#http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/154-stepwise-regression-essentials-in-r/
```

```r

#Stepwise
set.seed(123)

train.control <- trainControl(method = "cv", number = 10)

step.model <- train(MCI ~., data = MCI_db,
                    method = "leapSeq",
                    tuneGrid = data.frame(nvmax = 1:10),
                    trControl = train.control
)

#summary results
step.model$results

#Dispay model has the lowest RMSE
step.model$bestTune

#Figure 31: Summary showing the optimal set of variables
summary(step.model$finalModel)


#Backwards
set.seed(123)

train.control2 <- trainControl(method = "cv", number = 10)

step.model2 <- train(MCI ~., data = MCI_db,
                     method = "leapBackward",
                     tuneGrid = data.frame(nvmax = 1:10),
                     trControl = train.control
)

#summary results
step.model2$results

#Display model with the lowest RMSE
step.model2$bestTune

#Figure 32: Summary showing the optimal set of variables
summary(step.model2$finalModel)

#Further reduce features for final classification dataset
MCI_Final<-MCI_db[-c(1,3,7,10)]
str(MCI_Final)


#Table 2: check relative proportions of classes; data set quite imbalanced
MCI_prop<-table(MCI_Final$MCI)
MCI_prop
round(100*prop.table(MCI_prop))


#Addressing class imbalance
#Apply SMOTE method to balance the dataset.  Use smote family library
set.seed(123)
smote<-SMOTE(MCI_Final[,-11], MCI_Final$MCI)
smote=smote$data
Smote_prop<-table(smote$MCI)
Smote_prop
round(100*prop.table(Smote_prop))

#write to csv
write.csv(smote,"D:/Ryerson Big Data/CIND820 Big Data Analytics Project/Assignment3/MCI_mod.csv", row.names = FALSE)

```

```r
#Rename dataset then split the data: Train & Test
MCI_mod<-smote
#library(caTools)
set.seed(123)
TrainInd<-sample(1:nrow(MCI_mod), 0.8*nrow(MCI_mod))
Train<-MCI_mod[TrainInd,]
Test<-MCI_mod[-TrainInd,]

write.csv(Train,"D:/Ryerson Big Data/CIND820 Big Data Analytics Project/Assignment3/TrainingData.csv", row.names = FALSE)
write.csv(Test,"D:/Ryerson Big Data/CIND820 Big Data Analytics Project/Assignment3/TestingData.csv", row.names = FALSE)


#Model 1: Decision Tree Model (J48) with 10 fold cross validation; RWeka library
#Follow method as outlined here: https://cran.r-project.org/web/packages/RWeka/RWeka.pdf
#https://rdrr.io/cran/RWeka/man/Weka_control.html

#create training model and evaluate
set.seed(123)
DT <- J48(as.factor(class)~., data = Train, control=Weka_control(M=5))

#10 fold cross validation
EV<-evaluate_Weka_classifier(DT,numFolds = 10)
EV
EV$details

#predict using J48
predDT<-predict(DT, Test, type="class")

#DT confusion matrix
confDT<-table(Test$class, predDT, dnn=c("Actual", "Predicted"))
confDT

#evaluate model
confusionMatrix(as.factor(Test$class), as.factor(predDT))




#Model 2: multivariate logistic regression - same parameters as example
#https://stackoverflow.com/questions/39550118/cross-validation-function-for-logistic-regression-in-r
#https://www.youtube.com/watch?v=fDjKa7yWk1U; nnet library

#set up traint/test &  10 fold cross validation
LR_train<-Train
LR_test<-Test
set.seed(123)
tc <- trainControl(method = "cv", number = 10)

# Training the multinomial model
MN_model <- multinom(class ~ .,
                     data = LR_train,
                     method = 'glm',
                     trControl = tc,
                     family = binomial()
)

# Checking the model
summary(MN_model)

#convert coefficients to odds
exp(coef(MN_model))

#top observations
head(round(fitted(MN_model), 2))

```

```r

#Prediction
# Predicting the values for train dataset
LR_train$Pred <- predict(MN_model, newdata = LR_train, "class")


# Building classification table
tab <- table(LR_train$class, LR_train$Pred)

#confusion matrix for training model
CM_train<-confusionMatrix(tab)
CM_train

#misclassification error
1-sum(diag(tab))/sum(tab)


# Calculating accuracy - sum of diagonal elements divided by total obs
round((sum(diag(tab))/sum(tab))*100,2)

# Predicting the class for test dataset
LR_test$Pred <- predict(MN_model, newdata = LR_test, "class")
# Building classification table
tab2 <- table(LR_test$class, LR_test$Pred)
tab2

CM_mod<-confusionMatrix(tab2)
CM_mod

# Calculating accuracy of predictive model - sum of diagonal elements divided by total obs
round((sum(diag(tab2))/sum(tab2))*100,2)



#Model 3: Naive Bayes with 10 fold cross validation:Balanced Data
#https://www.geeksforgeeks.org/naive-bayes-classifier-in-r-programming/
#https://rpubs.com/maulikpatel/224581
#https://www.analyticsvidhya.com/blog/2021/03/introduction-to-k-fold-cross-validation-in-r/; naive bayes package
NB_train<-Train
NB_test<-Test

set.seed(100)
trctrl <- trainControl(method = "cv", number = 10, savePredictions=TRUE)
nb_fit <- train(as.factor(class) ~., data = NB_train, method = "naive_bayes", trControl=trctrl, tuneLength = 0)
nb_fit


#predict based on the NB Model
y_predNB <- predict(nb_fit, newdata = NB_test)

#NB Confusion Matrix & evaluation
cmNB <- table(NB_test$class, y_predNB, dnn = c("Actual", "Predicted"))
cmNB


confusionMatrix(as.factor(NB_test$class), as.factor(y_predNB))


```