

Final_R_Code

Zoe Lu

2024-12-13

Reading Data

```
raw <- read.csv("/Users/zoelu/Downloads/Crime_Data_from_2010_to_2019_20241126.csv")
```

At glance view of the data

```
summary(raw)
```

```
##      DR_NO      Date.Rptd      DATE.OCC      TIME.OCC
## Min.   : 1208575 Length:2093455 Length:2093455 Min.   :  1
## 1st Qu.:121815922 Class :character Class :character 1st Qu.: 930
## Median :161113993 Mode  :character Mode  :character Median :1430
## Mean   :154145763                      Mean   :1360
## 3rd Qu.:180907498                      3rd Qu.:1900
## Max.   :910220366                      Max.   :2359
##
##      AREA      AREA.NAME      Rpt.Dist.No      Part.1.2
## Min.   : 1.00 Length:2093455 Min.   : 100 Min.   :1.000
## 1st Qu.: 6.00 Class :character 1st Qu.: 636 1st Qu.:1.000
## Median :11.00 Mode  :character Median :1152 Median :1.000
## Mean   :10.88                      Mean   :1134 Mean   :1.441
## 3rd Qu.:16.00                      3rd Qu.:1622 3rd Qu.:2.000
## Max.   :21.00                      Max.   :2199 Max.   :2.000
##
##      Crm.Cd      Crm.Cd.Desc      Mocodes      Vict.Age
## Min.   :110 Length:2093455 Length:2093455 Min.   : -12.00
## 1st Qu.:330 Class :character Class :character 1st Qu.: 19.00
## Median :442 Mode  :character Mode  :character Median : 32.00
## Mean   :507                      Mean   : 31.71
## 3rd Qu.:626                      3rd Qu.: 46.00
## Max.   :956                      Max.   :118.00
##
##      Vict.Sex      Vict.Descent      Premis.Cd      Premis.Desc
## Length:2093455 Length:2093455 Min.   :101.0 Length:2093455
## Class :character Class :character 1st Qu.:102.0 Class :character
## Mode  :character Mode  :character Median :210.0 Mode  :character
```

```
##                               Mean    :309.4
##                               3rd Qu.:501.0
##                               Max.    :971.0
##                               NA's    :40
## Weapon.Used.Cd      Weapon.Desc      Status      Status.Desc
## Min.      :101.0      Length:2093455      Length:2093455      Length:2093455
## 1st Qu.:400.0      Class :character      Class :character      Class :character
## Median :400.0      Mode  :character      Mode  :character      Mode  :character
## Mean    :370.7
## 3rd Qu.:400.0
## Max.    :516.0
## NA's    :1383033
##      Crm.Cd.1      Crm.Cd.2      Crm.Cd.3      Crm.Cd.4
## Min.      :110.0      Min.      :210.0      Min.      : 93.0      Min.      :421.0
## 1st Qu.:330.0      1st Qu.:998.0      1st Qu.:998.0      1st Qu.:998.0
## Median :442.0      Median :998.0      Median :998.0      Median :998.0
## Mean    :506.8      Mean    :947.8      Mean    :973.2      Mean    :966.5
## 3rd Qu.:626.0      3rd Qu.:998.0      3rd Qu.:998.0      3rd Qu.:998.0
## Max.    :999.0      Max.    :999.0      Max.    :999.0      Max.    :999.0
## NA's    :10      NA's    :1951889      NA's    :2089623      NA's    :2093349
##      LOCATION      Cross.Street      LAT      LON
## Length:2093455      Length:2093455      Min.      : 0.00      Min.      : -118.8
## Class :character      Class :character      1st Qu.:34.01      1st Qu.: -118.4
## Mode  :character      Mode  :character      Median :34.06      Median : -118.3
##                               Mean    :34.06      Mean    : -118.3
##                               3rd Qu.:34.17      3rd Qu.: -118.3
##                               Max.    :34.71      Max.    :   0.0
##
```

Subsetting data and changing variable types

```
# Getting rid of Records Number, Location(Street Address of Crime),
# Cross Street(Cross Street of Rounded Address), Latitude, Longitude
raw_subset <- raw[2:24]
```

```
# setting variable types
raw_subset$AREA <- as.factor(raw_subset$AREA)
raw_subset$AREA.NAME <- as.factor(raw_subset$AREA.NAME)
raw_subset$Part.1.2 <- as.factor(raw_subset$Part.1.2)
raw_subset$Vict.Sex <- as.factor(raw_subset$Vict.Sex)
raw_subset$Vict.Descent <- as.factor(raw_subset$Vict.Descent)
raw_subset$Status <- as.factor(raw_subset$Status)
raw_subset$Status.Desc <- as.factor(raw_subset$Status.Desc)
```

```
# at glance view of categorical data
table(raw_subset$Status.Desc)
```

```
##
## Adult Arrest  Adult Other  Invest Cont  Juv Arrest  Juv Other  UNK
##      218880      255425      1598598      15229      5291      32
```

Re-Coding Response:Status of Case

```
# Setting rows marked as UNK(unclear) to NA
raw_subset$Status.Desc[raw_subset$Status.Desc == "UNK"] <- NA

# Creating new Variable with our proposed Binary Outcome Legal Actions vs. No Legal Action(Status Case)
raw_subset$Legal_Action <- raw_subset$Status.Desc
raw_subset$Legal_Action <- as.character(raw_subset$Legal_Action)
raw_subset$Legal_Action <- ifelse(raw_subset$Legal_Action == "Invest Cont", 0, 1)

# counts of response variable
table(raw_subset$Legal_Action)
```

```
##
##          0          1
## 1598598  494825
```

ReCoding Date & Coding Difference in Report Time

```
raw_subset$date_report <- raw_subset$Date.Rptd
raw_subset$date_report <- as.POSIXct(raw_subset$date_report, format = "%m/%d/%Y %I:%M:%S %p")
raw_subset$date_report <- as.Date(raw_subset$date_report)

# Occurrence Date
raw_subset$date_occur <- raw_subset$DATE.OCC
raw_subset$date_occur <- as.POSIXct(raw_subset$date_occur, format = "%m/%d/%Y %I:%M:%S %p")
raw_subset$date_occur <- as.Date(raw_subset$date_occur)

#Creating New Column: Difference Between Report vs Occurrence
raw_subset$date_occur_report_difference <- as.numeric(difftime(raw_subset$date_report,
                                                                raw_subset$date_occur,
                                                                units = "days"))
```

Categorizing Time Occurred

```
# Convert military time to string
raw_subset$military_time <- raw_subset$TIME.OCC

#Time Ranges:
# Morning 5 am to 12 pm (noon)
# Afternoon 12 pm to 5 pm.
# Evening 5 pm to 9 pm.
# Night 9 pm to 4 am.
military_times_str <- sprintf("%04d", raw_subset$military_time)

hours <- as.integer(substr(military_times_str, 1, 2))
```

```
categories <- ifelse(hours >= 5 & hours < 12, "Morning",
                    ifelse(hours >= 12 & hours < 17, "Afternoon",
                            ifelse(hours >= 17 & hours < 21, "Evening", "Night")))

raw_subset$time_occur_cat <- categories
```

Creating dictionaries to store descriptions of unique values and frequencies of each category

```
summary_tables_top20 <- function(key_input,value_input) {
  dict <- setNames(key_input,value_input)
  df <- data.frame(
    key = names(dict),
    value = unname(dict)
  ) %>%
  dplyr::group_by(key, value) %>%
  dplyr::summarize(frequency = n(), .groups = "drop") %>%
  arrange(desc(frequency))
  df_name <- paste0(gsub(".*\\$", "",deparse(substitute(value_input))))
  assign(df_name, df, envir = .GlobalEnv)

  #Frequency Counts
  row_counts <- c(10,15, 20, 25, 50)
  total_rows <- nrow(df)

  # printing out top categories
  cat("Cumulative sums of frequencies for the top categories:\n")
  for (n in row_counts) {
    if (n <= total_rows) {
      cat(paste0("Top ", n, " categories: ", sum(df$frequency[1:n]), "\n"))
    } else {
      cat(paste0("Top ", n, " categories: Not enough categories (only ", total_rows, " categories a
    }
  }
  return(head(df,20))
}

# view summaries of top categories
summary_tables_top20(raw_subset$AREA.NAME, raw_subset$AREA)
```

```
## Cumulative sums of frequencies for the top categories:
## Top 10 categories: 1123992
## Top 15 categories: 1583968
## Top 20 categories: 2019153
## Top 25 categories: Not enough categories (only 21 categories available).
## Top 50 categories: Not enough categories (only 21 categories available).
```

```
## # A tibble: 20 x 3
##   key    value    frequency
```

```
##      <chr> <fct>          <int>
##  1 12      77th Street    146824
##  2  3      Southwest      141044
##  3 14      Pacific        113389
##  4 15      N Hollywood    110365
##  5 18      Southeast      108188
##  6 13      Newton         104064
##  7  6      Hollywood      101551
##  8 11      Northeast      99985
##  9  9      Van Nuys       99669
## 10  1      Central        98913
## 11 19      Mission        94188
## 12  5      Harbor         92650
## 13 10      West Valley    91437
## 14  8      West LA        91171
## 15  7      Wilshire       90530
## 16  2      Rampart        89524
## 17 21      Topanga        89389
## 18 17      Devonshire     87873
## 19 20      Olympic        87398
## 20  4      Hollenbeck     81001
```

```
summary_tables_top20(raw_subset$Crm.Cd.Desc,raw_subset$Crm.Cd)
```

```
## Cumulative sums of frequencies for the top categories:
## Top 10 categories: 1325082
## Top 15 categories: 1671296
## Top 20 categories: 1786035
## Top 25 categories: 1875445
## Top 50 categories: 2037624
```

```
## # A tibble: 20 x 3
##   key   value                                frequency
##   <chr> <chr>                                <int>
##  1 624   BATTERY - SIMPLE ASSAULT                185209
##  2 330   BURGLARY FROM VEHICLE                  161295
##  3 510   VEHICLE - STOLEN                      160481
##  4 440   THEFT PLAIN - PETTY ($950 & UNDER)     148064
##  5 310   BURGLARY                              142573
##  6 354   THEFT OF IDENTITY                     121517
##  7 626   INTIMATE PARTNER - SIMPLE ASSAULT      112375
##  8 740   VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS) 109646
##  9 230   ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT      93679
## 10 420   THEFT FROM MOTOR VEHICLE - PETTY ($950 & UNDER)     90243
## 11 745   VANDALISM - MISDEAMEANOR ($399 OR UNDER)           87833
## 12 210   ROBBERY                                    84186
## 13 341   THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LIVESTK,PROD 70565
## 14 930   CRIMINAL THREATS - NO WEAPON DISPLAYED             54645
## 15 442   SHOPLIFTING - PETTY THEFT ($950 & UNDER)           48985
## 16 331   THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND OVER) 31103
## 17 888   TRESPASSING                                    22526
## 18 649   DOCUMENT FORGERY / STOLEN FELONY              21357
## 19 956   LETTERS, LEWD - TELEPHONE CALLS, LEWD             20264
## 20 946   OTHER MISCELLANEOUS CRIME                  19489
```

```
summary_tables_top20(raw_subset$Premis.Desc, raw_subset$Premis.Cd)
```

```
## Cumulative sums of frequencies for the top categories:
```

```
## Top 10 categories: 1684905
## Top 15 categories: 1781415
## Top 20 categories: 1845674
## Top 25 categories: 1890018
## Top 50 categories: 2008647
```

```
## # A tibble: 20 x 3
```

##	key	value	frequency
##	<chr>	<chr>	<int>
## 1	101	STREET	473543
## 2	501	SINGLE FAMILY DWELLING	416635
## 3	502	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	256929
## 4	108	PARKING LOT	150225
## 5	102	SIDEWALK	104771
## 6	203	OTHER BUSINESS	95144
## 7	122	VEHICLE, PASSENGER/TRUCK	78105
## 8	104	DRIVEWAY	42445
## 9	707	GARAGE/CARPORT	37700
## 10	210	RESTAURANT/FAST FOOD	29408
## 11	404	DEPARTMENT STORE	27105
## 12	402	MARKET	21727
## 13	123	PARKING UNDERGROUND/BUILDING	16573
## 14	406	OTHER STORE	16406
## 15	109	PARK/PLAYGROUND	14699
## 16	103	ALLEY	14173
## 17	121	YARD (RESIDENTIAL/BUSINESS)	14104
## 18	710	OTHER PREMISE	12927
## 19	721	HIGH SCHOOL	12370
## 20	403	DRUG STORE	10685

```
summary_tables_top20(raw_subset$Weapon.Desc, raw_subset$Weapon.Used.Cd)
```

```
## Cumulative sums of frequencies for the top categories:
```

```
## Top 10 categories: 2014130
## Top 15 categories: 2038301
## Top 20 categories: 2058118
## Top 25 categories: 2071307
## Top 50 categories: 2091121
```

```
## # A tibble: 20 x 3
```

##	key	value	frequency
##	<chr>	<chr>	<int>
## 1	<NA>	"	1383033
## 2	400	"STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)"	428182
## 3	511	"VERBAL THREAT"	58308
## 4	500	"UNKNOWN WEAPON/OTHER WEAPON"	58003
## 5	102	"HAND GUN"	34857
## 6	109	"SEMI-AUTOMATIC PISTOL"	13405
## 7	200	"KNIFE WITH BLADE 6INCHES OR LESS"	13250

##	8	207	"OTHER KNIFE"	9799
##	9	106	"UNKNOWN FIREARM"	7978
##	10	307	"VEHICLE"	7315
##	11	101	"REVOLVER"	5409
##	12	212	"BOTTLE"	5113
##	13	306	"ROCK/THROWN OBJECT"	4658
##	14	308	"STICK"	4639
##	15	204	"FOLDING KNIFE"	4352
##	16	304	"CLUB/BAT"	4337
##	17	512	"MACE/PEPPER SPRAY"	4225
##	18	302	"BLUNT INSTRUMENT"	4044
##	19	205	"KITCHEN KNIFE"	3845
##	20	113	"SIMULATED GUN"	3366

Weapons NA Recode

```
#Creating new category None instead of NA for no weapon used
raw_subset$Weapon.Used.Cd <- as.character(raw_subset$Weapon.Used.Cd)
raw_subset$Weapon.Used.Cd[is.na(raw_subset$Weapon.Used.Cd) == T] <- "None"
```

Subsetting Columns needed readying data for cleaning

```
columns_to_subset <- c("AREA", "Rpt.Dist.No", "Part.1.2", "Crm.Cd", "Mocodes",
  "Vict.Age", "Vict.Sex", "Vict.Descent", "Premis.Cd",
  "Weapon.Used.Cd", "Status", "Legal_Action",
  "date_occur_report_difference", "time_occur_cat")

subset1 <- raw_subset[, columns_to_subset]

#Only including rows whose Crm.Cd is in top 50
crime_top_50_string_vec <- Crm.Cd$key[1:50]
filtered_subset2 <- subset1[subset1$Crm.Cd %in% crime_top_50_string_vec, ]

#Only including rows whose crime took place in Premise in top 50
premise_top_50_string_vec <- Premis.Cd$key[1:50]
filtered_subset3 <- filtered_subset2[filtered_subset2$Premis.Cd %in% premise_top_50_string_vec, ]

#Only including rows if weapon Used in top 10
summary_tables_top20(raw_subset$Weapon.Desc, raw_subset$Weapon.Used.Cd)

## Cumulative sums of frequencies for the top categories:
## Top 10 categories: 2014130
## Top 15 categories: 2038301
## Top 20 categories: 2058118
## Top 25 categories: 2071307
## Top 50 categories: 2091121

## # A tibble: 20 x 3
```

##	key	value	frequency
##	<chr>	<chr>	<int>
## 1	None	"	1383033
## 2	400	"STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)"	428182
## 3	511	"VERBAL THREAT"	58308
## 4	500	"UNKNOWN WEAPON/OTHER WEAPON"	58003
## 5	102	"HAND GUN"	34857
## 6	109	"SEMI-AUTOMATIC PISTOL"	13405
## 7	200	"KNIFE WITH BLADE 6INCHES OR LESS"	13250
## 8	207	"OTHER KNIFE"	9799
## 9	106	"UNKNOWN FIREARM"	7978
## 10	307	"VEHICLE"	7315
## 11	101	"REVOLVER"	5409
## 12	212	"BOTTLE"	5113
## 13	306	"ROCK/THROWN OBJECT"	4658
## 14	308	"STICK"	4639
## 15	204	"FOLDING KNIFE"	4352
## 16	304	"CLUB/BAT"	4337
## 17	512	"MACE/PEPPER SPRAY"	4225
## 18	302	"BLUNT INSTRUMENT"	4044
## 19	205	"KITCHEN KNIFE"	3845
## 20	113	"SIMULATED GUN"	3366

```

weapon_top_10_string_vec <- Weapon.Used.Cd$key[1:10]
filtered_subset4 <- filtered_subset3[filtered_subset3$Weapon.Used.Cd %in% weapon_top_10_string_vec, ]

#Dropping Mocodes
filtered_subset5 <- filtered_subset4[, !(colnames(filtered_subset4) %in% "Mocodes")]

#Changing Column types
filtered_subset5$Rpt.Dist.No <- as.factor(filtered_subset5$Rpt.Dist.No)
filtered_subset5$Crm.Cd <- as.factor(filtered_subset5$Crm.Cd)
filtered_subset5$Premis.Cd <- as.factor(filtered_subset5$Premis.Cd)
filtered_subset5$Weapon.Used.Cd <- as.factor(filtered_subset5$Weapon.Used.Cd)
filtered_subset5$Legal_Action <- as.factor(filtered_subset5$Legal_Action)
filtered_subset5$time_occur_cat <- as.factor(filtered_subset5$time_occur_cat)

```

Cleaning Data: Changing Columns

```

# Identified and cleaning Negative Ages, one age of 118, and sex:X
filtered_subset6 <- filtered_subset5[filtered_subset5$Vict.Age > 0,]
filtered_subset7 <- filtered_subset6[filtered_subset6$Vict.Age <= 100, ]
filtered_subset8 <- filtered_subset7[filtered_subset7$Vict.Sex == "F" | filtered_subset7$Vict.Sex == "M", ]

#Identified and cleaning Null Race and "-" Race entering Race
filtered_subset9 <- filtered_subset8[filtered_subset8$Vict.Descent != "-", ]
filtered_subset9$Vict.Descent.Description <- ifelse(
  filtered_subset9$Vict.Descent == "A",
  "Other Asian", ifelse(filtered_subset9$Vict.Descent == "B",
  "Black", ifelse(filtered_subset9$Vict.Descent == "C",
  "Chinese", ifelse(filtered_subset9$Vict.Descent == "D",

```



```

"Cambodian",ifelse(filtered_subset9$Vict.Descent == "F",
"Filipino",ifelse(filtered_subset9$Vict.Descent == "G",
"Guamanian",ifelse(filtered_subset9$Vict.Descent == "H",
"Hispanic/Latin/Mexican",
ifelse(filtered_subset9$Vict.Descent == "I",
"American Indian/Alaskan Native",
ifelse(filtered_subset9$Vict.Descent == "J","Japanese",
ifelse(filtered_subset9$Vict.Descent == "K","Korean",
ifelse(filtered_subset9$Vict.Descent == "L","Laotian",
ifelse(filtered_subset9$Vict.Descent == "O","Other",
ifelse(filtered_subset9$Vict.Descent == "P",
"Pacific Islander",
ifelse(filtered_subset9$Vict.Descent == "S","Samoaan",
ifelse(filtered_subset9$Vict.Descent == "U",
"Hawaiian",ifelse(filtered_subset9$Vict.Descent == "V",
"Vietnamese",ifelse(filtered_subset9$Vict.Descent == "W",
"White",ifelse(filtered_subset9$Vict.Descent == "X",NA,
ifelse(filtered_subset9$Vict.Descent == "Z", "Asian Indian", NA)))))))))))))))))

filtered_subset10 <- filtered_subset9[, !(colnames(filtered_subset9) %in% "Vict.Descent")]
filtered_subset10$Vict.Descent.Description <- as.factor(filtered_subset10$Vict.Descent.Description)

#Only Including those with sex M or F
filtered_subset11 <- filtered_subset10[filtered_subset10$Vict.Sex == "M" | filtered_subset10$Vict.Sex == "F",]

#Removing Sub-Areas as redundant to Geographic Areas
filtered_subset12 <- filtered_subset11[, !(colnames(filtered_subset11) %in% "Rpt.Dist.No")]

#Removing Status as outcome coded into Legal Action
filtered_subset13 <- filtered_subset12[, !(colnames(filtered_subset12) %in% "Status")]

#Omitting Nulls
filtered_subset14 <- na.omit(filtered_subset13)

clean_data <- filtered_subset14

```

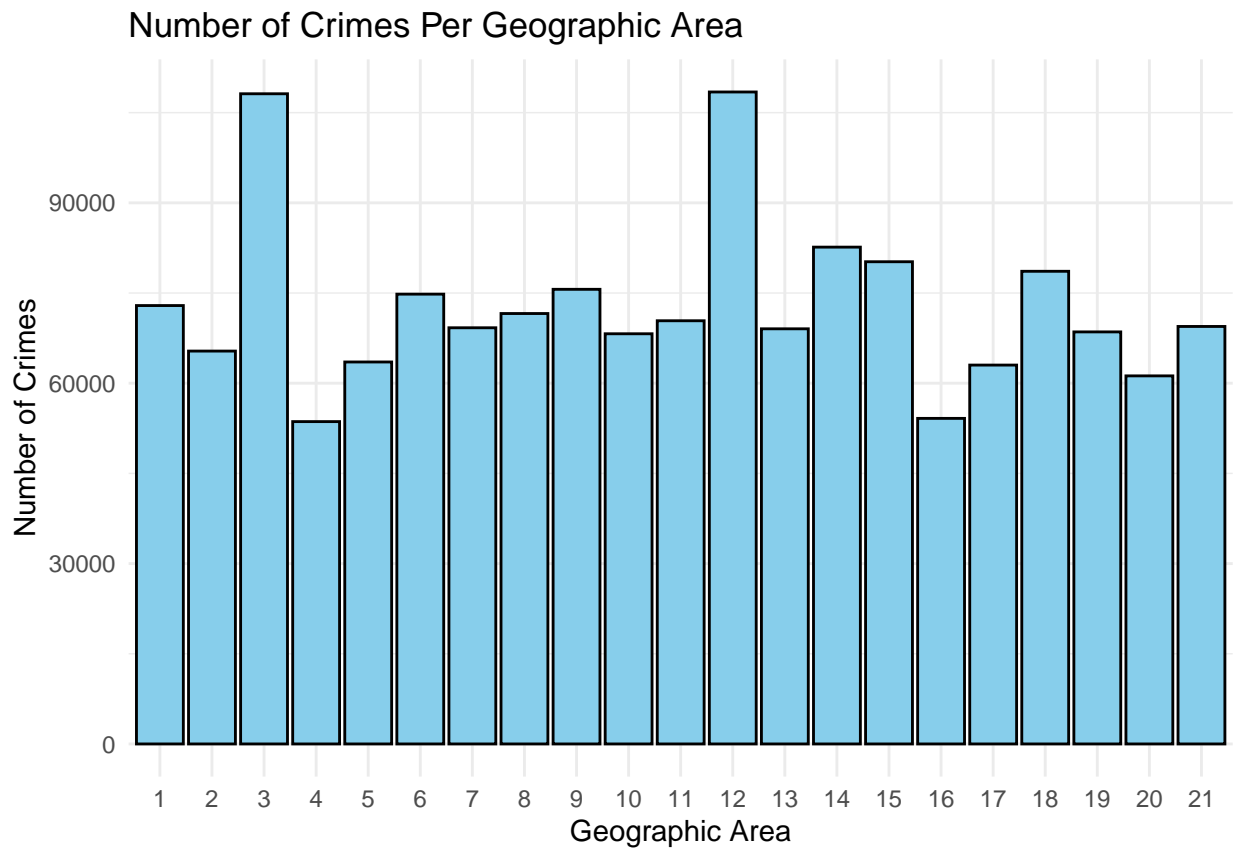
EDA: Count Plots for Some Categorical

```

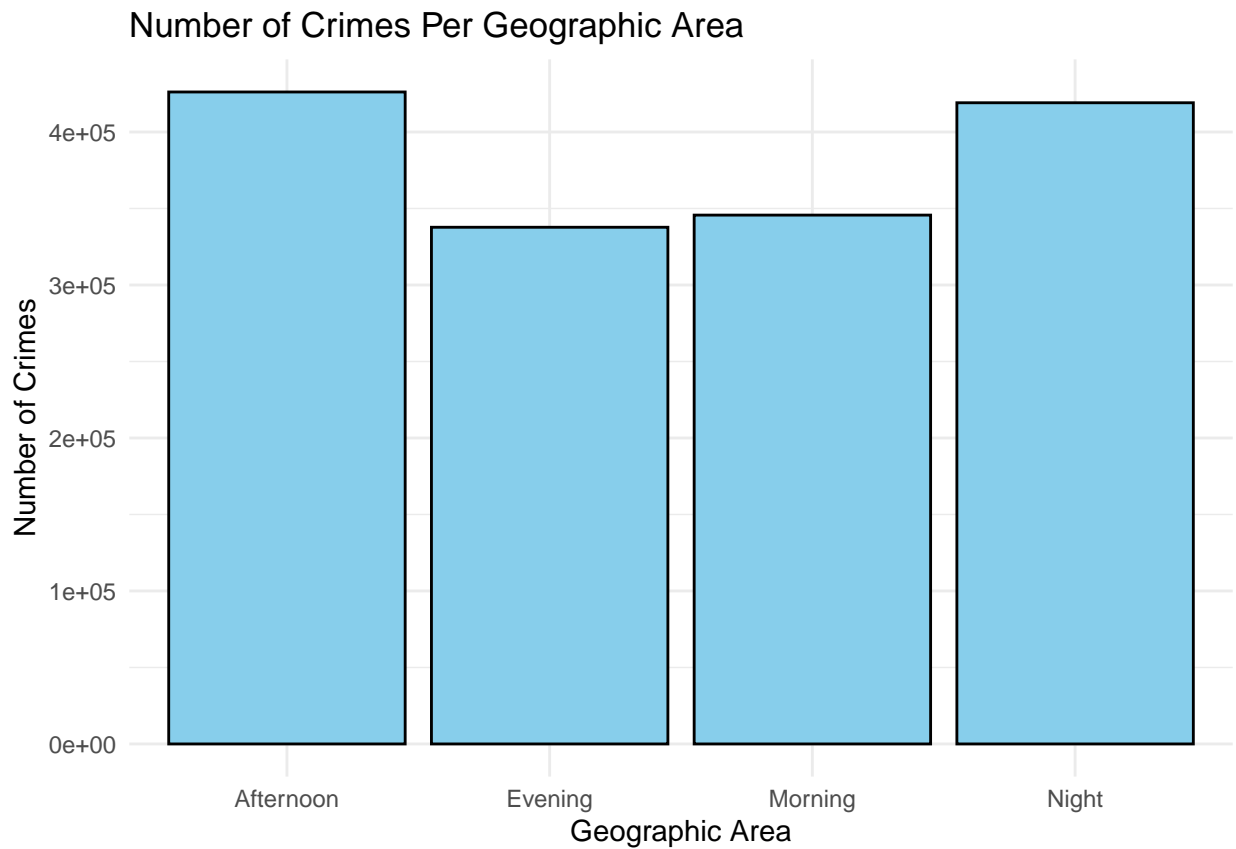
# function to create plots
barchart_fcn <- function(variable, title, x, y) {
  ggplot(clean_data, aes(x = .data[[variable]]), title, x, y) +
    geom_bar(color = "black", fill = "skyblue") +
    labs(title = title, x = x, y = y) +
    theme_minimal()
}

# area barchart
barchart_fcn("AREA",
  title = "Number of Crimes Per Geographic Area",
  x = "Geographic Area", y = "Number of Crimes")

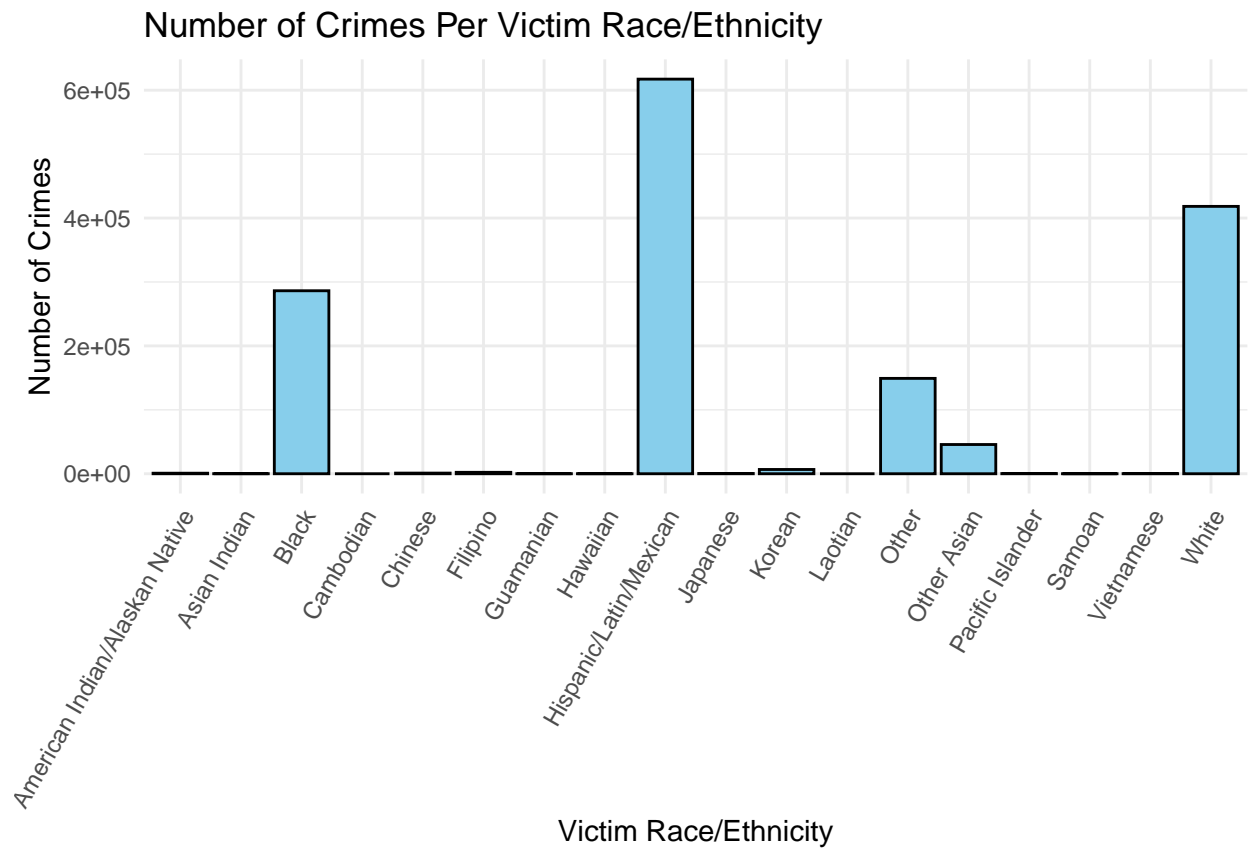
```



```
#Number of Crimes Per time_occur_cat  
barchart_fcn("time_occur_cat",  
             title = "Number of Crimes Per Geographic Area",  
             x = "Geographic Area", y = "Number of Crimes")
```



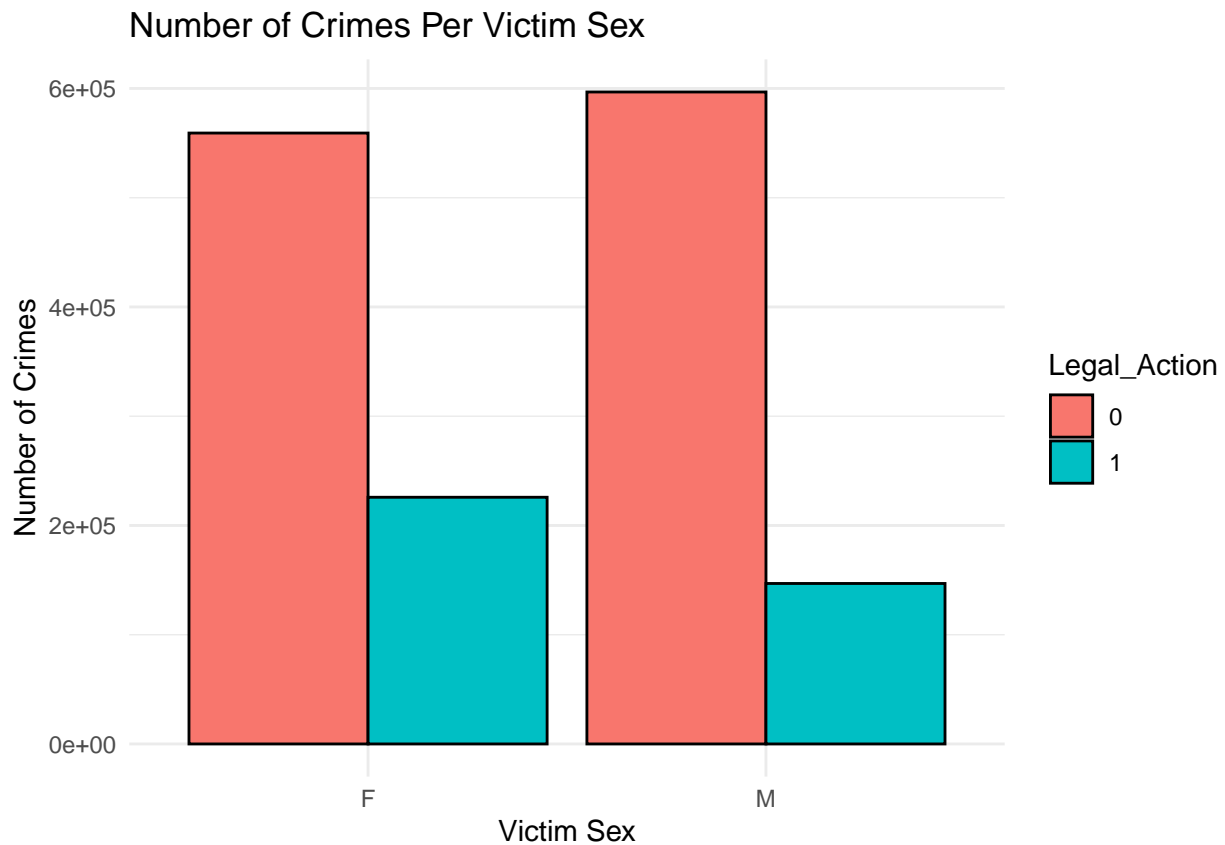
```
#Number of Crimes Per Vict.Descent.Description
ggplot(clean_data, aes(x = Vict.Descent.Description)) +
  geom_bar(color = "black", fill = "skyblue") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  labs(title = "Number of Crimes Per Victim Race/Ethnicity", x = "Victim Race/Ethnicity", y = "Number
```



```
# legal action barchart
barchart_fcn("Legal_Action",
  title = "Count of Ongoing Investitions vs Closed",
  x = "Case Status", y = "Number of Cases")
```

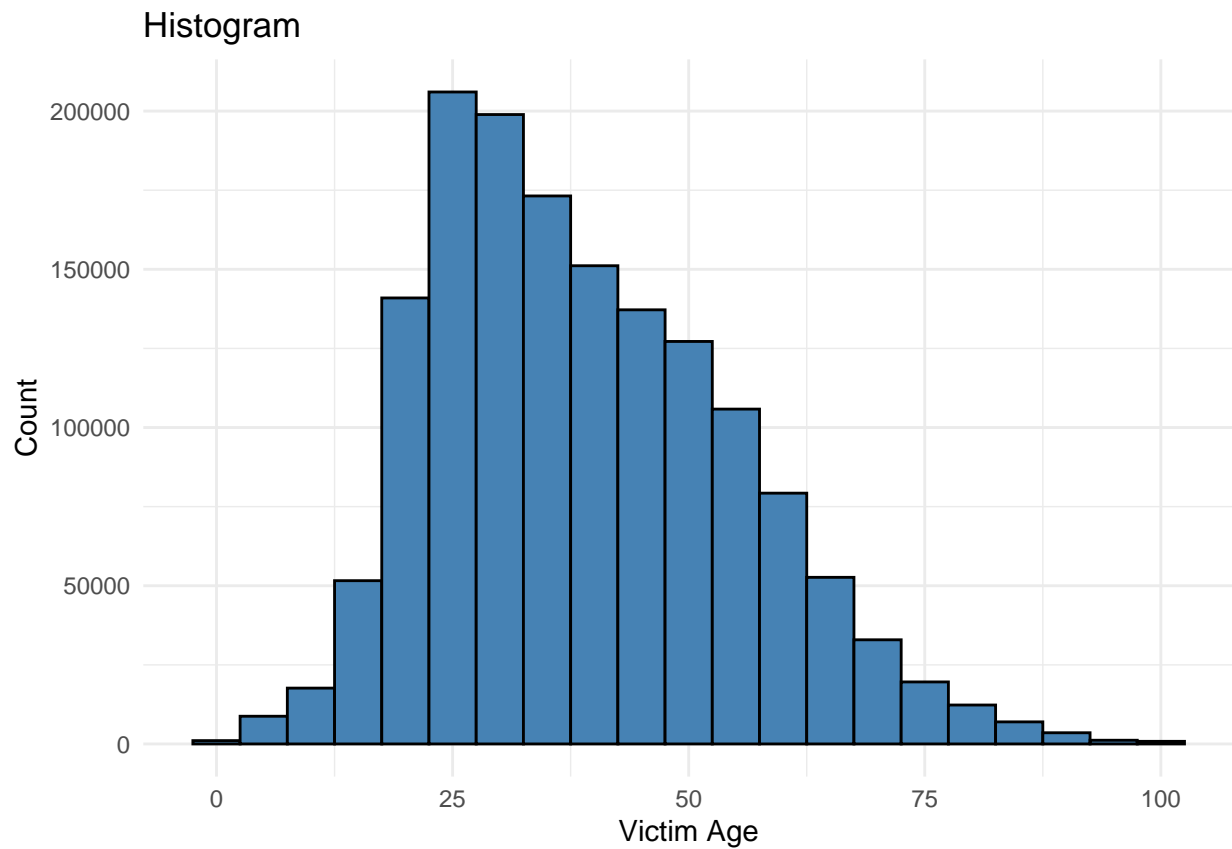


```
# Number of Crimes Per Victim Sex by Case Status  
ggplot(clean_data, aes(x = Vict.Sex, fill = Legal_Action)) +  
  geom_bar(color = "black", position = "dodge") +  
  labs(title = "Number of Crimes Per Victim Sex", x = "Victim Sex", y = "Number of Crimes") +  
  theme_minimal()
```

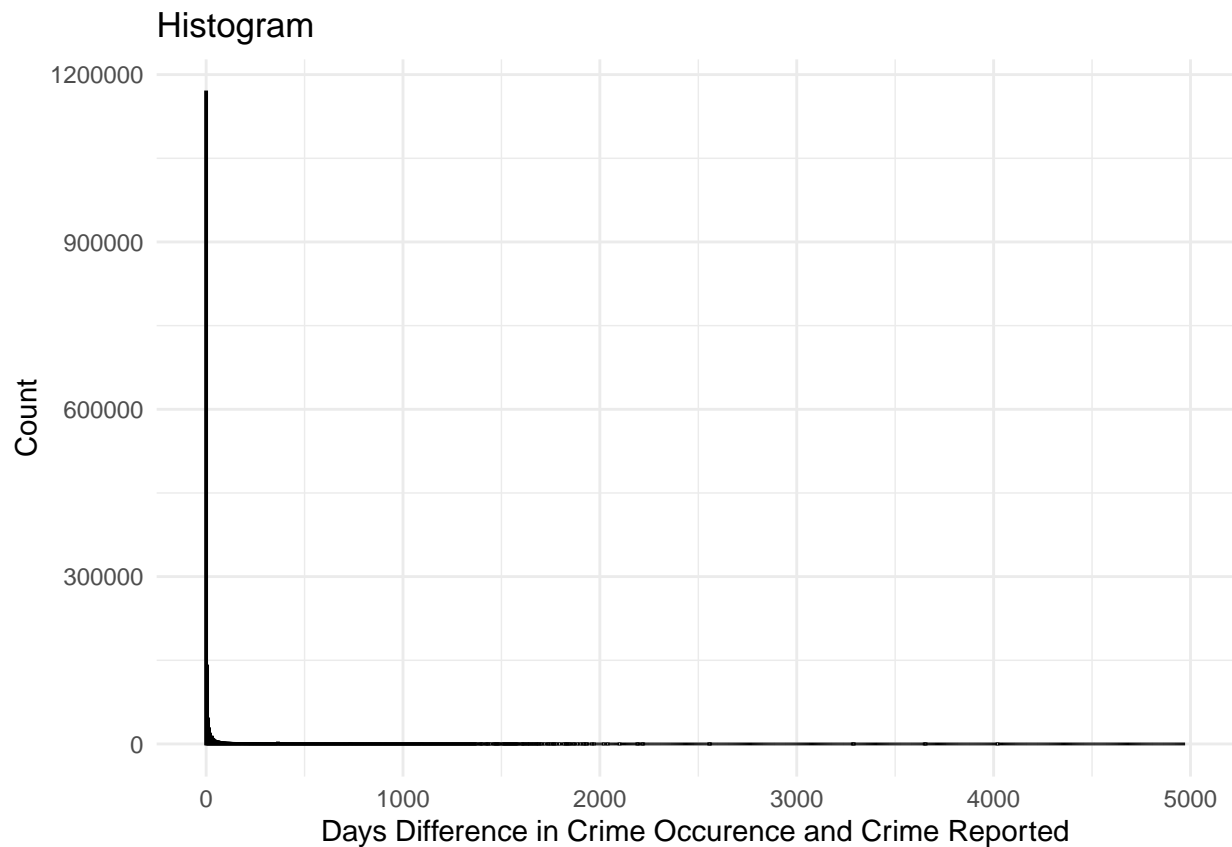


EDA: Box Plots/Histograms for Continuous

```
# Victim Age histogram  
ggplot(clean_data, aes(x = Vict.Age)) +  
  geom_histogram(binwidth = 5, fill = "steelblue", color = "black") +  
  labs(title = "Histogram", x = "Victim Age", y = "Count") +  
  theme_minimal()
```

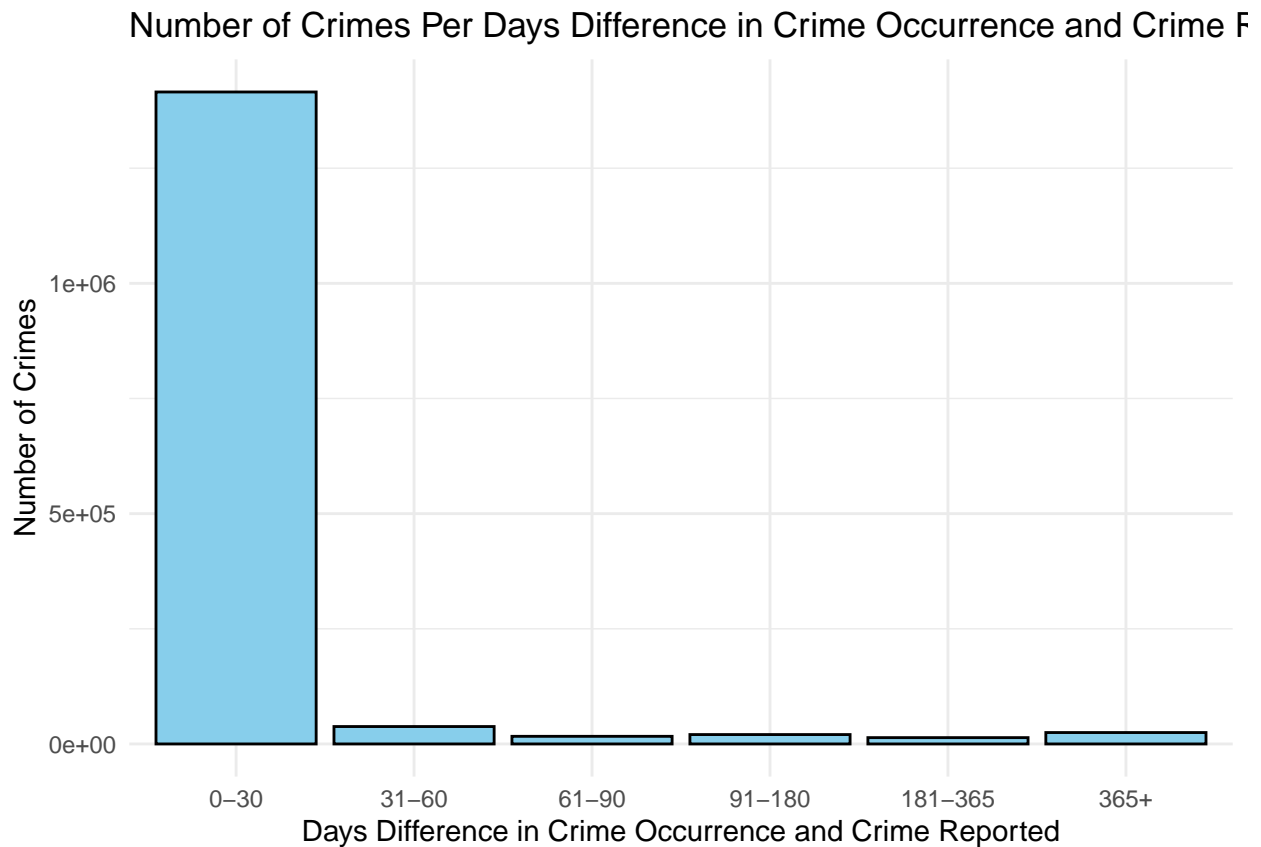


```
# Days Difference in Crime Occurrence and Crime Reported histogram
ggplot(clean_data, aes(x = date_occur_report_difference)) +
  geom_histogram(binwidth = 5, fill = "steelblue", color = "black") +
  labs(title = "Histogram", x = "Days Difference in Crime Occurrence and Crime Reported", y = "Count")
  theme_minimal()
```



```
#EDA Reveals date_occur_report_difference should be categorized
clean_data$date_occur_report_difference <- cut(clean_data$date_occur_report_difference,
  breaks = c(0, 30, 60, 90, 180, 365, Inf),
  labels = c("0-30", "31-60", "61-90", "91-180", "181-365", "365+"),
  right = FALSE)

#Number of Crimes Per Days Difference in Crime Occurrence and Crime Reported Count Plot
ggplot(clean_data, aes(x = date_occur_report_difference)) +
  geom_bar(position = position_dodge(width = 0.4), color = "black", fill = "skyblue") +
  theme_minimal() +
  labs(title = "Number of Crimes Per Days Difference in Crime Occurrence and Crime Reported Count Plot",
    x = "Days Difference in Crime Occurrence and Crime Reported", y = "Number of Crimes")
```

Subsetting bootstraps defining bootstrap function

```
set.seed(123)
# Randomly shuffling the data and dividing into train/test
clean_data <- clean_data[sample(nrow(clean_data)), ]

clean_data_indexes <- sample(2, nrow(clean_data),
                             replace = TRUE, prob = c(0.8,0.2))
clean_data_train <- clean_data[clean_data_indexes==1,]
clean_data_test <- clean_data[clean_data_indexes==2,]

#Subsetting Train into 31 datasets
clean_data_train$Group <- sample(1:31, size = nrow(clean_data_train), replace = T)
df_subsets_train <- split(clean_data_train, clean_data_train$Group)

#Subsetting Test into 31 datasets
clean_data_test$Group <- sample(1:31, size = nrow(clean_data_test), replace = T)
clean_data_test_list <- split(clean_data_test, clean_data_test$Group)

# Define oversampling function
oversample_data <- function(data) {
```

```

    return(ovun.sample(Legal_Action ~ ., data = data, p=0.5)$data)
}

```

Intializing Parallel and Bootstrapping

```

unregister_dopar <- function() {
  env <- foreach:::foreachGlobals
  rm(list=ls(name=env), pos=env)
}
unregister_dopar()

#initializing parallel processing
num_cores <- detectCores() - 1
cl <- makePSOCKcluster(num_cores)
registerDoParallel(cl)

oversampled_data_list <- foreach(data = df_subsets_train, .packages = c("ROSE")) %dopar% {
  oversample_data(data)
}

#Calling 31st dataset
extra_clean_train <- oversampled_data_list[[31]]

```

Running Logistic Regression

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
```

```
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
```

```
##
```

```
##      cov, smooth, var
```

```

trControl_log <- trainControl(method = 'repeatedcv',
                              number = 5,
                              repeats = 5,
                              search = 'random')

```

```

logit_gridsearch <- caret::train( Legal_Action ~ AREA + Part.1.2 + Crm.Cd + Vict.Age +
                                   Vict.Sex + Premis.Cd + Weapon.Used.Cd +
                                   date_occur_report_difference + time_occur_cat + Vict.Descent.Description
                                   data = extra_clean_train,
                                   method = 'glmnet',

```

```

trControl = trControl_log,
family = 'binomial',
metric = 'Accuracy')

#Optimal Parameters
logit_gridsearch$bestTune

##          alpha    lambda
## 1 0.00126282 6.382388

#Creating empty lists
accuracy_vector_logit <- numeric(length(1:30))
conf_mat_list_logit <- vector("list",length(1:30))
auc_list_logit <- numeric(length(1:30))

results_logit <- foreach (i = 1:30,
  .packages = c("caret", "dplyr", "pROC")) %dopar% {
  logit_alpha <- logit_gridsearch$bestTune$alpha
  logit_lambda <- logit_gridsearch$bestTune$lambda

  # Training the Random Forest model 30 times w/optimal parameters
  logit_model <- caret::train(
    Legal_Action ~ AREA + Part.1.2 + Crm.Cd + Vict.Age + Vict.Sex + Premis.C,
    data = oversampled_data_list[[i]],
    method = "glmnet",
    trControl = trainControl(method = "none"),
    tuneGrid = expand.grid(alpha = logit_alpha, lambda = logit_lambda))

  #Confusion Matrix of final model predicting Resolved Case
  predictions_logit <- predict(logit_model, newdata = clean_data_test_list[[i]],
  confusion_mat_logit <- confusionMatrix(predictions_logit, clean_data_test_list[[i]])

  accuracy_vector_logit[i] <- confusion_mat_logit$overall['Accuracy']

  #Predictions with probabilities
  logit_auc_pred <- predict(logit_model, clean_data_test_list[[i]], 'prob')
  logit_auc_pred <- cbind(logit_auc_pred[,1], as.character(clean_data_test_list[[i]]$V2))
  logit_auc_pred <- as.data.frame(logit_auc_pred)
  logit_auc_pred$V2 <- as.factor(logit_auc_pred$V2)
  logit_auc_pred$V1 <- as.numeric(logit_auc_pred$V1)

  #AUC
  logit_auc_roc <- roc(logit_auc_pred$V2, logit_auc_pred$V1)
  auc_list_logit <- logit_auc_roc$auc[1]

  list(
    confusion_matrix = confusion_mat_logit,
    accuracy = confusion_mat_logit$overall['Accuracy'],
    auc = auc_list_logit
  )
}

for (i in 1:length(results_logit)){
  conf_mat_list_logit[[i]] <- results_logit[[i]]$confusion_matrix

```

```

    accuracy_vector_logit[i] <- results_logit[[i]]$accuracy
    auc_list_logit[[i]] <- results_logit[[i]]$auc
}

```

```

accuracy_vector_logit <- unlist(accuracy_vector_logit)

```

```

# mean accuracy for logistic regression
mean_logit_vec <- mean(accuracy_vector_logit)
mean_logit_vec

```

```

## [1] 0.7526045

```

```

cat("Creating 95% Confidence Interval for Accuracy of Model
    predicting if case was resolved\n")

```

```

## Creating 95% Confidence Interval for Accuracy of Model
##     predicting if case was resolved

```

```

#standard error
std_error_logit <- sd(accuracy_vector_logit) / sqrt(30)

```

```

#critical t value for 95% CI
critical_value_logit <- qt(0.975, df = 30 - 1)

```

```

#confidence interval
lower_ci_logit <- mean_logit_vec - (critical_value_logit * std_error_logit)
upper_ci_logit <- mean_logit_vec + (critical_value_logit * std_error_logit)

```

```

# 95% CI

```

```

cat("95% Confidence Interval Predicting if Case Resolved: [", lower_ci_logit, ", ", upper_ci_logit, "]\n")

```

```

## 95% Confidence Interval Predicting if Case Resolved: [ 0.7503001 , 0.7549089 ]

```

```

#Finding Index of accuracy value closest to mean
closest_index_logit <- which.min(abs(accuracy_vector_logit - mean_logit_vec))

```

```

#Confusion Matrix of Model closest to mean accuracy
print(conf_mat_list_logit[closest_index_logit])

```

```

## [[1]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 5752 636
##           1 1849 1788
##
##           Accuracy : 0.7521
##           95% CI : (0.7435, 0.7605)
##           No Information Rate : 0.7582

```

```
##      P-Value [Acc > NIR] : 0.924
##
##              Kappa : 0.4224
##
## Mcnemar's Test P-Value : <2e-16
##
##      Sensitivity : 0.7567
##      Specificity : 0.7376
##      Pos Pred Value : 0.9004
##      Neg Pred Value : 0.4916
##      Prevalence : 0.7582
##      Detection Rate : 0.5738
##      Detection Prevalence : 0.6372
##      Balanced Accuracy : 0.7472
##
##      'Positive' Class : 0
##
```

```
#AUC of Model closest to mean
print(auc_list_logit[closest_index_logit])
```

```
## [1] 0.8076606
```

Running Neural Network

```
library(NeuralNetTools)
library(nnet)
library(NeuralSens)
###      ###
###Neural Net###
###      ###

tr_control_nnet <- trainControl(
  method = "cv",
  number = 5,
  allowParallel = TRUE)

#Neural Net
fit.nnet <- caret::train(Legal_Action ~ AREA + Part.1.2 + Crm.Cd + Vict.Age + Vict.Sex + Premis.Cd + We
  data = extra_clean_train,
  method = "nnet",
  trControl = tr_control_nnet,
  tuneGrid = expand.grid(size=c(1:5),
                          decay=c(0,0.1,0.01)),
  skip = TRUE)

## # weights: 965
## initial value 97620.367076
## iter 10 value 30020.875482
## iter 20 value 24520.733259
```

```
## iter 30 value 22048.473116
## iter 40 value 20849.769117
## iter 50 value 20004.517781
## iter 60 value 19761.371945
## iter 70 value 19559.912288
## iter 80 value 19417.940840
## iter 90 value 19339.611826
## iter 100 value 19291.465254
## final value 19291.465254
## stopped after 100 iterations
```

```
pred.nnet <- predict(fit.nnet,clean_data_test_list[[31]])
confusionMatrix(table(clean_data_test_list[[31]]$Legal_Action, pred.nnet))
```

```
## Confusion Matrix and Statistics
##
##      pred.nnet
##      0      1
## 0 5471 1873
## 1  541 1859
##
##              Accuracy : 0.7523
##              95% CI : (0.7436, 0.7608)
##      No Information Rate : 0.617
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.4378
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##              Sensitivity : 0.9100
##              Specificity : 0.4981
##              Pos Pred Value : 0.7450
##              Neg Pred Value : 0.7746
##              Prevalence : 0.6170
##              Detection Rate : 0.5615
##      Detection Prevalence : 0.7537
##              Balanced Accuracy : 0.7041
##
##      'Positive' Class : 0
##
```

```
#Optimal Parameters
fit.nnet$bestTune
```

```
##      size decay
## 14      5 0.01
```

```
#Creating empty lists
accuracy_vector_nnet <- numeric(length(1:30))
conf_mat_list_nnet <- vector("list",length(1:30))
variable_importance_list_nnet <- vector("list",length(1:30))
```

```

tr_control2_nnet <- trainControl(
  method = "none",
  allowParallel = TRUE)

results_nnet <- foreach (i = 1:30,
  .packages = c("caret", "dplyr", "nnet")) %dopar% {
  num_neurons <- fit.nnet$bestTune$size
  decay_rate <- fit.nnet$bestTune$decay

  # Training the neural net model 30 times w/1 neuron
  nnet_model <- caret::train(
    Legal_Action ~ AREA + Part.1.2 + Crm.Cd + Vict.Age + Vict.Sex + Premis.C
    data = oversampled_data_list[[i]],
    method = "nnet",
    tuneGrid = expand.grid(size=num_neurons,decay=decay_rate),
    trControl = tr_control2_nnet,
    skip = TRUE
  )

  #Confusion Matrix of final model predicting Resolved Case
  predictions_nnet <- predict(nnet_model, newdata = clean_data_test_list[[i]])
  confusion_mat_nnet <- confusionMatrix(predictions_nnet, clean_data_test_list

  accuracy_vector_nnet[i] <- confusion_mat_nnet$overall['Accuracy']

  var_importance_nnet <- varImp(nnet_model)
  variable_importance_list_nnet[[i]] <- var_importance_nnet

  list(
    confusion_matrix = confusion_mat_nnet,
    accuracy = confusion_mat_nnet$overall['Accuracy'],
    variable_importance = var_importance_nnet
  )
}

for (i in 1:length(results_nnet)){
  conf_mat_list_nnet[[i]] <- results_nnet[[i]]$confusion_matrix
  accuracy_vector_nnet[i] <- results_nnet[[i]]$accuracy
  variable_importance_list_nnet[[i]] <- results_nnet[[i]]$variable_importance
}
accuracy_vector_nnet <- unlist(accuracy_vector_nnet)

# mean accuracy for neural network
mean_nnet_vec <- mean(accuracy_vector_nnet)
mean_nnet_vec

```

```
## [1] 0.7613686
```

```

cat("Creating 95% Confidence Interval for Accuracy of Model
predicting if case was resolved")

```

```
## Creating 95% Confidence Interval for Accuracy of Model
##     predicting if case was resolved
```

```
#standard error
```

```
std_error_nnet <- sd(accuracy_vector_nnet) / sqrt(30)
```

```
#critical t value for 95% CI
```

```
critical_value_nnet <- qt(0.975, df = 30 - 1)
```

```
#confidence interval
```

```
lower_ci_nnet <- mean_nnet_vec - (critical_value_nnet * std_error_nnet)
```

```
upper_ci_nnet <- mean_nnet_vec + (critical_value_nnet * std_error_nnet)
```

```
# 95% CI
```

```
cat("95% Confidence Interval Predicting if Case Resolved: [", lower_ci_nnet, ", ", upper_ci_nnet, "]\n")
```

```
## 95% Confidence Interval Predicting if Case Resolved: [ 0.7591919 , 0.7635454 ]
```

```
#Finding Index of accuracy value closest to mean
```

```
closest_index_nnet <- which.min(abs(accuracy_vector_nnet - mean_nnet_vec))
```

```
#Confusion Matrix of Model closest to mean accuracy
```

```
print(conf_mat_list_nnet[closest_index_nnet])
```

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

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction    0    1
```

```
##           0 5707  579
```

```
##           1 1771 1771
```

```
##
```

```
##           Accuracy : 0.7609
```

```
##           95% CI : (0.7523, 0.7693)
```

```
##           No Information Rate : 0.7609
```

```
##           P-Value [Acc > NIR] : 0.5055
```

```
##
```

```
##           Kappa : 0.4402
```

```
##
```

```
##           McNemar's Test P-Value : <2e-16
```

```
##
```

```
##           Sensitivity : 0.7632
```

```
##           Specificity : 0.7536
```

```
##           Pos Pred Value : 0.9079
```

```
##           Neg Pred Value : 0.5000
```

```
##           Prevalence : 0.7609
```

```
##           Detection Rate : 0.5807
```

```
##           Detection Prevalence : 0.6396
```

```
##           Balanced Accuracy : 0.7584
```

```
##
```

```
##           'Positive' Class : 0
```

```
##
```


Running Naive Bayes

```
library(naivebayes)

## naivebayes 1.0.0 loaded

## For more information please visit:

## https://majkamichal.github.io/naivebayes/

extra_clean_train_nb <- extra_clean_train
extra_clean_train_nb <- extra_clean_train_nb[,-12]

nb_grid <- expand.grid(usekernel = c(TRUE, FALSE),
                      laplace = c(0, 0.5, 1),
                      adjust = c(0.75, 1, 1.25, 1.5))

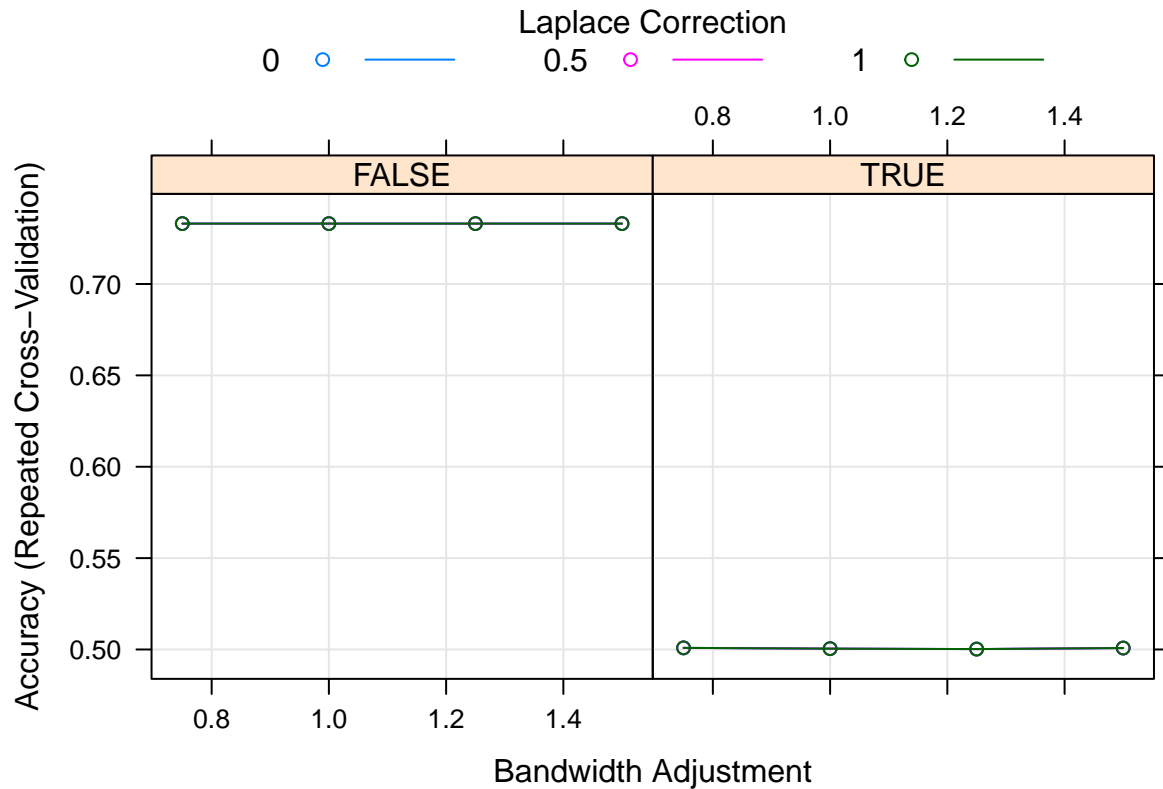
tr_control_nb <- trainControl(
  method = "repeatedcv",
  number = 10,
  repeats = 3,
  allowParallel = TRUE
)

nb_fit <- caret::train(
  Legal_Action ~ .,
  data = extra_clean_train_nb,
  method = "naive_bayes",
  trControl = tr_control_nb,
  tuneGrid = nb_grid
)

#Checking model optimal parameters
nb_fit$finalModel$tuneValue

##   laplace usekernel adjust
## 1      0      FALSE  0.75

#Visualizing tuning process
nb_fit_plot <- plot(nb_fit)
nb_fit_plot
```



```
# Performing classification
predictions_nb1 <- predict(nb_fit, newdata = clean_data_test_list[[31]])
confusion_mat_nb1 <- confusionMatrix(predictions_nb1, clean_data_test_list[[31]]$Legal_Action)

#Creating accuracy and confusion matrices vectors
accuracy_vector_nb <- numeric(length(1:30))
conf_mat_list_nb <- vector("list",length(1:30))

results_nb <- foreach (i = 1:30,
  .packages = c("caret", "dplyr", "naivebayes")) %dopar% {
  #Optimal Parameters
  laplace_param <- nb_fit$finalModel$tuneValue$laplace
  usekernel_param <- nb_fit$finalModel$tuneValue$usekernel
  adjust_param <- nb_fit$finalModel$tuneValue$adjust

  # Training the Naive Bayes model 30 times w/optimal parameters
  nb_model <- caret::train(
    Legal_Action ~ AREA + Part.1.2 + Crm.Cd + Vict.Age + Vict.Sex + Premis.
    data = oversampled_data_list[[i]],
    method = "naive_bayes",
    trControl = trainControl(method = "none"),
    tuneGrid = expand.grid(.laplace = laplace_param, .usekernel = usekernel.
  )

  #Confusion Matrix of final model predicting Resolved Case
  predictions_nb <- predict(nb_model, newdata = clean_data_test_list[[i]])
  confusion_mat_nb <- confusionMatrix(predictions_nb, clean_data_test_list[[i]]$Legal_Action)
```

```

        accuracy_vector_nb[i] <- confusion_mat_nb$overall['Accuracy']

        list(confusion_matrix = confusion_mat_nb,
              accuracy = confusion_mat_nb$overall['Accuracy']
            )
      }

for (i in 1:length(results_nb)){
  conf_mat_list_nb[[i]] <- results_nb[[i]]$confusion_matrix
  accuracy_vector_nb[i] <- results_nb[[i]]$accuracy
}
accuracy_vector_nb <- unlist(accuracy_vector_nb)

# mean accuracy for naive bayes
mean_nb_vec <- mean(accuracy_vector_nb)
mean_nb_vec

```

```
## [1] 0.6759287
```

```

cat("Creating 95% Confidence Interval for Accuracy of Model
    predicting if case was resolved\n")

```

```

## Creating 95% Confidence Interval for Accuracy of Model
##    predicting if case was resolved

```

```

#standard error
std_error_nb <- sd(accuracy_vector_nb) / sqrt(30)

#critical t value for 95% CI
critical_value_nb <- qt(0.975, df = 30 - 1)

#confidence interval
lower_ci_nb <- mean_nb_vec - (critical_value_nb * std_error_nb)
upper_ci_nb <- mean_nb_vec + (critical_value_nb * std_error_nb)

# 95% CI
cat("95% Confidence Interval Predicting if Case Resolved: [", lower_ci_nb, ", ", upper_ci_nb, "]\n")

```

```
## 95% Confidence Interval Predicting if Case Resolved: [ 0.662266 , 0.6895914 ]
```

```

#Finding Index of accuracy value closest to mean
closest_index_nb <- which.min(abs(accuracy_vector_nb - mean_nb_vec))

#Confusion Matrix of Model closest to mean accuracy
print(conf_mat_list_nb[closest_index_nb])

```

```

## [[1]]
## Confusion Matrix and Statistics
##

```

```

##           Reference
## Prediction    0    1
##           0 4701  414
##           1 2790 1981
##
##           Accuracy : 0.6759
##           95% CI : (0.6666, 0.6851)
##           No Information Rate : 0.7577
##           P-Value [Acc > NIR] : 1
##
##           Kappa : 0.34
##
## Mcnemar's Test P-Value : <2e-16
##
##           Sensitivity : 0.6276
##           Specificity : 0.8271
##           Pos Pred Value : 0.9191
##           Neg Pred Value : 0.4152
##           Prevalence : 0.7577
##           Detection Rate : 0.4755
##           Detection Prevalence : 0.5174
##           Balanced Accuracy : 0.7273
##
##           'Positive' Class : 0
##

```