NYPD Shoting Incident Data Report

DTSA 5301 Data Science as a Field

MS Data Science, University of Colorado Boulder

2024-08-20

Setup Knit Options

echo = true will display code chunks in the output

Load Libraries

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(conflicted)
library(lubridate)
library(caret)
```

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

```
library(xgboost)
library(pROC)
```

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

```
library(PRROC)
library(MLmetrics)
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library(car)
```

```
## Loading required package: carData
```

```
library(smotefamily)
library(ROSE)
```

```
## Loaded ROSE 0.0-4
```

Read Dataset

Download NYPD Shooting csv dataset and store in a data frame

```
url <- "https://data.cityofnewyork.us/api/views/833y-fsy8/rows.csv?accessType=DOWNLOAD"
data <- read_csv(url)
```

```
## Rows: 28562 Columns: 21
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (12): OCCUR_DATE, BORO, LOC_OF_OCCUR_DESC, LOC_CLASSFCTN_DESC, LOCATION...
## dbl   (7): INCIDENT_KEY, PRECINCT, JURISDICTION_CODE, X_COORD_CD, Y_COORD_CD...
## lgl   (1): STATISTICAL_MURDER_FLAG
## time  (1): OCCUR_TIME
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Inspect Data

Display raw data structure

```
# display structure
str(data)
```

```
## spc_tbl_ [28,562 x 21] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ INCIDENT_KEY           : num [1:28562] 2.45e+08 2.48e+08 8.50e+07 2.03e+08 2.71e+07 ...
##  $ OCCUR_DATE             : chr [1:28562] "05/05/2022" "07/04/2022" "05/27/2012" "09/24/2019" ...
##  $ OCCUR_TIME             : 'hms' num [1:28562] 00:10:00 22:20:00 19:35:00 21:00:00 ...
```

```
##   ..- attr(*, "units")= chr "secs"
##  $ BORO                   : chr [1:28562] "MANHATTAN" "BRONX" "QUEENS" "BRONX" ...
##  $ LOC_OF_OCCUR_DESC      : chr [1:28562] "INSIDE" "OUTSIDE" NA NA ...
##  $ PRECINCT               : num [1:28562] 14 48 103 42 83 23 113 77 48 49 ...
##  $ JURISDICTION_CODE      : num [1:28562] 0 0 0 0 0 2 0 0 0 0 ...
##  $ LOC_CLASSFCTN_DESC     : chr [1:28562] "COMMERCIAL" "STREET" NA NA ...
##  $ LOCATION_DESC          : chr [1:28562] "VIDEO STORE" "(null)" NA NA ...
##  $ STATISTICAL_MURDER_FLAG: logi [1:28562] TRUE TRUE FALSE FALSE FALSE FALSE ...
##  $ PERP_AGE_GROUP         : chr [1:28562] "25-44" "(null)" NA "25-44" ...
##  $ PERP_SEX               : chr [1:28562] "M" "(null)" NA "M" ...
##  $ PERP_RACE              : chr [1:28562] "BLACK" "(null)" NA "UNKNOWN" ...
##  $ VIC_AGE_GROUP          : chr [1:28562] "25-44" "18-24" "18-24" "25-44" ...
##  $ VIC_SEX                : chr [1:28562] "M" "M" "M" "M" ...
##  $ VIC_RACE               : chr [1:28562] "BLACK" "BLACK" "BLACK" "BLACK" ...
##  $ X_COORD_CD             : num [1:28562] 986050 1016802 1048632 1014493 1009149 ...
##  $ Y_COORD_CD             : num [1:28562] 214231 250581 198262 242565 190105 ...
##  $ Latitude               : num [1:28562] 40.8 40.9 40.7 40.8 40.7 ...
##  $ Longitude              : num [1:28562] -74 -73.9 -73.8 -73.9 -73.9 ...
##  $ Lon_Lat                : chr [1:28562] "POINT (-73.9935 40.754692)" "POINT (-73.88233 40.854402)"
##  - attr(*, "spec")=
##   .. cols(
##   ..   INCIDENT_KEY = col_double(),
##   ..   OCCUR_DATE = col_character(),
##   ..   OCCUR_TIME = col_time(format = ""),
##   ..   BORO = col_character(),
##   ..   LOC_OF_OCCUR_DESC = col_character(),
##   ..   PRECINCT = col_double(),
##   ..   JURISDICTION_CODE = col_double(),
##   ..   LOC_CLASSFCTN_DESC = col_character(),
##   ..   LOCATION_DESC = col_character(),
##   ..   STATISTICAL_MURDER_FLAG = col_logical(),
##   ..   PERP_AGE_GROUP = col_character(),
##   ..   PERP_SEX = col_character(),
##   ..   PERP_RACE = col_character(),
##   ..   VIC_AGE_GROUP = col_character(),
##   ..   VIC_SEX = col_character(),
##   ..   VIC_RACE = col_character(),
##   ..   X_COORD_CD = col_double(),
##   ..   Y_COORD_CD = col_double(),
##   ..   Latitude = col_double(),
##   ..   Longitude = col_double(),
##   ..   Lon_Lat = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Here we can see the first 5 rows of the data and can page to the right to see all the columns.

```
# display first 5 rows
head(data)
```

```
## # A tibble: 6 x 21
##   INCIDENT_KEY OCCUR_DATE OCCUR_TIME BORO      LOC_OF_OCCUR_DESC PRECINCT
##          <dbl> <chr>      <time>     <chr>     <chr>                <dbl>
## 1    244608249 05/05/2022 00:10      MANHATTAN INSIDE                  14
```

```
## 2    247542571 07/04/2022 22:20      BRONX     OUTSIDE                    48
## 3     84967535 05/27/2012 19:35      QUEENS    <NA>                      103
## 4    202853370 09/24/2019 21:00      BRONX     <NA>                       42
## 5     27078636 02/25/2007 21:00      BROOKLYN  <NA>                       83
## 6    230311078 07/01/2021 23:07      MANHATTAN <NA>                       23
## # i 15 more variables: JURISDICTION_CODE <dbl>, LOC_CLASSFCTN_DESC <chr>,
## #   LOCATION_DESC <chr>, STATISTICAL_MURDER_FLAG <lgl>, PERP_AGE_GROUP <chr>,
## #   PERP_SEX <chr>, PERP_RACE <chr>, VIC_AGE_GROUP <chr>, VIC_SEX <chr>,
## #   VIC_RACE <chr>, X_COORD_CD <dbl>, Y_COORD_CD <dbl>, Latitude <dbl>,
## #   Longitude <dbl>, Lon_Lat <chr>
```

This will display the summary statistics of the data. If the data is numerical it will produce the mean, median, min, max, and quartiles. If the data is categorical it will produce the counts of each category.

```r
# display summary statistics
summary(data)
```

```
##   INCIDENT_KEY        OCCUR_DATE         OCCUR_TIME           BORO          
##  Min.   :  9953245   Length:28562       Length:28562      Length:28562      
##  1st Qu.: 65439914   Class :character   Class1:hms        Class :character  
##  Median : 92711254   Mode  :character   Class2:difftime   Mode  :character  
##  Mean   :127405824                      Mode  :numeric                      
##  3rd Qu.:203131993                                                          
##  Max.   :279758069                                                          
##                                                                             
##  LOC_OF_OCCUR_DESC     PRECINCT     JURISDICTION_CODE LOC_CLASSFCTN_DESC
##  Length:28562       Min.   :  1.0   Min.   :0.0000    Length:28562      
##  Class :character   1st Qu.: 44.0   1st Qu.:0.0000    Class :character  
##  Mode  :character   Median : 67.0   Median :0.0000    Mode  :character  
##                     Mean   : 65.5   Mean   :0.3219                      
##                     3rd Qu.: 81.0   3rd Qu.:0.0000                      
##                     Max.   :123.0   Max.   :2.0000                      
##                                     NA's   :2                           
##  LOCATION_DESC      STATISTICAL_MURDER_FLAG PERP_AGE_GROUP    
##  Length:28562       Mode :logical           Length:28562      
##  Class :character   FALSE:23036             Class :character  
##  Mode  :character   TRUE :5526              Mode  :character  
##                                                               
##                                                               
##                                                               
##                                                               
##    PERP_SEX          PERP_RACE         VIC_AGE_GROUP        VIC_SEX         
##  Length:28562       Length:28562       Length:28562       Length:28562      
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##                                                                             
##                                                                             
##                                                                             
##                                                                             
##    VIC_RACE           X_COORD_CD        Y_COORD_CD        Latitude    
##  Length:28562       Min.   : 914928   Min.   :125757   Min.   :40.51  
##  Class :character   1st Qu.:1000068   1st Qu.:182912   1st Qu.:40.67  
##  Mode  :character   Median :1007772   Median :194901   Median :40.70  
```

```
##                       Mean   :1009424   Mean   :208380   Mean   :40.74
##                       3rd Qu.:1016807   3rd Qu.:239814   3rd Qu.:40.82
##                       Max.   :1066815   Max.   :271128   Max.   :40.91
##                                                          NA's   :59
##    Longitude         Lon_Lat
##  Min.   :-74.25   Length:28562
##  1st Qu.:-73.94   Class :character
##  Median :-73.92   Mode  :character
##  Mean   :-73.91
##  3rd Qu.:-73.88
##  Max.   :-73.70
##  NA's   :59
```

Above we can see three of the columns are lowercase and the rest are uppercase. So we now can rename those three columns to uppercase for consistency.

```r
# rename columns to uppercase for consistency
colnames(data)[colnames(data) == "Latitude"] <- "LATITUDE"
colnames(data)[colnames(data) == "Longitude"] <- "LONGITUDE"
colnames(data)[colnames(data) == "Lon_Lat"] <- "LON_LAT"

# display first 5 rows
head(data)
```

```
## # A tibble: 6 x 21
##   INCIDENT_KEY OCCUR_DATE OCCUR_TIME BORO      LOC_OF_OCCUR_DESC PRECINCT
##          <dbl> <chr>      <time>     <chr>     <chr>                <dbl>
## 1    244608249 05/05/2022 00:10      MANHATTAN INSIDE                  14
## 2    247542571 07/04/2022 22:20      BRONX     OUTSIDE                 48
## 3     84967535 05/27/2012 19:35      QUEENS    <NA>                   103
## 4    202853370 09/24/2019 21:00      BRONX     <NA>                    42
## 5     27078636 02/25/2007 21:00      BROOKLYN  <NA>                    83
## 6    230311078 07/01/2021 23:07      MANHATTAN <NA>                    23
## # i 15 more variables: JURISDICTION_CODE <dbl>, LOC_CLASSFCTN_DESC <chr>,
## #   LOCATION_DESC <chr>, STATISTICAL_MURDER_FLAG <lgl>, PERP_AGE_GROUP <chr>,
## #   PERP_SEX <chr>, PERP_RACE <chr>, VIC_AGE_GROUP <chr>, VIC_SEX <chr>,
## #   VIC_RACE <chr>, X_COORD_CD <dbl>, Y_COORD_CD <dbl>, LATITUDE <dbl>,
## #   LONGITUDE <dbl>, LON_LAT <chr>
```

Missing Values

This is checking for missing values in the data, of special type NA.

```r
# display counts of missing values
colSums(is.na(data))
```

```
##            INCIDENT_KEY              OCCUR_DATE              OCCUR_TIME
##                       0                       0                       0
##                    BORO       LOC_OF_OCCUR_DESC                PRECINCT
##                       0                   25596                       0
##       JURISDICTION_CODE      LOC_CLASSFCTN_DESC           LOCATION_DESC
```

```
##                       2                   25596                   14977
## STATISTICAL_MURDER_FLAG          PERP_AGE_GROUP                PERP_SEX
##                       0                    9344                    9310
##               PERP_RACE           VIC_AGE_GROUP                 VIC_SEX
##                    9310                       0                       0
##                VIC_RACE              X_COORD_CD              Y_COORD_CD
##                       0                       0                       0
##                LATITUDE               LONGITUDE                 LON_LAT
##                      59                      59                      59
```

Above we can see several columns have a large number of missing counts. Next we will display the percentage of missing values in each column to understand the ratio of missing values, rather than just the counts.

```
# display percentage of missing values
round(colMeans(is.na(data)), 2)
```

```
##            INCIDENT_KEY              OCCUR_DATE              OCCUR_TIME
##                    0.00                    0.00                    0.00
##                    BORO       LOC_OF_OCCUR_DESC                PRECINCT
##                    0.00                    0.90                    0.00
##       JURISDICTION_CODE      LOC_CLASSFCTN_DESC           LOCATION_DESC
##                    0.00                    0.90                    0.52
## STATISTICAL_MURDER_FLAG          PERP_AGE_GROUP                PERP_SEX
##                    0.00                    0.33                    0.33
##               PERP_RACE           VIC_AGE_GROUP                 VIC_SEX
##                    0.33                    0.00                    0.00
##                VIC_RACE              X_COORD_CD              Y_COORD_CD
##                    0.00                    0.00                    0.00
##                LATITUDE               LONGITUDE                 LON_LAT
##                    0.00                    0.00                    0.00
```

We can see that the columns LOC_OF_OCCUR_DESC, LOC_CLASSFCTN_DESC, LOCATION_DESC, PERP_AGE_GROUP, PERP_SEX, and PERP_RACE all have a high percentage of missing values. We will need a strategy to handle these missing values soon.

Here we will display the counts of each category in the categorical features.

```
# print counts of categorical features
printCountsCategoricalFeatures = function(data, feature) {
    print(feature)
    print(table(addNA(data[[feature]]), useNA = "ifany"))
    cat("\n")
}
```

```
# print counts of categorical features, with missing values
printCountsCategoricalFeatures(data, "LOC_OF_OCCUR_DESC")
```

```
## [1] "LOC_OF_OCCUR_DESC"
##
##   INSIDE OUTSIDE    <NA>
##      460    2506   25596
```

```r
printCountsCategoricalFeatures(data, "JURISDICTION_CODE")
```

```
## [1] "JURISDICTION_CODE"
##
##     0     1     2  <NA>
## 23923    81  4556     2
```

```r
printCountsCategoricalFeatures(data, "LOC_CLASSFCTN_DESC")
```

```
## [1] "LOC_CLASSFCTN_DESC"
##
##      (null)  COMMERCIAL    DWELLING     HOUSING       OTHER PARKING LOT
##           2         208         243         460          59          15
##  PLAYGROUND      STREET     TRANSIT     VEHICLE        <NA>
##          41        1886          23          29       25596
```

```r
printCountsCategoricalFeatures(data, "LOCATION_DESC")
```

```
## [1] "LOCATION_DESC"
##
##                   (null)                      ATM                     BANK
##                     1711                        1                        3
##           BAR/NIGHT CLUB        BEAUTY/NAIL SALON              CANDY STORE
##                      668                      119                        7
##              CHAIN STORE               CHECK CASH         CLOTHING BOUTIQUE
##                        7                        1                       14
##          COMMERCIAL BLDG               DEPT STORE           DOCTOR/DENTIST
##                      304                        9                        1
##               DRUG STORE     DRY CLEANER/LAUNDRY        FACTORY/WAREHOUSE
##                       14                       32                        8
##                FAST FOOD              GAS STATION           GROCERY/BODEGA
##                      130                       74                      750
##      GYM/FITNESS FACILITY                 HOSPITAL              HOTEL/MOTEL
##                        4                       77                       35
##            JEWELRY STORE             LIQUOR STORE             LOAN COMPANY
##                       14                       42                        1
##   MULTI DWELL - APT BUILD MULTI DWELL - PUBLIC HOUS                     NONE
##                     2964                     5007                      175
##         PHOTO/COPY STORE                PVT HOUSE        RESTAURANT/DINER
##                        1                      983                      212
##                   SCHOOL               SHOE STORE           SMALL MERCHANT
##                        1                       10                       44
## SOCIAL CLUB/POLICY LOCATI         STORAGE FACILITY       STORE UNCLASSIFIED
##                       73                        1                       37
##              SUPERMARKET           TELECOMM. STORE            VARIETY STORE
##                       21                       11                       11
##              VIDEO STORE                     <NA>
##                        8                    14977
```

```r
printCountsCategoricalFeatures(data, "PERP_AGE_GROUP")
```

```
## [1] "PERP_AGE_GROUP"
##
##  (null)     <18    1020    1028   18-24     224   25-44   45-64     65+     940
##    1141    1682       1       1    6438       1    6041     699      65       1
## UNKNOWN    <NA>
##    3148    9344
```

```r
printCountsCategoricalFeatures(data, "PERP_SEX")
```

```
## [1] "PERP_SEX"
##
## (null)      F      M      U   <NA>
##   1141    444  16168   1499   9310
```

```r
printCountsCategoricalFeatures(data, "PERP_RACE")
```

```
## [1] "PERP_RACE"
##
##                         (null) AMERICAN INDIAN/ALASKAN NATIVE
##                           1141                              2
##       ASIAN / PACIFIC ISLANDER                          BLACK
##                            169                          11903
##                 BLACK HISPANIC                        UNKNOWN
##                           1392                           1837
##                          WHITE                 WHITE HISPANIC
##                            298                           2510
##                           <NA>
##                           9310
```

```r
printCountsCategoricalFeatures(data, "BORO")
```

```
## [1] "BORO"
##
##         BRONX      BROOKLYN     MANHATTAN        QUEENS STATEN ISLAND
##          8376         11346          3762          4271           807
##          <NA>
##             0
```

```r
printCountsCategoricalFeatures(data, "PRECINCT")
```

```
## [1] "PRECINCT"
##
##    1    5    6    7    9   10   13   14   17   18   19   20   22   23   24   25
##   25   67   28  120  114   74   61   61   10   38   24   43    1  505  113  494
##   26   28   30   32   33   34   40   41   42   43   44   45   46   47   48   49
##  157  353  234  663  242  335  947  519  890  796 1076  195  972 1006  841  368
##   50   52   60   61   62   63   66   67   68   69   70   71   72   73   75   76
##  162  604  383  157   72  292   53 1259   36  484  479  595  117 1500 1628  179
##   77   78   79   81   83   84   88   90   94  100  101  102  103  104  105  106
##  821   65 1045  821  520  131  294  328   87  178  502  229  605  108  488  233
##  107  108  109  110  111  112  113  114  115  120  121  122  123 <NA>
##  105   75  123  174   12   23  834  397  185  597  114   63   33    0
```

```r
printCountsCategoricalFeatures(data, "STATISTICAL_MURDER_FLAG")
```

```
## [1] "STATISTICAL_MURDER_FLAG"
##
## FALSE  TRUE  <NA>
## 23036  5526     0
```

```r
printCountsCategoricalFeatures(data, "VIC_AGE_GROUP")
```

```
## [1] "VIC_AGE_GROUP"
##
##     <18    1022   18-24   25-44   45-64     65+ UNKNOWN    <NA>
##    2954       1   10384   12973    1981     205      64       0
```

```r
printCountsCategoricalFeatures(data, "VIC_SEX")
```

```
## [1] "VIC_SEX"
##
##     F     M     U  <NA>
##  2760 25790    12     0
```

```r
printCountsCategoricalFeatures(data, "VIC_RACE")
```

```
## [1] "VIC_RACE"
##
## AMERICAN INDIAN/ALASKAN NATIVE       ASIAN / PACIFIC ISLANDER
##                             11                            440
##                          BLACK                 BLACK HISPANIC
##                          20235                           2795
##                        UNKNOWN                          WHITE
##                             70                            728
##                 WHITE HISPANIC                           <NA>
##                           4283                              0
```

Above we can see which categorical variables have the least/most categories and how evenly distributed they may be. We also see that there are a few values of "(null)" which is a character string value coming from the data set, and not the datatype NA.

Impute Categorical Features

Next we will impute (fill in) the missing values for the categorical features. This function will replace all NA datatypes with the character value of "UNKNOWN" for a new category. This is done so the values are not missing which would prevent the models from running and force us to remove the records from the training.

```r
cleanCategoricalFeature = function(feature, outlier_list = c()) {
  outlier_found <- length(outlier_list > 0) & feature %in% outlier_list
  return(replace(feature, is.na(feature) | outlier_found == TRUE, "UNKNOWN"))
}
```

```r
# replace missing values with "Unknown" for categorical features
data$LOC_OF_OCCUR_DESC <- cleanCategoricalFeature(data$LOC_OF_OCCUR_DESC)
data$JURISDICTION_CODE <- cleanCategoricalFeature(data$JURISDICTION_CODE)
data$LOC_CLASSFCTN_DESC <- cleanCategoricalFeature(data$LOC_CLASSFCTN_DESC, c("(null)"))
data$LOCATION_DESC <- cleanCategoricalFeature(data$LOCATION_DESC, c("(null)"))
data$PERP_AGE_GROUP <- cleanCategoricalFeature(data$PERP_AGE_GROUP, c("(null)", "1020", "1028", "224",
data$PERP_SEX <- cleanCategoricalFeature(data$PERP_SEX, c("(null)", "U"))
data$PERP_RACE <- cleanCategoricalFeature(data$PERP_RACE, c("(null)"))
data$VIC_AGE_GROUP <- cleanCategoricalFeature(data$VIC_AGE_GROUP, c("(null)", "1022"))
data$VIC_SEX <- cleanCategoricalFeature(data$VIC_SEX, c("U"))
```

```r
# print counts of categorical features, with missing values
printCountsCategoricalFeatures(data, "LOC_OF_OCCUR_DESC")
```

```
## [1] "LOC_OF_OCCUR_DESC"
##
##  INSIDE OUTSIDE UNKNOWN    <NA>
##     460    2506   25596       0
```

```r
printCountsCategoricalFeatures(data, "JURISDICTION_CODE")
```

```
## [1] "JURISDICTION_CODE"
##
##       0       1       2 UNKNOWN    <NA>
##   23923      81    4556       2       0
```

```r
printCountsCategoricalFeatures(data, "LOC_CLASSFCTN_DESC")
```

```
## [1] "LOC_CLASSFCTN_DESC"
##
##  COMMERCIAL    DWELLING     HOUSING       OTHER PARKING LOT  PLAYGROUND
##         208         243         460          59          15          41
##      STREET     TRANSIT     UNKNOWN     VEHICLE        <NA>
##        1886          23       25598          29           0
```

```r
printCountsCategoricalFeatures(data, "LOCATION_DESC")
```

```
## [1] "LOCATION_DESC"
##
##                        ATM                       BANK              BAR/NIGHT CLUB
##                          1                          3                         668
##          BEAUTY/NAIL SALON                CANDY STORE                 CHAIN STORE
##                        119                          7                           7
##                 CHECK CASH         CLOTHING BOUTIQUE             COMMERCIAL BLDG
##                          1                         14                         304
##                 DEPT STORE             DOCTOR/DENTIST                  DRUG STORE
##                          9                          1                          14
##       DRY CLEANER/LAUNDRY          FACTORY/WAREHOUSE                   FAST FOOD
##                         32                          8                         130
##                GAS STATION             GROCERY/BODEGA        GYM/FITNESS FACILITY
```

```
##                        74                        750                          4
##                  HOSPITAL                HOTEL/MOTEL              JEWELRY STORE
##                        77                         35                         14
##              LIQUOR STORE               LOAN COMPANY   MULTI DWELL - APT BUILD
##                        42                          1                       2964
## MULTI DWELL - PUBLIC HOUS                       NONE           PHOTO/COPY STORE
##                      5007                        175                          1
##                 PVT HOUSE          RESTAURANT/DINER                      SCHOOL
##                       983                        212                          1
##                SHOE STORE             SMALL MERCHANT SOCIAL CLUB/POLICY LOCATI
##                        10                         44                         73
##          STORAGE FACILITY          STORE UNCLASSIFIED               SUPERMARKET
##                         1                         37                         21
##            TELECOMM. STORE                    UNKNOWN              VARIETY STORE
##                        11                      16688                         11
##               VIDEO STORE                       <NA>
##                         8                          0
```

```r
printCountsCategoricalFeatures(data, "PERP_AGE_GROUP")
```

```
## [1] "PERP_AGE_GROUP"
##
##     <18   18-24   25-44   45-64     65+ UNKNOWN    <NA>
##    1682    6438    6041     699      65   13637       0
```

```r
printCountsCategoricalFeatures(data, "PERP_SEX")
```

```
## [1] "PERP_SEX"
##
##       F       M UNKNOWN    <NA>
##     444   16168   11950       0
```

```r
printCountsCategoricalFeatures(data, "PERP_RACE")
```

```
## [1] "PERP_RACE"
##
## AMERICAN INDIAN/ALASKAN NATIVE       ASIAN / PACIFIC ISLANDER
##                              2                            169
##                          BLACK                 BLACK HISPANIC
##                          11903                           1392
##                        UNKNOWN                          WHITE
##                          12288                            298
##                 WHITE HISPANIC                           <NA>
##                           2510                              0
```

```r
printCountsCategoricalFeatures(data, "BORO")
```

```
## [1] "BORO"
##
##         BRONX      BROOKLYN     MANHATTAN        QUEENS STATEN ISLAND
##          8376         11346          3762          4271           807
##          <NA>
##             0
```

```r
printCountsCategoricalFeatures(data, "PRECINCT")
```

```
## [1] "PRECINCT"
##
##    1    5    6    7    9   10   13   14   17   18   19   20   22   23   24   25
##   25   67   28  120  114   74   61   61   10   38   24   43    1  505  113  494
##   26   28   30   32   33   34   40   41   42   43   44   45   46   47   48   49
##  157  353  234  663  242  335  947  519  890  796 1076  195  972 1006  841  368
##   50   52   60   61   62   63   66   67   68   69   70   71   72   73   75   76
##  162  604  383  157   72  292   53 1259   36  484  479  595  117 1500 1628  179
##   77   78   79   81   83   84   88   90   94  100  101  102  103  104  105  106
##  821   65 1045  821  520  131  294  328   87  178  502  229  605  108  488  233
##  107  108  109  110  111  112  113  114  115  120  121  122  123 <NA>
##  105   75  123  174   12   23  834  397  185  597  114   63   33    0
```

```r
printCountsCategoricalFeatures(data, "STATISTICAL_MURDER_FLAG")
```

```
## [1] "STATISTICAL_MURDER_FLAG"
##
## FALSE  TRUE  <NA>
## 23036  5526     0
```

```r
printCountsCategoricalFeatures(data, "VIC_AGE_GROUP")
```

```
## [1] "VIC_AGE_GROUP"
##
##     <18   18-24   25-44   45-64     65+ UNKNOWN    <NA>
##    2954   10384   12973    1981     205      65       0
```

```r
printCountsCategoricalFeatures(data, "VIC_SEX")
```

```
## [1] "VIC_SEX"
##
##       F       M UNKNOWN    <NA>
##    2760   25790      12       0
```

```r
printCountsCategoricalFeatures(data, "VIC_RACE")
```

```
## [1] "VIC_RACE"
##
## AMERICAN INDIAN/ALASKAN NATIVE       ASIAN / PACIFIC ISLANDER
##                             11                            440
##                          BLACK                 BLACK HISPANIC
##                          20235                           2795
##                        UNKNOWN                          WHITE
##                             70                            728
##                 WHITE HISPANIC                           <NA>
##                           4283                              0
```

After printing out the category counts above we can see that there are no more NA data types and instead have been converted into "UNKNOWN" categories. Next we will drop the UNKNOWN cagtegory from the JURISDICTION_CODE feature as it only has a count of 2 which is not enough to train a model.

```
# drop records with low category counts
data <- dplyr::filter(data, JURISDICTION_CODE != "UNKNOWN")
printCountsCategoricalFeatures(data, "JURISDICTION_CODE")
```

```
## [1] "JURISDICTION_CODE"
##
##     0     1     2  <NA>
## 23923    81  4556     0
```

For this continuous feature, we will get the counts of all NA vs not NA values.

```
# print counts of NA vs not NA for continuous features
printCountsContinuousFeatures = function(data, feature) {
    na_count <- sum(is.na(data[[feature]]))
    non_na_count <- sum(!is.na(data[[feature]]))
    print(feature)
    cat("NA:", na_count, "    Not NA:", non_na_count, "\n\n")
}
```

```
# print counts of NA vs not NA for continuous features
printCountsContinuousFeatures(data, "LATITUDE")
```

```
## [1] "LATITUDE"
## NA: 59     Not NA: 28501
```

```
printCountsContinuousFeatures(data, "LONGITUDE")
```

```
## [1] "LONGITUDE"
## NA: 59     Not NA: 28501
```

```
printCountsContinuousFeatures(data, "LON_LAT")
```

```
## [1] "LON_LAT"
## NA: 59     Not NA: 28501
```

We can see that most of the records for this continuous feature have a not NA value.

Impute Continuous Features

For continuous features, we will replace the missing values with the mean of the feature. This is a common strategy for continuous features as it is a simple way to fill in the missing values. Also from the large counts above, we know this distribution will approach a normal distribution and the mean is a good estimate for this type of distribution as the distribution curve is naturally forming around the mean value.

```r
# replace missing values with mean for continuous features
lon_mean = mean(data$LONGITUDE, na.rm = TRUE)
lat_mean = mean(data$LATITUDE, na.rm = TRUE)
data$LONGITUDE <- ifelse(is.na(data$LONGITUDE), lon_mean, data$LONGITUDE)
data$LATITUDE <- ifelse(is.na(data$LATITUDE), lat_mean, data$LATITUDE)
data$LON_LAT <- ifelse(is.na(data$LON_LAT), paste("POINT (", lon_mean, lat_mean, ")"), data$LON_LAT)
```

```r
# print counts of NA vs not NA for continuous features
printCountsContinuousFeatures(data, "LATITUDE")
```

```
## [1] "LATITUDE"
## NA: 0     Not NA: 28560
```

```r
printCountsContinuousFeatures(data, "LONGITUDE")
```

```
## [1] "LONGITUDE"
## NA: 0     Not NA: 28560
```

```r
printCountsContinuousFeatures(data, "LON_LAT")
```

```
## [1] "LON_LAT"
## NA: 0     Not NA: 28560
```

Above we can see that there are no longer any missing values for the continuous features. Next we will clean the date and time features with some built in functions from the lubridate package.

Clean Date and Time

```r
# combine date and time into new date features
data$OCCUR_DATE <- mdy(data$OCCUR_DATE)
data$OCCUR_DATE_TIME <- ymd_hms(paste(data$OCCUR_DATE, data$OCCUR_TIME))
data$OCCUR_HOUR <- hour(data$OCCUR_TIME)
data$OCCUR_DAY_OF_WEEK <- wday(data$OCCUR_DATE, week_start = 1)
data$OCCUR_MONTH <- month(data$OCCUR_DATE)
head(data)
```

```
## # A tibble: 6 x 25
##   INCIDENT_KEY OCCUR_DATE OCCUR_TIME BORO      LOC_OF_OCCUR_DESC PRECINCT
##          <dbl> <date>     <time>     <chr>     <chr>                <dbl>
## 1    244608249 2022-05-05 00:10      MANHATTAN INSIDE                  14
## 2    247542571 2022-07-04 22:20      BRONX     OUTSIDE                 48
## 3     84967535 2012-05-27 19:35      QUEENS    UNKNOWN                103
## 4    202853370 2019-09-24 21:00      BRONX     UNKNOWN                 42
## 5     27078636 2007-02-25 21:00      BROOKLYN  UNKNOWN                 83
## 6    230311078 2021-07-01 23:07      MANHATTAN UNKNOWN                 23
## # i 19 more variables: JURISDICTION_CODE <chr>, LOC_CLASSFCTN_DESC <chr>,
## #   LOCATION_DESC <chr>, STATISTICAL_MURDER_FLAG <lgl>, PERP_AGE_GROUP <chr>,
## #   PERP_SEX <chr>, PERP_RACE <chr>, VIC_AGE_GROUP <chr>, VIC_SEX <chr>,
## #   VIC_RACE <chr>, X_COORD_CD <dbl>, Y_COORD_CD <dbl>, LATITUDE <dbl>,
## #   LONGITUDE <dbl>, LON_LAT <chr>, OCCUR_DATE_TIME <dttm>, OCCUR_HOUR <int>,
## #   OCCUR_DAY_OF_WEEK <dbl>, OCCUR_MONTH <dbl>
```

Above we can see that we created new features OCCUR_DATE_TIME, OCCUR_HOUR, OCCUR_DAY_OF_WEEK, and OCCUR_MONTH. Later we will see that confirm if these new features are useful for the model. The hypothesis is that the time of day, day of week, and month may have an impact on the number of incidents.

Here is a check for any duplicate records in the data set which could cause issues training the model.

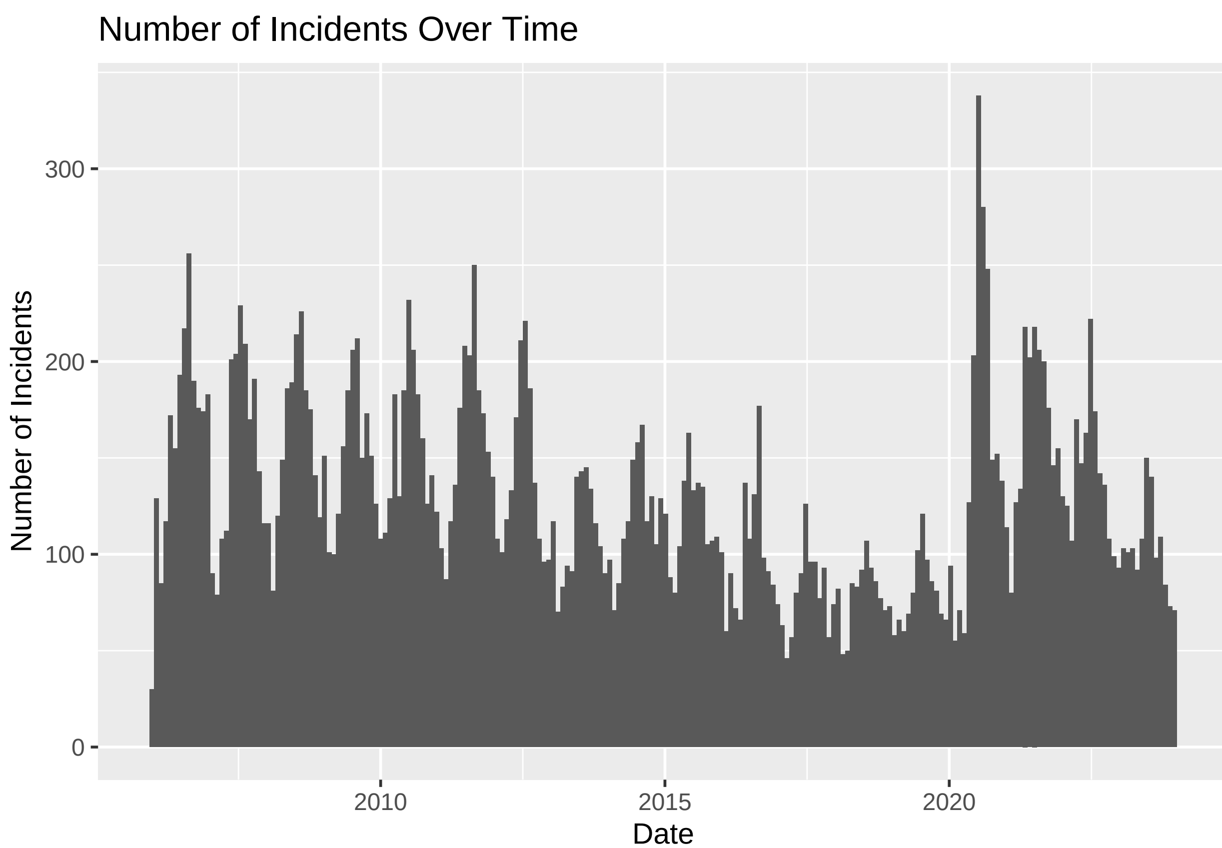
```
# check for duplicates
sum(duplicated(data))
```

```
## [1] 0
```

Visualize Data

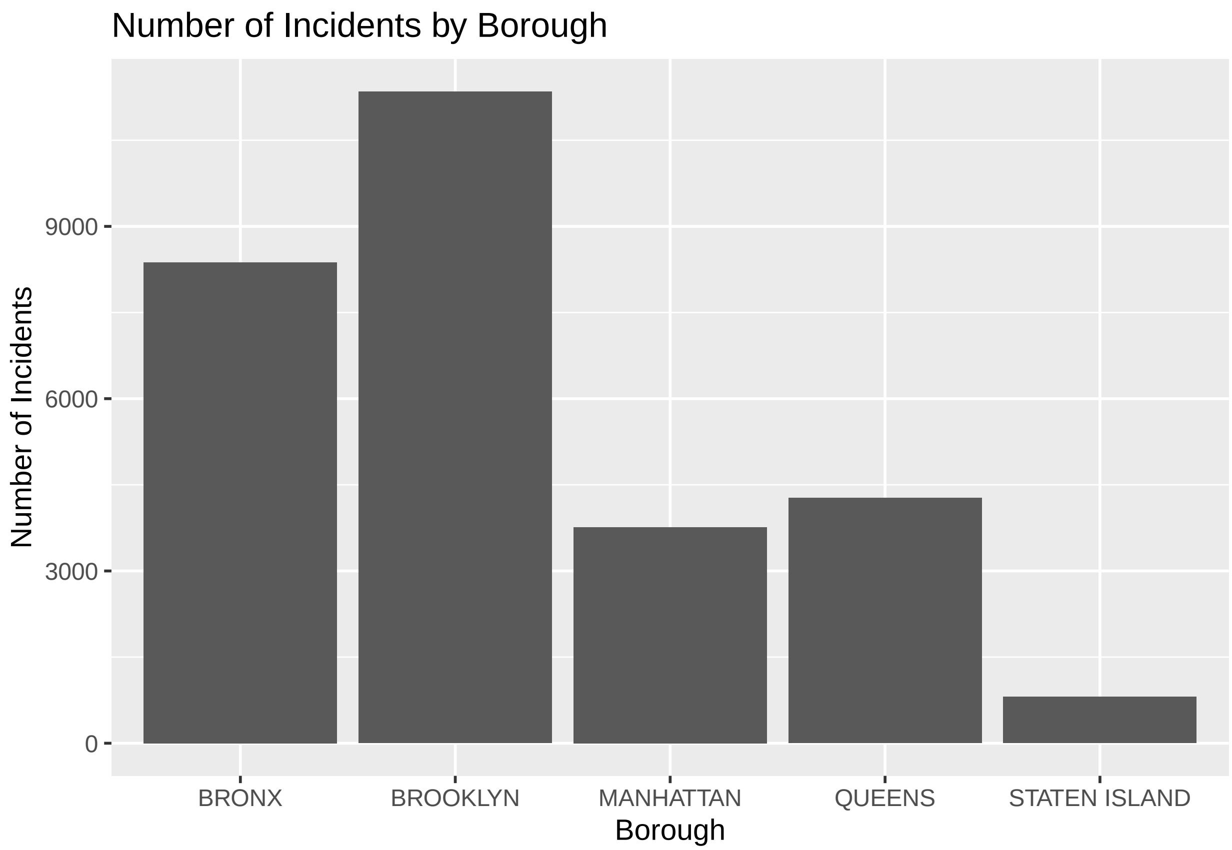
Next we start to visualize the data to understand the distribution of the data and relationships between features. First we look at the number of incidents per day over several years.

```
# Plot the number of incidents over time
ggplot(data, aes(x = OCCUR_DATE)) +
  geom_histogram(binwidth = 30) +
  labs(title = "Number of Incidents Over Time", x = "Date", y = "Number of Incidents")
```



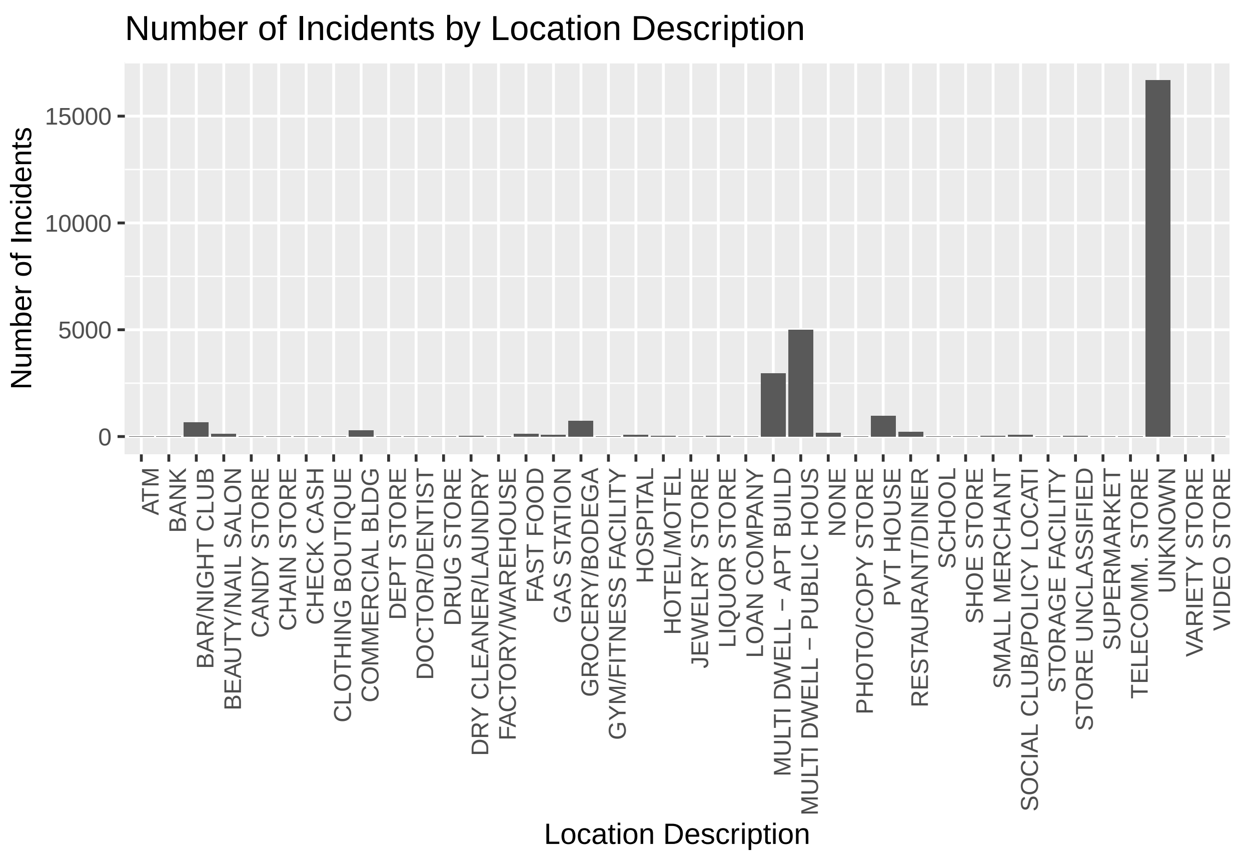
On first glance this data appears to be cyclic which is likely caused by the date, or maybe the month of the year. This could have many causes including weather, number of hours daylight, number of people outside, etc. Next we will look at number of incidents by the borough to see if any stand out compared to the others.

```r
# Plot incidents by borough
ggplot(data, aes(x = BORO)) +
  geom_bar() +
  labs(title = "Number of Incidents by Borough", x = "Borough", y = "Number of Incidents")
```



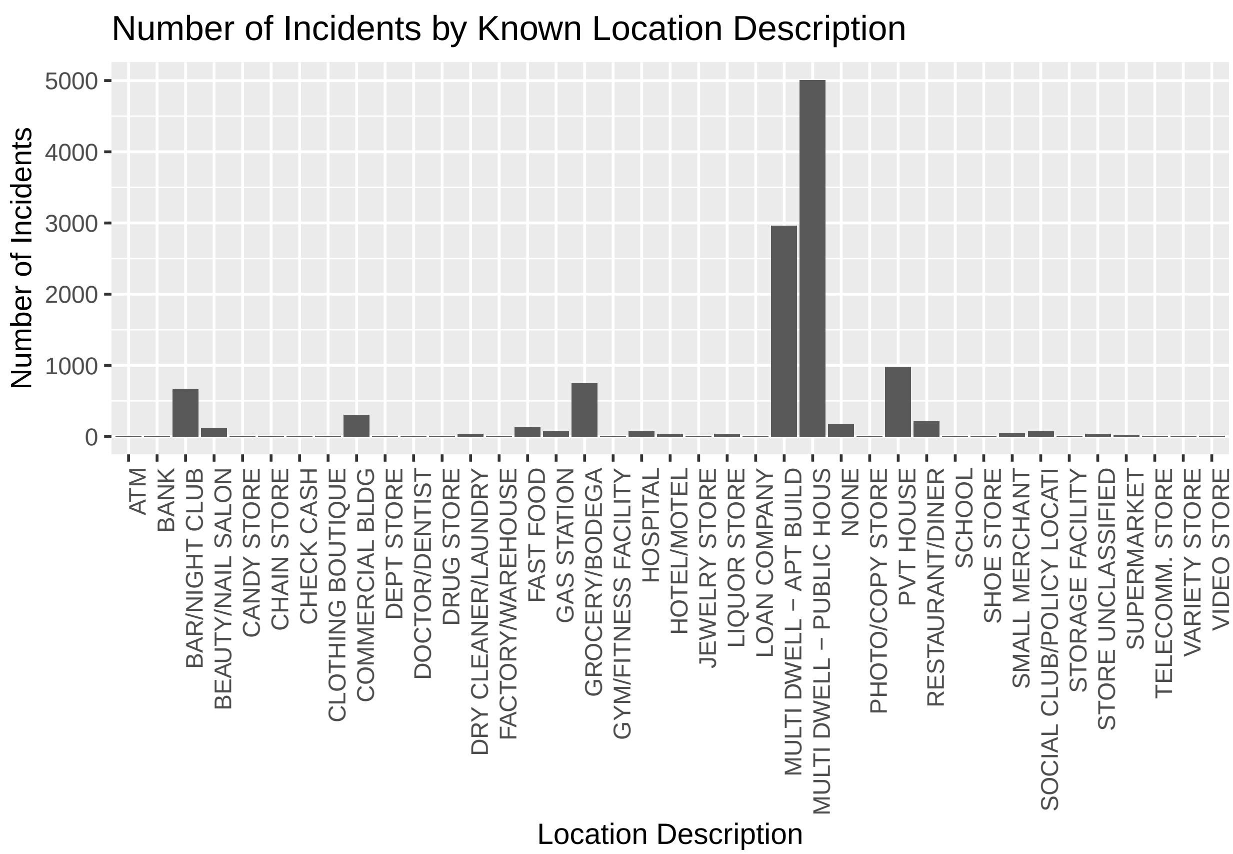
We can see that Brooklyn and Bronx have the highest number of incidents, and Staten Island with the fewest. Next we will get more specific in the location and look at the number of incidents by location description.

```r
# Plot incidents by location description
ggplot(data, aes(x = LOCATION_DESC)) +
  geom_bar() +
  labs(title = "Number of Incidents by Location Description", x = "Location Description", y = "Number of
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



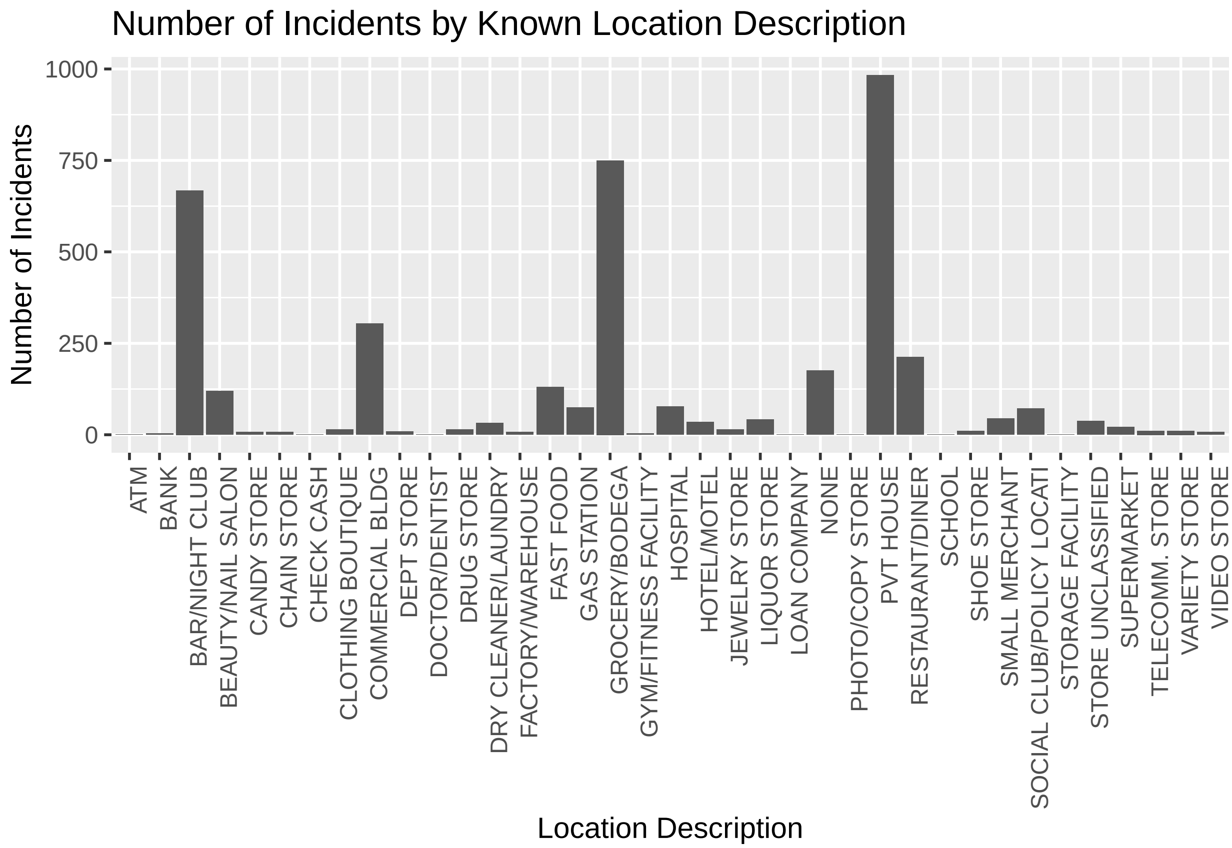
There is an obvious outlier in the data with the category "UNKNOWN" which is our placeholder for missing values. Next let's plot this again, ignoring the UNKNOWNS so we can see the range of the other values.

```
# Plot incidents by location description, ignoring UNKNOWN
ggplot(data = subset(data, LOCATION_DESC != "UNKNOWN"), aes(x = LOCATION_DESC)) +
  geom_bar() +
  labs(title = "Number of Incidents by Known Location Description", x = "Location Description", y = "Nu
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



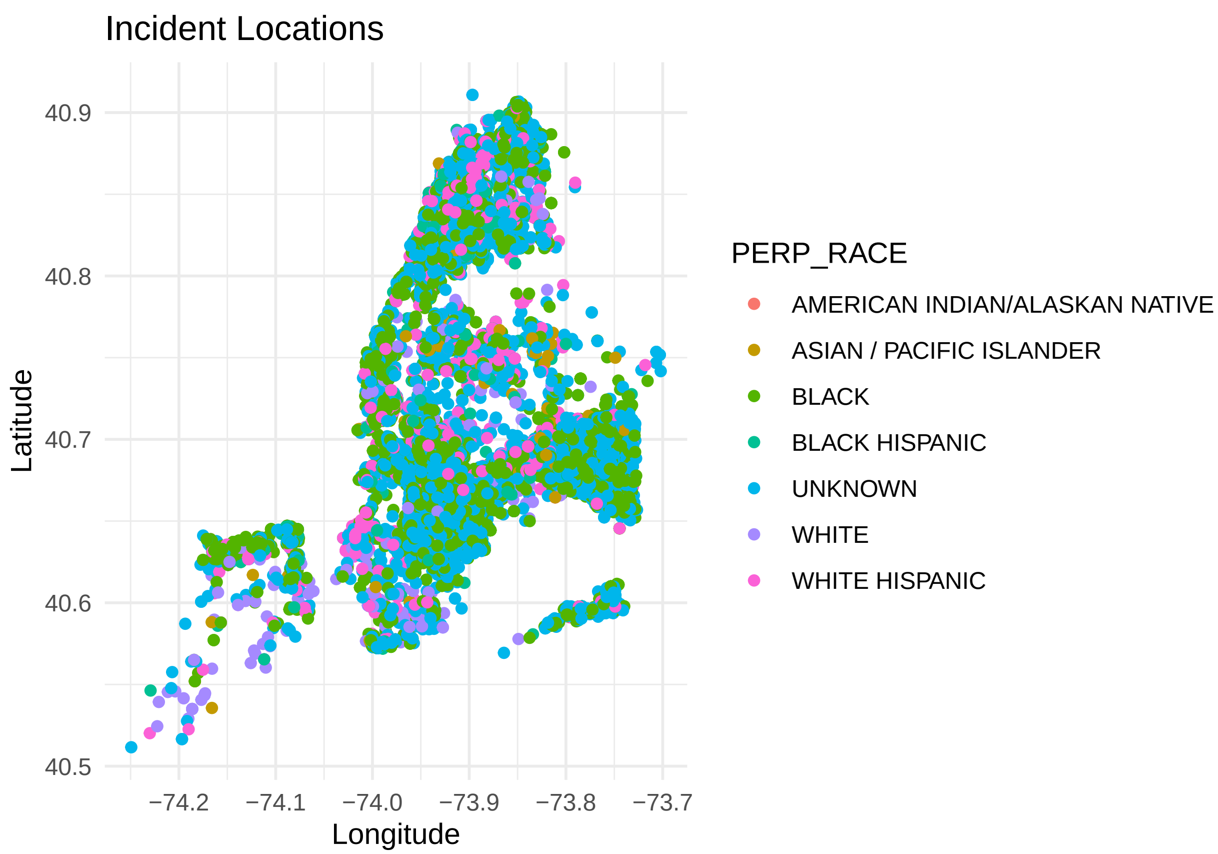
Even further, let's plot this one more time removing the two highest values that remain above: "MULTI DWELL - APT BUILD", "MULTI DWELL - PUBLIC HOUS."

```r
# Plot incidents by location description, ignoring UNKNOWN, MULTI DWELL - APT BUILD, and MULTI DWELL - 
ggplot(data = subset(data, LOCATION_DESC != "UNKNOWN" & LOCATION_DESC != "MULTI DWELL - APT BUILD" & LO
  geom_bar() +
  labs(title = "Number of Incidents by Known Location Description", x = "Location Description", y = "Nu
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



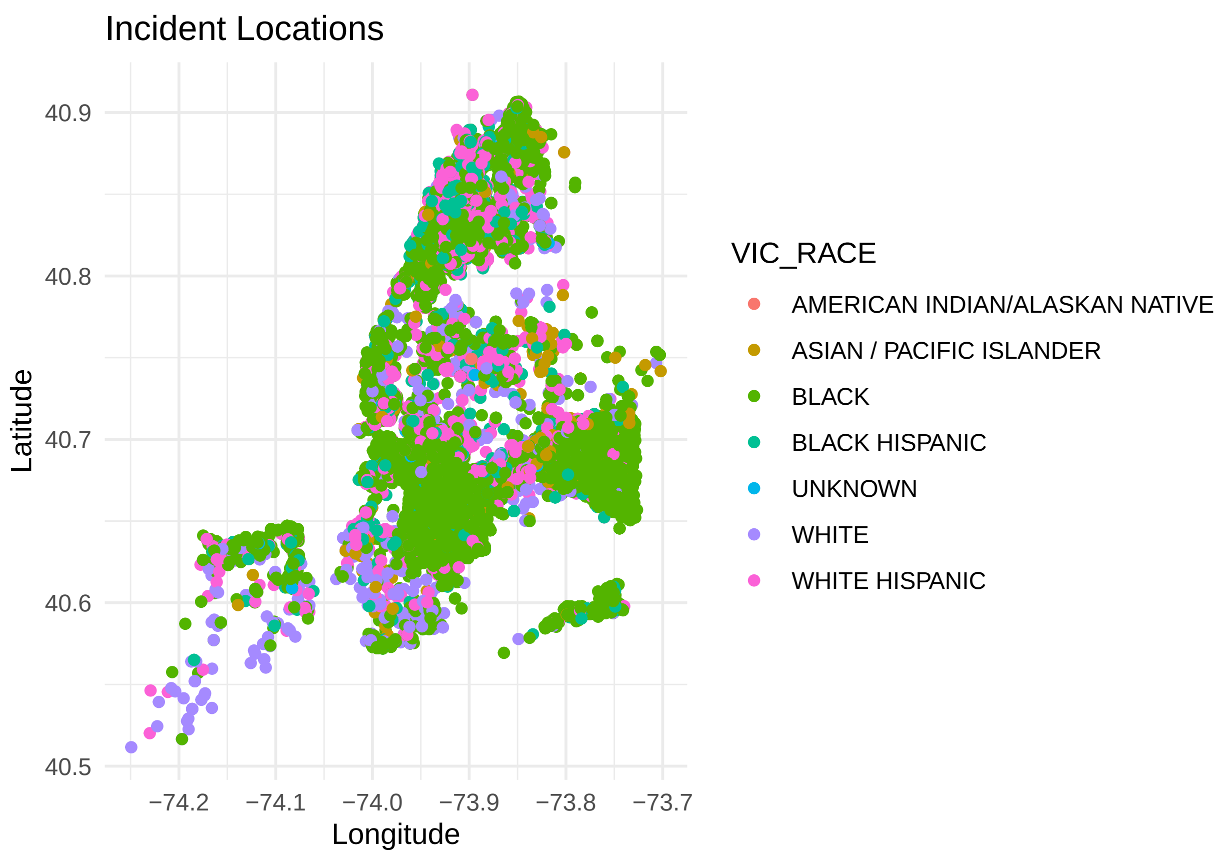
We could continue this further to determine the range of the remaining values but we need to keep in mind the number of incidents per category is starting to become negligible compared to the highest frequency values removed already. Next we will create a scatter plot of the latitude and longitude coordinates where each point is colored by the race of the perpetrator.

```
# create scatter plot for latitude and longitude coordinates of incidents
ggplot(data) +
  geom_point(mapping = aes(x = LONGITUDE, y = LATITUDE, color = PERP_RACE)) +
  labs(title = "Incident Locations", x = "Longitude", y = "Latitude") +
  theme_minimal()
```



Next we will create a similar scatter plot of the latitude and longitude coordinates, but this time where each point is colored by the race of the victim.

```
# create scatter plot for latitude and longitude coordinates of incidents
ggplot(data) +
  geom_point(mapping = aes(x = LONGITUDE, y = LATITUDE, color = VIC_RACE)) +
  labs(title = "Incident Locations", x = "Longitude", y = "Latitude") +
  theme_minimal()
```



Comparing the two scatter plots, the top graph of PERP_RACE has a significant amount of blue "UNKNOWN" points which line up pretty well with the green "BLACK" points in the bottom graph of VIC_RACE. The other colors also seem to line up fairly well matched, pink on top of pink, purple on top of purple, etc. This would indicate that most incidents involve both the perpetrator and victim being the race. If we assume this trend can be extrapolated, we can hypothesize the "UNKNOWN" perpetrators are likely "BLACK" like the victims. However, we must be careful not to make assumptions as there are many other factors that could be at play here and we need to caution over generalizing the data.

Analyze Data

Next we will start to analyze the relationships between the features in the data. First we will look at the relationship between the sex of the perpetrator and victim.

```
# create data frame to analyze relationship between sex
sex_df <- as.data.frame(table(data$PERP_SEX, data$VIC_SEX))
colnames(sex_df) <- c("PERP_SEX", "VIC_SEX", "n")
sex_df
```

```
##   PERP_SEX VIC_SEX     n
## 1        F       F    77
## 2        M       F  1755
## 3  UNKNOWN       F   928
## 4        F       M   366
## 5        M       M 14404
```

```
## 6  UNKNOWN       M 11018
## 7        F UNKNOWN     1
## 8        M UNKNOWN     7
## 9  UNKNOWN UNKNOWN     4
```

Here we will look at the relationship between the race of the perpetrator and victim.

```r
# create data frame to analyze relationship between race
race_df <- as.data.frame(table(data$PERP_RACE, data$VIC_RACE))
colnames(race_df) <- c("PERP_RACE", "VIC_RACE", "n")
race_df
```

```
##                         PERP_RACE                       VIC_RACE    n
## 1  AMERICAN INDIAN/ALASKAN NATIVE AMERICAN INDIAN/ALASKAN NATIVE    0
## 2        ASIAN / PACIFIC ISLANDER AMERICAN INDIAN/ALASKAN NATIVE    0
## 3                           BLACK AMERICAN INDIAN/ALASKAN NATIVE    4
## 4                  BLACK HISPANIC AMERICAN INDIAN/ALASKAN NATIVE    0
## 5                         UNKNOWN AMERICAN INDIAN/ALASKAN NATIVE    6
## 6                           WHITE AMERICAN INDIAN/ALASKAN NATIVE    0
## 7                  WHITE HISPANIC AMERICAN INDIAN/ALASKAN NATIVE    1
## 8  AMERICAN INDIAN/ALASKAN NATIVE       ASIAN / PACIFIC ISLANDER    0
## 9        ASIAN / PACIFIC ISLANDER       ASIAN / PACIFIC ISLANDER   61
## 10                          BLACK       ASIAN / PACIFIC ISLANDER  164
## 11                 BLACK HISPANIC       ASIAN / PACIFIC ISLANDER   20
## 12                        UNKNOWN       ASIAN / PACIFIC ISLANDER  140
## 13                          WHITE       ASIAN / PACIFIC ISLANDER   13
## 14                 WHITE HISPANIC       ASIAN / PACIFIC ISLANDER   42
## 15 AMERICAN INDIAN/ALASKAN NATIVE                          BLACK    2
## 16       ASIAN / PACIFIC ISLANDER                          BLACK   56
## 17                          BLACK                          BLACK 9410
## 18                 BLACK HISPANIC                          BLACK  561
## 19                        UNKNOWN                          BLACK 9318
## 20                          WHITE                          BLACK   42
## 21                 WHITE HISPANIC                          BLACK  845
## 22 AMERICAN INDIAN/ALASKAN NATIVE                 BLACK HISPANIC    0
## 23       ASIAN / PACIFIC ISLANDER                 BLACK HISPANIC   14
## 24                          BLACK                 BLACK HISPANIC  839
## 25                 BLACK HISPANIC                 BLACK HISPANIC  365
## 26                        UNKNOWN                 BLACK HISPANIC 1114
## 27                          WHITE                 BLACK HISPANIC   23
## 28                 WHITE HISPANIC                 BLACK HISPANIC  440
## 29 AMERICAN INDIAN/ALASKAN NATIVE                        UNKNOWN    0
## 30       ASIAN / PACIFIC ISLANDER                        UNKNOWN    0
## 31                          BLACK                        UNKNOWN   25
## 32                 BLACK HISPANIC                        UNKNOWN    6
## 33                        UNKNOWN                        UNKNOWN   26
## 34                          WHITE                        UNKNOWN    1
## 35                 WHITE HISPANIC                        UNKNOWN   12
## 36 AMERICAN INDIAN/ALASKAN NATIVE                          WHITE    0
## 37       ASIAN / PACIFIC ISLANDER                          WHITE   12
## 38                          BLACK                          WHITE  205
## 39                 BLACK HISPANIC                          WHITE   36
## 40                        UNKNOWN                          WHITE  207
```

```
## 41                              WHITE                              WHITE  165
## 42                     WHITE HISPANIC                              WHITE  103
## 43 AMERICAN INDIAN/ALASKAN NATIVE                         WHITE HISPANIC    0
## 44       ASIAN / PACIFIC ISLANDER                         WHITE HISPANIC   26
## 45                              BLACK                     WHITE HISPANIC 1255
## 46                     BLACK HISPANIC                     WHITE HISPANIC  404
## 47                            UNKNOWN                     WHITE HISPANIC 1477
## 48                              WHITE                     WHITE HISPANIC   54
## 49                     WHITE HISPANIC                     WHITE HISPANIC 1066
```

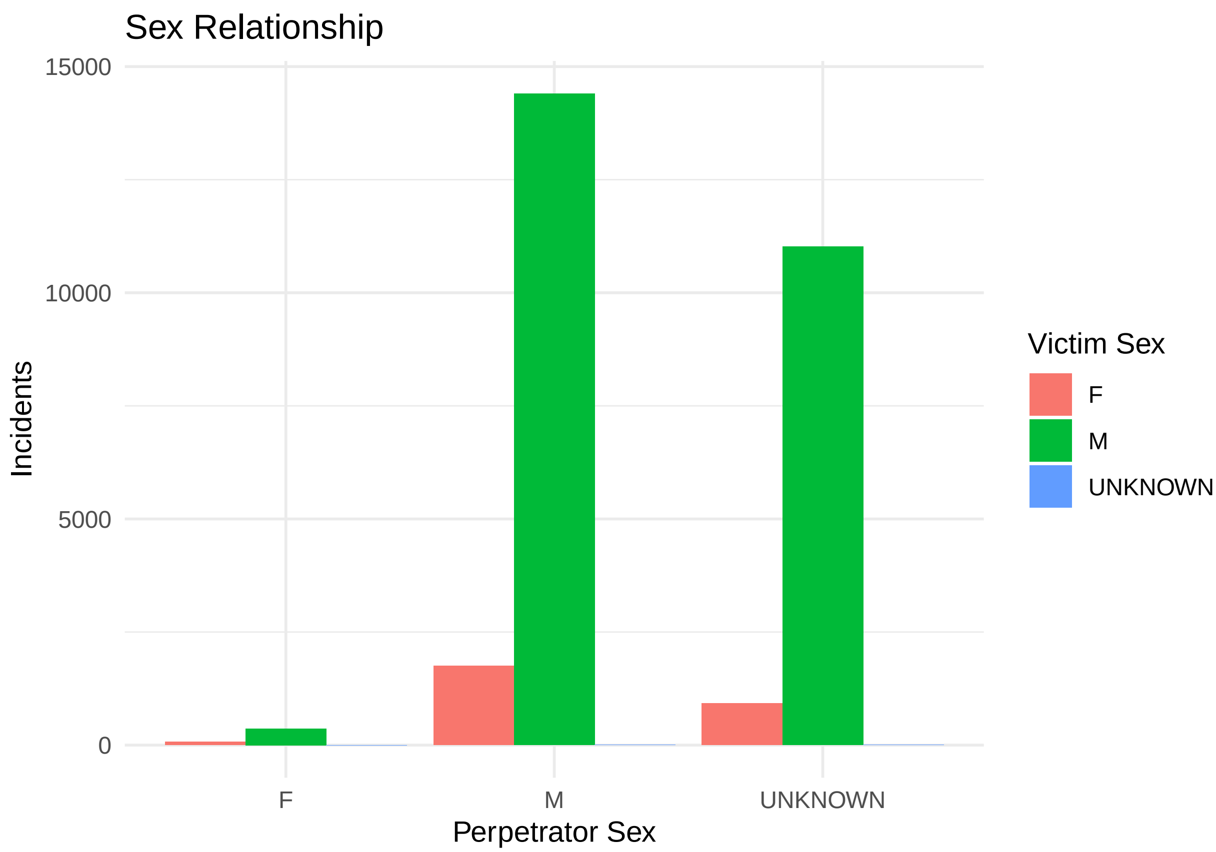
Lastly we will look at the relationship between the age of the perpetrator and victim.

```
# create data frame to analyze relationship between age
age_df <- as.data.frame(table(data$PERP_AGE_GROUP, data$VIC_AGE_GROUP))
colnames(age_df) <- c("PERP_AGE_GROUP", "VIC_AGE_GROUP", "n")
age_df
```

```
##    PERP_AGE_GROUP VIC_AGE_GROUP    n
## 1             <18           <18  521
## 2           18-24           <18  808
## 3           25-44           <18  270
## 4           45-64           <18   21
## 5             65+           <18    0
## 6         UNKNOWN           <18 1334
## 7             <18         18-24  651
## 8           18-24         18-24 2841
## 9           25-44         18-24 1560
## 10          45-64         18-24   85
## 11            65+         18-24    2
## 12        UNKNOWN         18-24 5244
## 13            <18         25-44  413
## 14          18-24         25-44 2394
## 15          25-44         25-44 3600
## 16          45-64         25-44  373
## 17            65+         25-44   27
## 18        UNKNOWN         25-44 6165
## 19            <18         45-64   79
## 20          18-24         45-64  335
## 21          25-44         45-64  524
## 22          45-64         45-64  202
## 23            65+         45-64   24
## 24        UNKNOWN         45-64  817
## 25            <18           65+   15
## 26          18-24           65+   47
## 27          25-44           65+   49
## 28          45-64           65+   13
## 29            65+           65+   12
## 30        UNKNOWN           65+   69
## 31            <18       UNKNOWN    2
## 32          18-24       UNKNOWN   13
## 33          25-44       UNKNOWN   38
## 34          45-64       UNKNOWN    5
## 35            65+       UNKNOWN    0
## 36        UNKNOWN       UNKNOWN    7
```

Next we will visualize these relationships with bar plots, starting with the sex relationship.

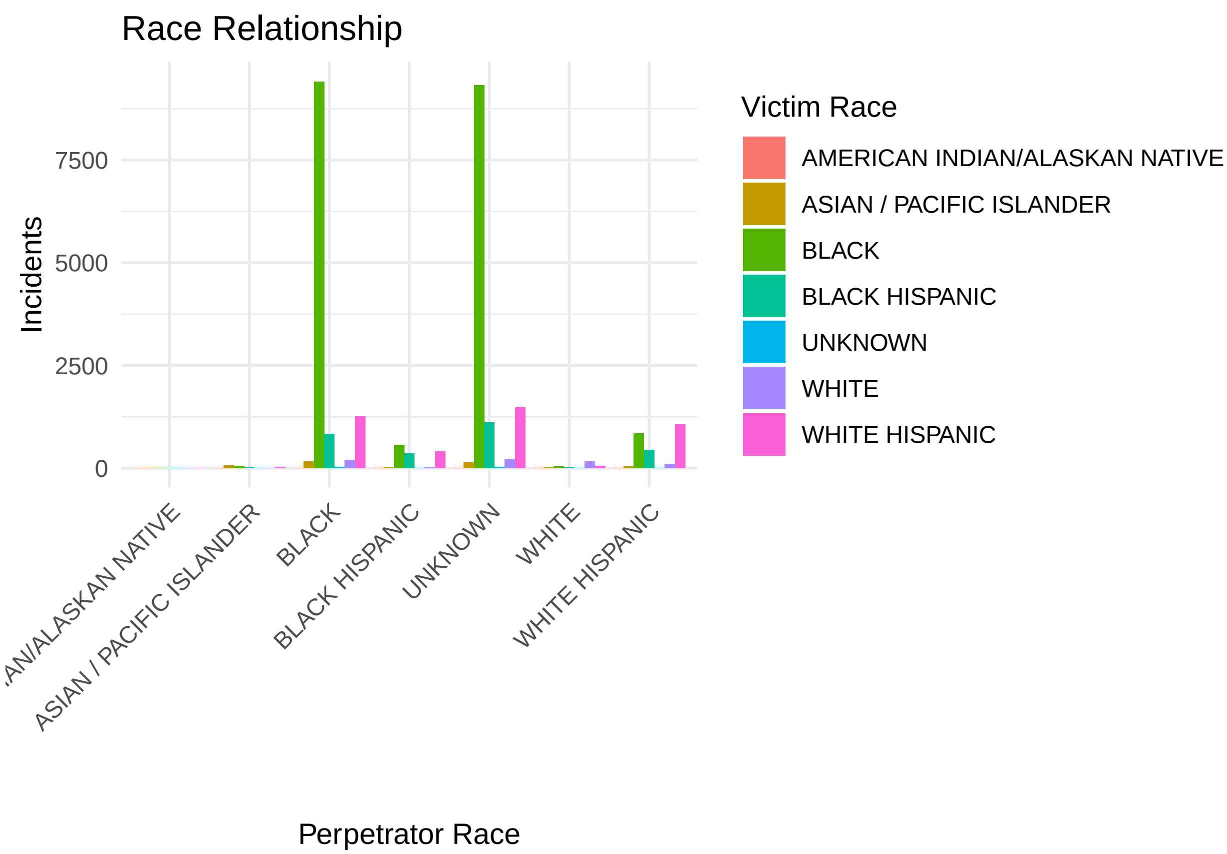
```r
# plot relationship between perpetrator and victim sex
ggplot(sex_df, aes(x = PERP_SEX, y = n, fill = VIC_SEX)) +
  geom_bar(stat = "identity", position="dodge") +
  labs(title = "Sex Relationship", x = "Perpetrator Sex", y = "Incidents", fill = "Victim Sex") +
  theme_minimal() +
  theme(legend.position = "right")
```



Above we can see there is a clear relationship between MALE perpetrators and victims. We can also see there are many incidents where the perpetrator is UNKNOWN and the victim is MALE. We could reach a similar conclusion as before that the UNKNOWN perpetrators are likely MALE. However, must be cautious again to not over generalize the data and make assumptions. We can also see there are very few incidents where the perpetrator and victim are both FEMALE, especially compared to the other categories.

Next we will plot the relationship between race.

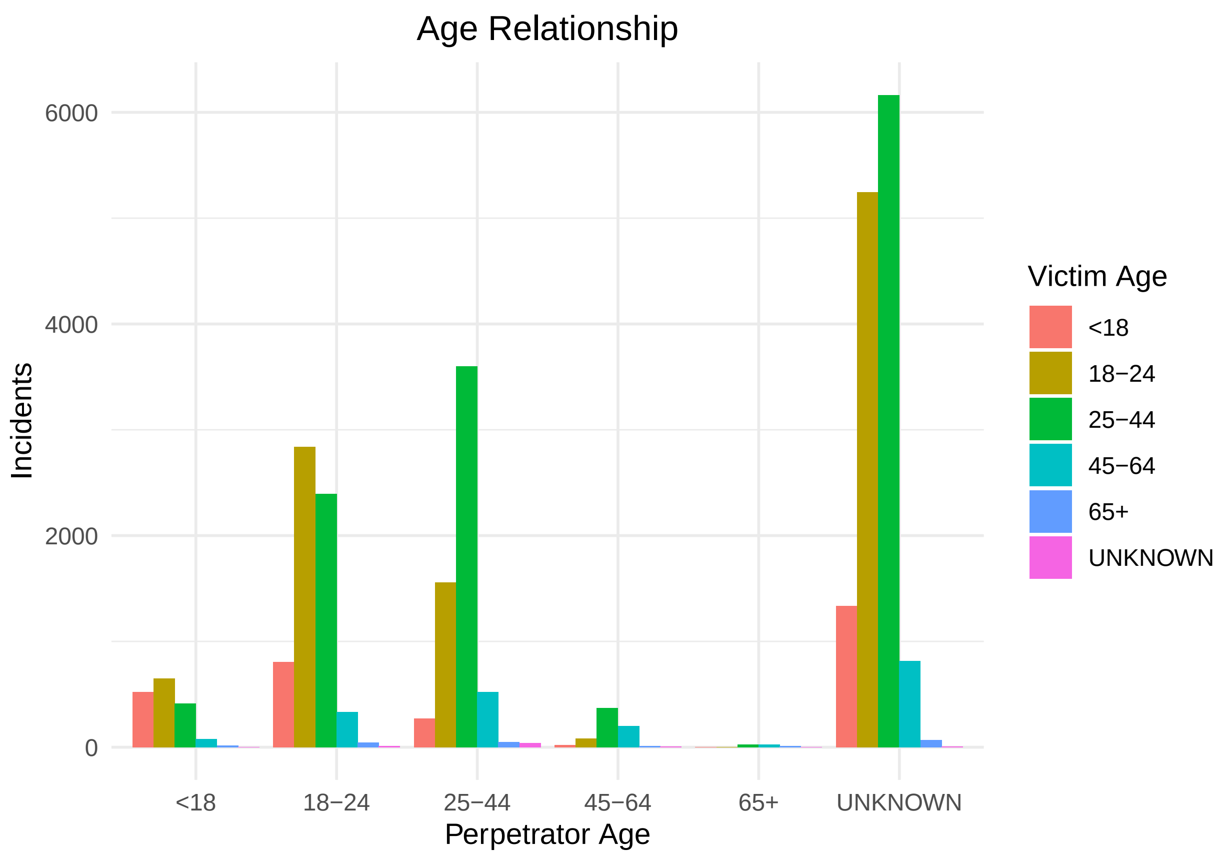
```r
# plot relationship between perpetrator and victim race
ggplot(race_df, aes(x = PERP_RACE, y = n, fill = VIC_RACE)) +
  geom_bar(stat = "identity", position="dodge") +
  labs(title = "Race Relationship", x = "Perpetrator Race", y = "Incidents", fill = "Victim Race") +
  theme_minimal() +
  theme(legend.position = "right", axis.text.x = element_text(angle = 45, hjust = 1))
```



Above we can see there is a clear relationship between BLACK perpetrators and BLACK victims. We can also see there is a significant number of incidents where the perpetrator is UNKNOWN and the victim is BLACK. Again, we could reach a similar conclusion as before that the UNKNOWN perpetrators are likely BLACK but we must be cautious to not over generalize the data and make assumptions. Besides those two relationships, we can see there are very few incidents among the other categories.

Next we will plot the relationship between age.

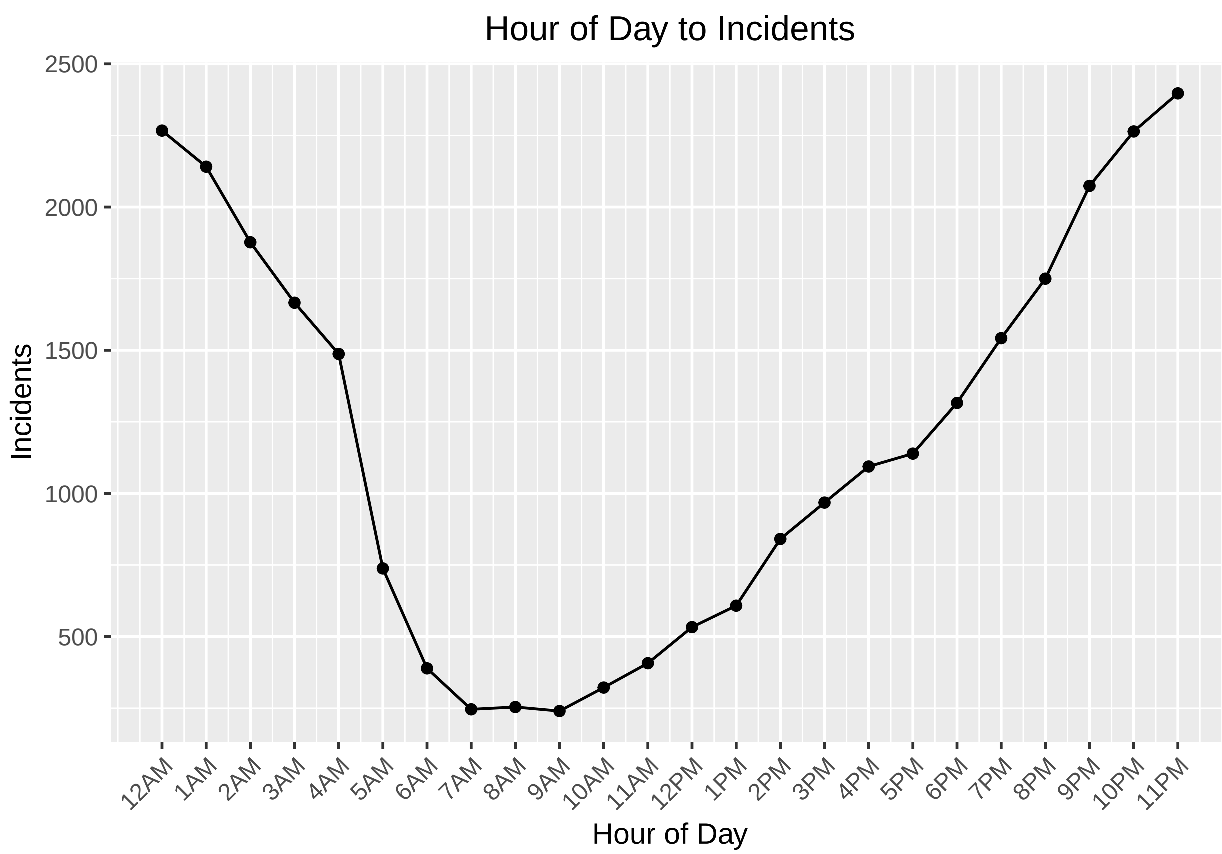
```r
# plot relationship between perpetrator and victim age
ggplot(age_df, aes(x = PERP_AGE_GROUP, y = n, fill = VIC_AGE_GROUP)) +
  geom_bar(stat = "identity", position="dodge") +
  labs(title = "Age Relationship", x = "Perpetrator Age", y = "Incidents", fill = "Victim Age") +
  theme_minimal() +
  theme(legend.position = "right", plot.title = element_text(hjust = 0.5))
```



Above we can see there might be a relationship between perpetrator and victim age but it is not as clear as the other relationships. Ignoring the unknown perpetrator relationships, the majority of the incidents are between the age groups 18-24 and 25-44. However, we can see that the unknown perpetrators also have the highest occurrence of incidents with the 18-24 and 25-44 age groups. Again we could reach a similar conclusion as before with the various examples but we must be cautious not to over generalize the data and make assumptions.

Next we will look at the relationship between the hour of the day and the number of incidents.

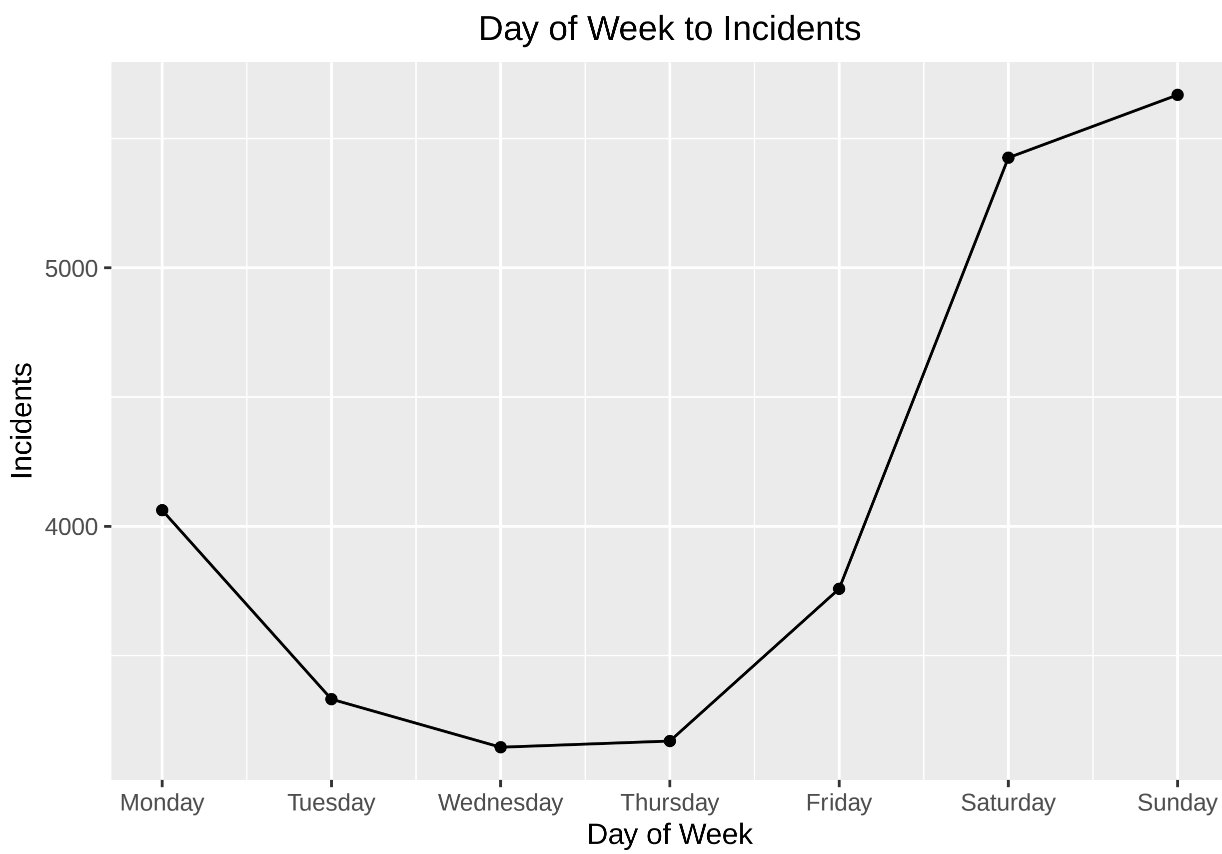
```
# plot relationship between hour of day and incidents
hour_data <- data %>%
  count(OCCUR_HOUR)
ggplot(hour_data, aes(x = OCCUR_HOUR, y = n)) +
  geom_line() +
  geom_point() +
  labs(title = "Hour of Day to Incidents", x = "Hour of Day", y = "Incidents") +
  theme(plot.title = element_text(hjust = 0.5), axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_x_continuous(breaks = 0:23, labels = c("12AM", "1AM", "2AM", "3AM", "4AM", "5AM", "6AM", "7AM",
```



Above we can see there is a clear relationship between the hour of the day and the number of incidents. The greatest number of incidents occur over night and drop off during the day. This is a potentially good feature to use in the model.

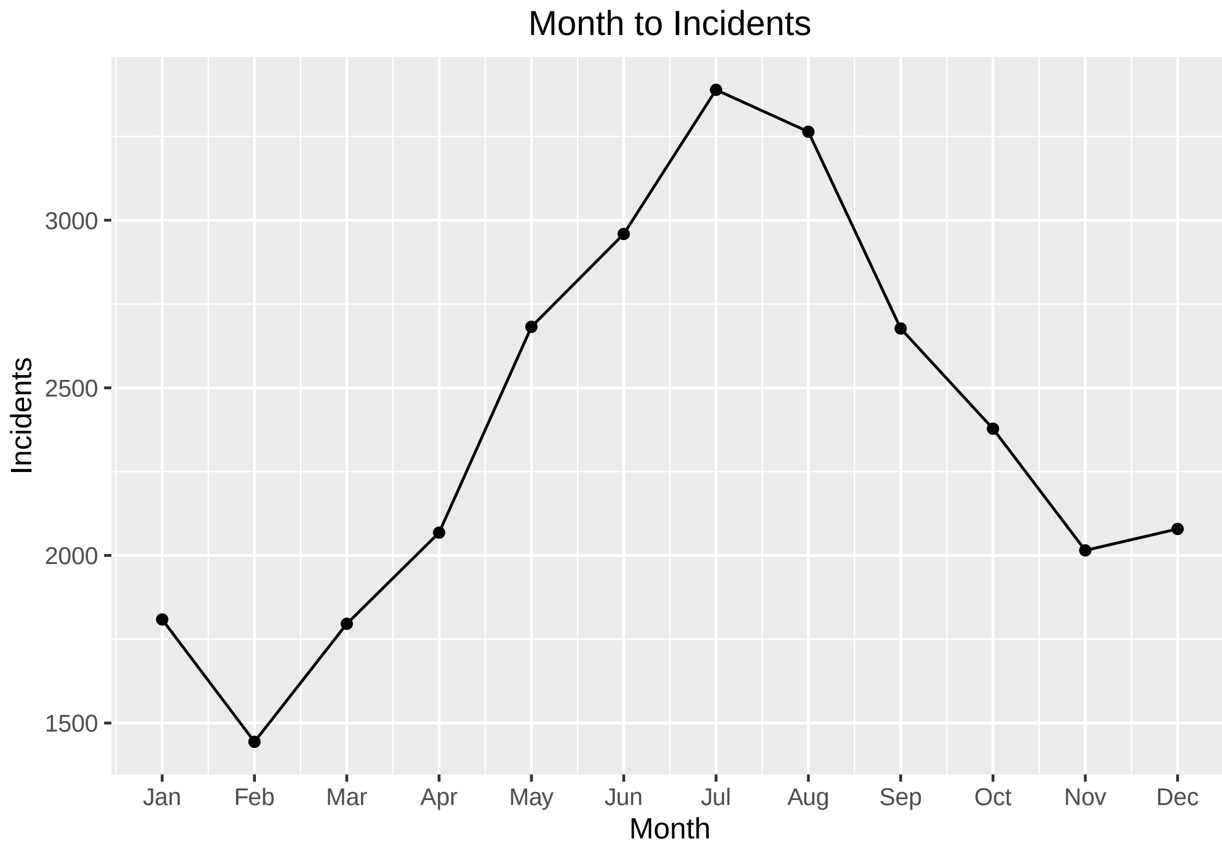
Next we will look at the relationship between the day of the week and the number of incidents.

```
# plot relationship between day of week to incidents
day_of_week_data <- data %>%
  count(OCCUR_DAY_OF_WEEK)
ggplot(day_of_week_data, aes(x = OCCUR_DAY_OF_WEEK, y = n)) +
  geom_line() +
  geom_point() +
  labs(title = "Day of Week to Incidents", x = "Day of Week", y = "Incidents") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_x_continuous(breaks = 1:7, labels = c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "S
```



Above we can see there is a clear relationship between the day of the week and the number of incidents, with the greatest number of incidents occurring on the weekends and dropping off during the week. Next we will look at the relationship between the month of the year and the number of incidents.

```
# plot relationship between month to incidents
month_data <- data %>%
  count(OCCUR_MONTH)
ggplot(month_data, aes(x = OCCUR_MONTH, y = n)) +
  geom_line() +
  geom_point() +
  labs(title = "Month to Incidents", x = "Month", y = "Incidents") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_x_continuous(breaks = 1:12, labels = c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug",
```



Above we can see a relationship between the month of the year and the number of incidents, where the greatest number of incidents occur in the summer months and drop off during the winter months.

Features for Model

Now we will start to prepare the model by selecting the features we want to use. These vectors of column names will come in handy later when we preprocess the data for the model. We break these up into different types of features: nominal, ordinal, numeric, numeric factor, logical, and date features. This is done to help us understand the data better and to help us decide how to preprocess the data for the model.

```
# define features
nominal_features <- c(
   "JURISDICTION_CODE" # nominal
  ,"VIC_AGE_GROUP"     # nominal
)

# ordinal features
ordinal_features <- c()

# numeric features
numeric_features <- c(

)

# numeric factor features
```

```
numeric_factor_features <- c(
   "OCCUR_HOUR"        # numeric
  ,"OCCUR_DAY_OF_WEEK" # numeric
  ,"OCCUR_MONTH"       # numeric
)

# logical features
logical_features <- c()

# date features
date_features <- c()

# target variable
target <- "STATISTICAL_MURDER_FLAG"
```

Preprocess Data for Model

Next we begin to preprocess the data for the model. We start by filtering down the data frame with only the features of interest and the target variable. Then we need to convert some data types to something that can be used by the model. The nominal features need to be turned into factors, and the ordinal features need to be turned into ordered factors. The numeric features get turned into factors, and so does the target variable which contains TRUE/FALSE values. We then split the data set into training and test sets, where 80% of the data is used for training the model and 20% is used for testing the model. We need to be careful to align the factors between the training and test set as the randomness of the split can produce categorical values in the test set that do not exist in the training set which will cause an error. Lastly we print the train and test sets for inspection and validation before modeling.

```
# select the relevant columns
model_data <- data %>%
  select(all_of(c(nominal_features, ordinal_features, numeric_features, numeric_factor_features, logica

# convert nominal features to factors
model_data <- model_data %>%
  mutate(across(all_of(nominal_features), as.factor))

# convert ordinal features to ordered factors
if (!is.null(ordinal_features)) {
  model_data <- model_data %>%
    mutate(across(all_of(ordinal_features), ~factor(.x, ordered = TRUE)))
}

# convert numeric features to factors
if (!is.null(numeric_factor_features)) {
  model_data <- model_data %>%
    mutate(across(all_of(numeric_factor_features), as.factor))
}

# convert target variable to factor
model_data[[target]] <- factor(model_data[[target]], levels = c(FALSE, TRUE), labels = c("No", "Yes"))

# split data into training and test sets
set.seed(123)
```

```r
splitIndex <- createDataPartition(model_data[[target]], p = 0.8, list = FALSE)
train_data <- model_data[splitIndex, ]
test_data <- model_data[-splitIndex, ]

# remove near zero variance predictors
nzv <- nearZeroVar(train_data, saveMetrics = TRUE)
train_data <- train_data[, !nzv$nzv]
test_data <- test_data[, colnames(test_data) %in% colnames(train_data)]

# align factors in test set with training set
for (col in names(test_data)) {
  if (is.factor(train_data[[col]])) {
    test_data[[col]] <- factor(test_data[[col]], levels = levels(train_data[[col]]))
  }
}

# print train and test sets before modeling
sapply(train_data, class)
```

```
##       JURISDICTION_CODE           VIC_AGE_GROUP              OCCUR_HOUR
##                "factor"                "factor"                "factor"
##       OCCUR_DAY_OF_WEEK             OCCUR_MONTH STATISTICAL_MURDER_FLAG
##                "factor"                "factor"                "factor"
```

```r
summary(train_data)
```

```
##  JURISDICTION_CODE VIC_AGE_GROUP     OCCUR_HOUR    OCCUR_DAY_OF_WEEK
##  0:19156           <18     : 2352   23      : 1927   1:3228
##  1:   69           18-24   : 8343   0       : 1820   2:2662
##  2: 3624           25-44   :10366   22      : 1808   3:2468
##                    45-64   : 1567   1       : 1728   4:2547
##                    65+     :  165   21      : 1681   5:3041
##                    UNKNOWN:   56    2       : 1504   6:4320
##                                     (Other):12381    7:4583
##   OCCUR_MONTH   STATISTICAL_MURDER_FLAG
##  7      :2742   No :18428
##  8      :2629   Yes: 4421
##  6      :2341
##  9      :2170
##  5      :2100
##  10     :1894
##  (Other):8973
```

Train Logistic Regression Model

Next we train a simple logistic regression model which is a classification model that produces a binary output. This model is a good starting point for classification problems and is easy to interpret. We will train the model on the training data and then evaluate the model on the test data.

```r
# train model
formula <- as.formula(paste(target, "~ ."))
train_data <- ovun.sample(formula, data = train_data, method = "under")$data
model <- glm(formula, data = train_data, family = "binomial")
```

Evaluate Model

Next we evaluate the model on the test data to see how well it performs on unseen data. First we get the probabilities of each prediction and then evaluate the classification based on a threshold of 0.5. We then evaluate the model using a variety of metrics including a confusion matrix, accuracy, precision, recall, F1 score, ROC AUC, and PR AUC.

```r
# get probabilities on the test set
probabilities <- predict(model, newdata = test_data, type = "response")

# make predictions on the test set
threshold <- 0.5
predictions <- ifelse(probabilities > threshold, "Yes", "No")

# Ensure predictions and target are factors with the same levels
predictions <- factor(predictions, levels = c("No", "Yes"))
actuals <- factor(test_data[[target]], levels = c("No", "Yes"))

# Print the unique predictions and their counts
cat("Unique predictions: \n", summary(predictions), "\n")
```

```
## Unique predictions:
##  2791 2920
```

```r
cat("Actuals summary: \n", summary(actuals), "\n")
```

```
## Actuals summary:
##  4606 1105
```

```r
# confusion matrix with positive class as "Yes"
conf_matrix <- confusionMatrix(predictions, actuals, positive = "Yes")

# evaluation metrics
accuracy <- conf_matrix$overall["Accuracy"]
precision <- posPredValue(predictions, actuals, positive = "Yes")
recall <- sensitivity(predictions, actuals, positive = "Yes")
f1_score <- F1_Score(predictions, actuals, positive = "Yes")
avg_precision <- PRAUC(probabilities, actuals)

# roc auc
roc_curve <- roc(actuals, probabilities)
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```r
roc_auc <- roc_curve$auc

# convert the target factor to numeric (0 and 1) for precision-recall calculation
numeric_flag <- as.numeric(actuals) - 1
pr_curve <- pr.curve(scores.class0 = probabilities, weights.class0 = numeric_flag, curve = TRUE)
pr_auc <- pr_curve$auc.integral
```

Now lets review the model summary and evaluation metrics for the logistic regression model.

```r
# print summary of model
summary(model)
```

```
##
## Call:
## glm(formula = formula, family = "binomial", data = train_data)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -0.446415   0.144631  -3.087  0.00202 **
## JURISDICTION_CODE1    0.078615   0.418593   0.188  0.85103
## JURISDICTION_CODE2   -0.193493   0.061348  -3.154  0.00161 **
## VIC_AGE_GROUP18-24    0.265334   0.081677   3.249  0.00116 **
## VIC_AGE_GROUP25-44    0.576290   0.079646   7.236 4.63e-13 ***
## VIC_AGE_GROUP45-64    0.635399   0.106613   5.960 2.52e-09 ***
## VIC_AGE_GROUP65+      0.515162   0.229412   2.246  0.02473 *
## VIC_AGE_GROUPUNKNOWN  0.692923   0.444133   1.560  0.11872
## OCCUR_HOUR1          -0.008414   0.111275  -0.076  0.93973
## OCCUR_HOUR2          -0.009818   0.114383  -0.086  0.93159
## OCCUR_HOUR3          -0.026756   0.121074  -0.221  0.82510
## OCCUR_HOUR4           0.063661   0.120744   0.527  0.59803
## OCCUR_HOUR5           0.421443   0.155609   2.708  0.00676 **
## OCCUR_HOUR6           0.322964   0.196743   1.642  0.10068
## OCCUR_HOUR7           0.159482   0.207992   0.767  0.44322
## OCCUR_HOUR8           0.381187   0.228074   1.671  0.09466 .
## OCCUR_HOUR9           0.298544   0.249335   1.197  0.23117
## OCCUR_HOUR10          0.489584   0.229830   2.130  0.03315 *
## OCCUR_HOUR11          0.198782   0.180045   1.104  0.26956
## OCCUR_HOUR12          0.335620   0.176299   1.904  0.05695 .
## OCCUR_HOUR13          0.329098   0.167113   1.969  0.04892 *
## OCCUR_HOUR14          0.043245   0.143911   0.300  0.76380
## OCCUR_HOUR15          0.106058   0.142830   0.743  0.45775
## OCCUR_HOUR16         -0.119993   0.142437  -0.842  0.39955
## OCCUR_HOUR17          0.250934   0.134452   1.866  0.06199 .
## OCCUR_HOUR18          0.311350   0.128059   2.431  0.01504 *
## OCCUR_HOUR19         -0.008681   0.121033  -0.072  0.94282
## OCCUR_HOUR20         -0.037653   0.118950  -0.317  0.75159
## OCCUR_HOUR21          0.205503   0.111214   1.848  0.06463 .
## OCCUR_HOUR22          0.108014   0.107733   1.003  0.31605
## OCCUR_HOUR23         -0.155433   0.108157  -1.437  0.15069
## OCCUR_DAY_OF_WEEK2   -0.081640   0.084635  -0.965  0.33473
## OCCUR_DAY_OF_WEEK3    0.065001   0.085637   0.759  0.44783
## OCCUR_DAY_OF_WEEK4   -0.020085   0.085626  -0.235  0.81455
## OCCUR_DAY_OF_WEEK5    0.087569   0.081771   1.071  0.28421
```

```
## OCCUR_DAY_OF_WEEK6   -0.030604   0.076409  -0.401  0.68876
## OCCUR_DAY_OF_WEEK7    0.021199   0.075862   0.279  0.77990
## OCCUR_MONTH2          0.136395   0.127199   1.072  0.28359
## OCCUR_MONTH3         -0.122912   0.118153  -1.040  0.29821
## OCCUR_MONTH4          0.110121   0.116930   0.942  0.34631
## OCCUR_MONTH5          0.087563   0.111055   0.788  0.43042
## OCCUR_MONTH6         -0.079106   0.109279  -0.724  0.46913
## OCCUR_MONTH7         -0.098837   0.105753  -0.935  0.34999
## OCCUR_MONTH8         -0.123667   0.107337  -1.152  0.24927
## OCCUR_MONTH9         -0.059392   0.108171  -0.549  0.58296
## OCCUR_MONTH10        -0.124646   0.112110  -1.112  0.26622
## OCCUR_MONTH11        -0.034619   0.118597  -0.292  0.77036
## OCCUR_MONTH12         0.163754   0.115497   1.418  0.15624
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 12205  on 8803  degrees of freedom
## Residual deviance: 12016  on 8756  degrees of freedom
## AIC: 12112
##
## Number of Fisher Scoring iterations: 4
```

First we can see the coefficients of the model which are the weights of the features. The coefficients are the log odds of the target variable being TRUE given the feature. The coefficients can be interpreted as the log odds of the target variable being TRUE when the feature is 1 compared to when the feature is 0. We can see the coefficients are all negative which means the log odds of the target variable being TRUE decreases as the feature increases. Next we can see the p-values of the coefficients which tell us if the feature is statistically significant in predicting the target variable. These statistically significant features are: JURISDICTION_CODE, OCCUR_HOUR, OCCUR_DAY_OF_WEEK, and OCCUR_MONTH.

Display Evaluation Metrics

Next we print the confusion matrix which is a 2x2 grid of the true positives, false positives, true negatives, and false negatives. This allows us to see where the model is correctly predicting the target variable and where it is not. We can also breakdown the prediction failures into type I errors (false positives) and type II errors (false negatives). We could also adjust the threshold of the model to reduce the number of false positives or false negatives depending on the use case and where we want the model to perform well and where we are fine with errors.

```r
# print confusion matrix
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  2358  433
##        Yes 2248  672
##
##                Accuracy : 0.5306
```

```
##                  95% CI : (0.5175, 0.5436)
##     No Information Rate : 0.8065
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0739
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6081
##             Specificity : 0.5119
##          Pos Pred Value : 0.2301
##          Neg Pred Value : 0.8449
##              Prevalence : 0.1935
##          Detection Rate : 0.1177
##    Detection Prevalence : 0.5113
##       Balanced Accuracy : 0.5600
##
##        'Positive' Class : Yes
##
```

Next we print the evaluation metrics which include accuracy, precision, recall, F1 score, ROC AUC, and PR AUC.

```r
# print evaluation metrics
cat("Accuracy: ", accuracy, "\n")
```

```
## Accuracy:  0.5305551
```

```r
cat("Precision: ", precision, "\n")
```

```
## Precision:  0.230137
```

```r
cat("Recall: ", recall, "\n")
```

```
## Recall:  0.6081448
```

```r
cat("F1 Score: ", f1_score, "\n")
```

```
## F1 Score:  0.333913
```

```r
cat("ROC AUC: ", roc_auc, "\n")
```

```
## ROC AUC:  0.577624
```

```r
cat("PR AUC: ", pr_auc, "\n")
```

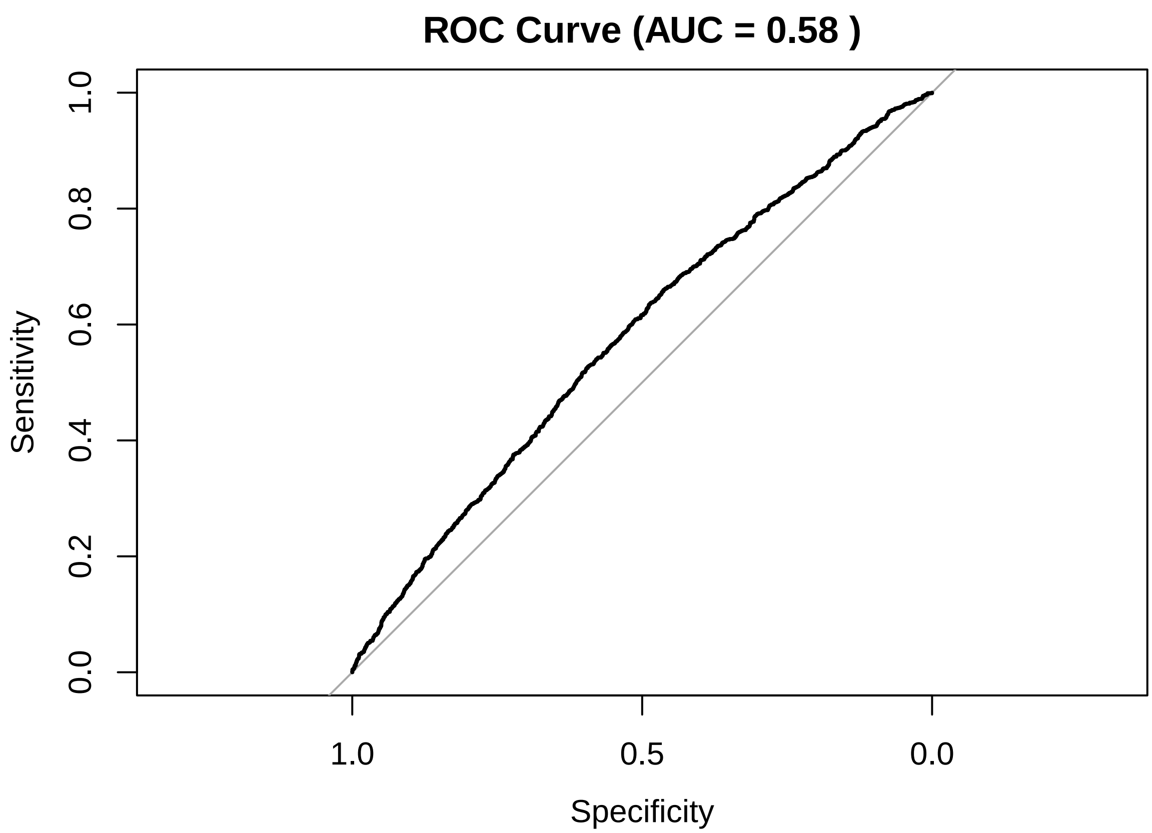
```
## PR AUC:  0.2420361
```

We can see that the model is not performing great yet with an accuracy of 0.53 which is not far from a random guess. We can also see that the recall is much better than the precision. This means the model is better at finding the positive class but it comes at a cost of over predicting the positive class and producing many false positives which results in a low precision score.

Plot Curves

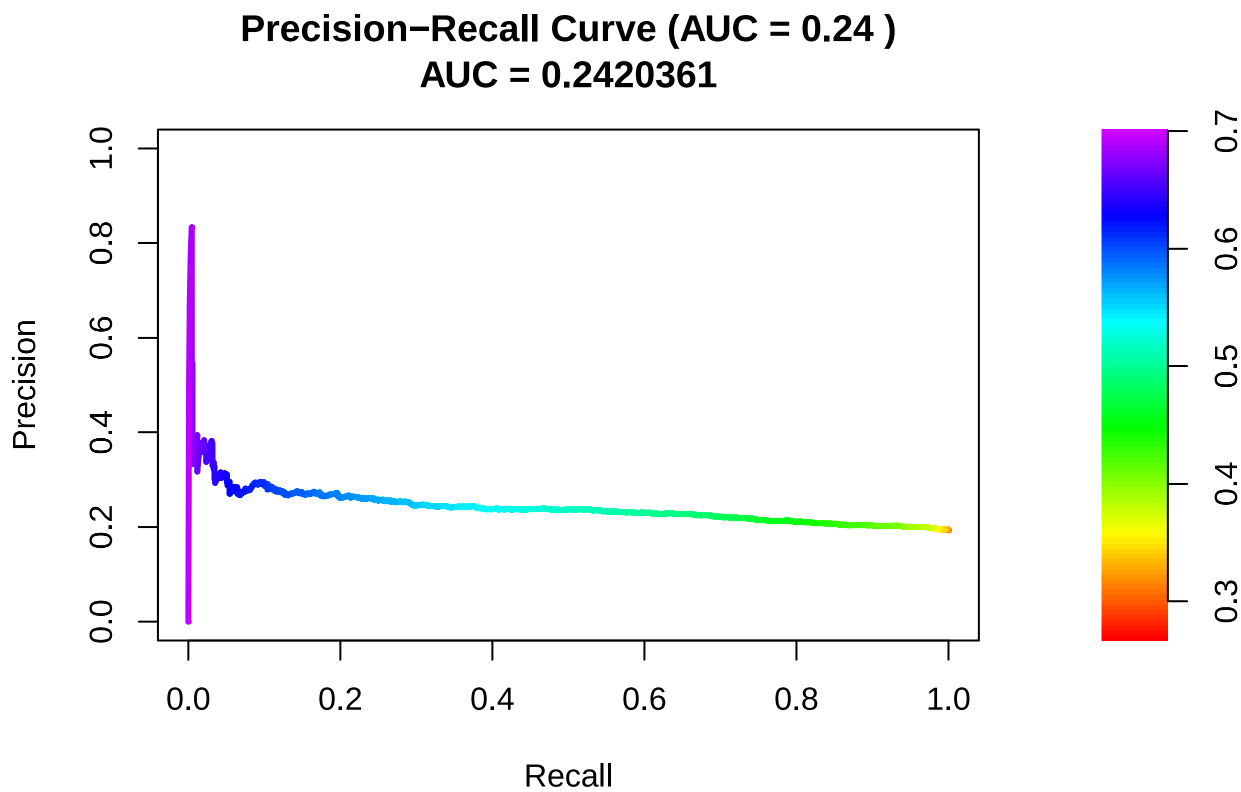
Next we will plot the ROC curve to visualize the trade off between the true positive rate and false positive rate. The ideal ROC curve hugs the top left corner of the plot which would indicate a perfect model. The diagonal line represents a random guess model. Our model lies somewhere in between those which is not great but not terrible either.

```
# plot roc curve
plot(roc_curve, main = paste("ROC Curve (AUC =", round(roc_auc, 2), ")"))
```



Next we will plot the precision recall curve to visualize the trade off between precision and recall. The ideal precision recall curve hugs the top right corner of the plot which would indicate a perfect model. We can see that the precision recall curve is much better than the ROC curve which is common for imbalanced data sets.

```
# plot precision recall curve
plot(pr_curve, main = paste("Precision-Recall Curve (AUC =", round(pr_auc, 2), ")"))
```



Conclusion

In conclusion, we have successfully cleaned the data, visualized the data, analyzed the data, and trained a logistic regression model. We have also evaluated the model and displayed the evaluation metrics to understand how well the model is performing. We found several features that appeared to show a relationship with the target variable and used those features to train the model. Those features included the hour of the day, day of the week, and month of the year, the race, sex, and possibly age of the perpetrator and victim, and the location of the incident. However, we were careful not to over generalize the data and make assumptions which might have led to bias in the model's performance. Overall the model is not performing great yet with an accuracy of 0.53. We also found that the model is better at finding the positive class as indicated by the recall score. However, it comes at a cost of over predicting the positive class and producing many false positives which results in a low precision score. Some future work could include trying different models, tuning hyperparameters, and engineering new features to improve the model's performance. We could also test the assumptions that we were hesitant to make and see if they improve the model's performance.