NYPD

2025-03-03

```r
# Libaries
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.2

## -- 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   3.5.1     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
```

```r
library(lubridate)
library(dplyr)
library(ggplot2)
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 4.4.3

##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.3

## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 4.4.3
```

```r
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

Introduction

The data contains NYPD shooting historical data

The task is to evaluate the data to see if we can identify any associations between fatal shootings and time of day, date, borough, or sex

Import the Data

```r
# Import data
data = read.csv("https://data.cityofnewyork.us/api/views/833y-fsy8/rows.csv?accessType=DOWNLOAD")
```

```r
# Create a quick summary of the data
summary(data)
```

```
##   INCIDENT_KEY        OCCUR_DATE         OCCUR_TIME             BORO
##  Min.   :  9953245   Length:28562       Length:28562       Length:28562
##  1st Qu.: 65439914   Class :character   Class :character   Class :character
##  Median : 92711254   Mode  :character   Mode  :character   Mode  :character
##  Mean   :127405824
##  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       Length:28562            Length:28562
##  Class :character   Class :character        Class :character
##  Mode  :character   Mode  :character        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
```

Data Preparation and Transformation

There are data and time columns that are character values. This needs to be string values.

```r
# Change date and time from chr to a string and creates a new combined column
data$OCCUR_DATE = as.Date(data$OCCUR_DATE, format = "%m/%d/%Y")
data$DATETIME = as.POSIXct(paste(data$OCCUR_DATE, data$OCCUR_TIME),
                           format = "%Y-%m-%d %H:%M:%S")
# Move DATETIME to the front by reordering all columns
data = data[, c("DATETIME", setdiff(names(data), "DATETIME"))]
```

```r
# Remove the original data and time columns
data$OCCUR_DATE = NULL
data$OCCUR_TIME = NULL
```

Preparation: Evaluate the missing data

```r
# Total missing
sum(is.na(data))
```

```
## [1] 120
```

```r
# Missing by column
colSums(is.na(data))
```

```
##                DATETIME            INCIDENT_KEY                    BORO
##                       0                       0                       0
##        LOC_OF_OCCUR_DESC                PRECINCT       JURISDICTION_CODE
##                       0                       0                       2
##       LOC_CLASSFCTN_DESC           LOCATION_DESC STATISTICAL_MURDER_FLAG
##                       0                       0                       0
##           PERP_AGE_GROUP                PERP_SEX               PERP_RACE
##                       0                       0                       0
##            VIC_AGE_GROUP                 VIC_SEX                VIC_RACE
##                       0                       0                       0
##               X_COORD_CD              Y_COORD_CD                Latitude
##                       0                       0                      59
##                Longitude                 Lon_Lat
##                      59                       0
```

```r
# Percent missing
missing_percent = colMeans(is.na(data)) * 100
print(missing_percent)
```

```
##                DATETIME            INCIDENT_KEY                    BORO
##             0.000000000             0.000000000             0.000000000
##        LOC_OF_OCCUR_DESC                PRECINCT       JURISDICTION_CODE
##             0.000000000             0.000000000             0.007002311
##       LOC_CLASSFCTN_DESC           LOCATION_DESC STATISTICAL_MURDER_FLAG
##             0.000000000             0.000000000             0.000000000
##           PERP_AGE_GROUP                PERP_SEX               PERP_RACE
##             0.000000000             0.000000000             0.000000000
##            VIC_AGE_GROUP                 VIC_SEX                VIC_RACE
##             0.000000000             0.000000000             0.000000000
##               X_COORD_CD              Y_COORD_CD                Latitude
##             0.000000000             0.000000000             0.206568167
##                Longitude                 Lon_Lat
##             0.206568167             0.000000000
```

Some of the columns have about 21% of missing data. The ways to handle this would be to impute the data. With this much missing data, this could introduce bias. Imputation would work better with continuous variables.The best way to handle this data would be to remove the missing data.If we remove the rows with the missing data, we will remove 61 rows, 0.21% of the data, preserving 99.79% of the data.

Preparation: Remove Missing Data

```
# remove the rows with na
data_clean = na.omit(data)
```

```
summary(data_clean)
```

```
##     DATETIME                       INCIDENT_KEY          BORO          
##  Min.   :2006-01-01 02:00:00.00   Min.   :  9953245   Length:28501      
##  1st Qu.:2009-08-31 00:50:00.00   1st Qu.: 65276038   Class :character  
##  Median :2013-09-09 05:26:00.00   Median : 92550741   Mode  :character  
##  Mean   :2014-06-01 03:12:59.67   Mean   :127118170                     
##  3rd Qu.:2019-09-16 00:20:00.00   3rd Qu.:202504685                     
##  Max.   :2023-12-29 21:22:00.00   Max.   :279758069                     
##  LOC_OF_OCCUR_DESC     PRECINCT     JURISDICTION_CODE LOC_CLASSFCTN_DESC
##  Length:28501       Min.   :  1.0   Min.   :0.0000    Length:28501      
##  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   :0.3225                      
##                     3rd Qu.: 81.0   3rd Qu.:0.0000                      
##                     Max.   :123.0   Max.   :2.0000                      
##  LOCATION_DESC      STATISTICAL_MURDER_FLAG PERP_AGE_GROUP    
##  Length:28501       Length:28501            Length:28501      
##  Class :character   Class :character        Class :character  
##  Mode  :character   Mode  :character        Mode  :character  
##                                                               
##                                                               
##                                                               
##    PERP_SEX          PERP_RACE         VIC_AGE_GROUP        VIC_SEX         
##  Length:28501       Length:28501       Length:28501       Length:28501      
##  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:28501       Min.   : 914928   Min.   :125757   Min.   :40.51  
##  Class :character   1st Qu.:1000068   1st Qu.:182905   1st Qu.:40.67  
##  Mode  :character   Median :1007776   Median :194872   Median :40.70  
##                     Mean   :1009438   Mean   :208375   Mean   :40.74  
##                     3rd Qu.:1016807   3rd Qu.:239814   3rd Qu.:40.82  
##                     Max.   :1066815   Max.   :271128   Max.   :40.91  
##    Longitude        Lon_Lat         
##  Min.   :-74.25   Length:28501      
##  1st Qu.:-73.94   Class :character  
##  Median :-73.92   Mode  :character  
```

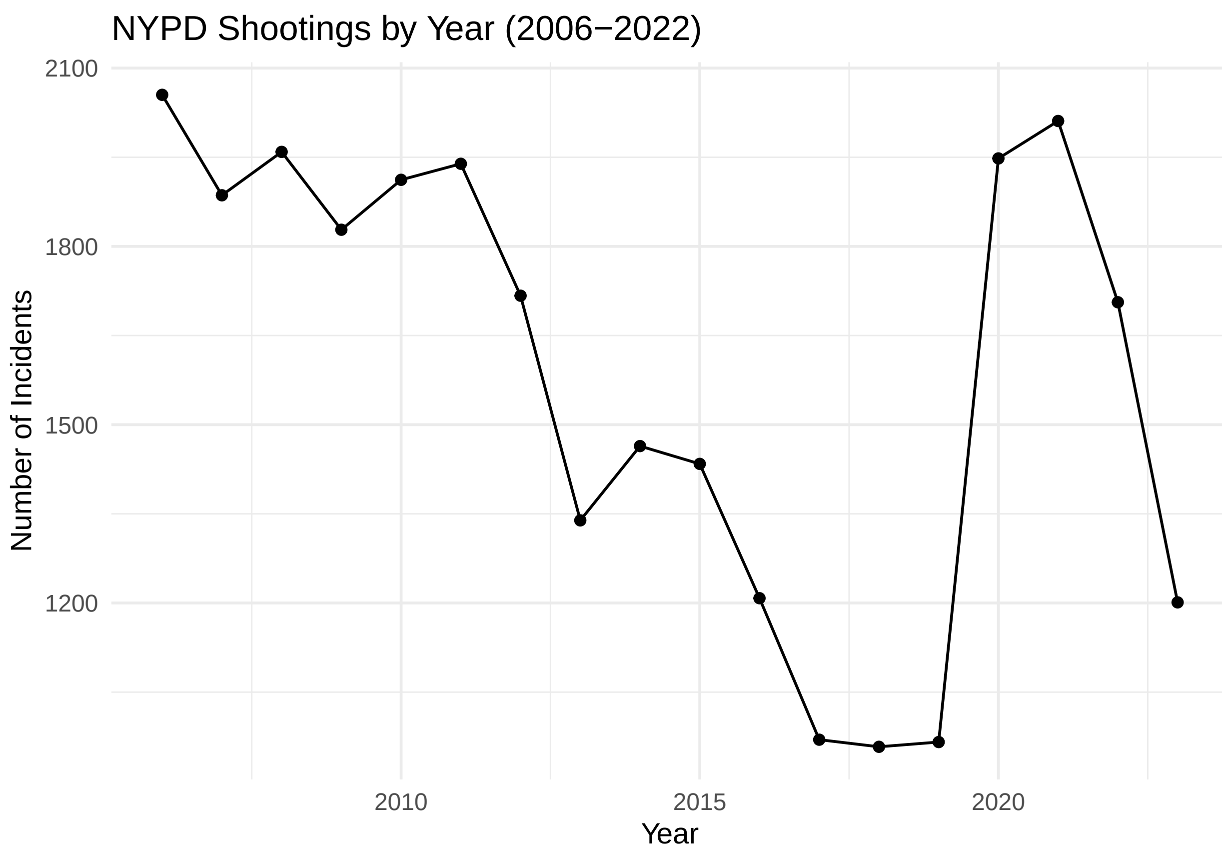
```
##  Mean   :-73.91
##  3rd Qu.:-73.88
##  Max.   :-73.70
```

```r
# Add year column
data_clean$year = year(data_clean$DATETIME)

# quick check to ensure it worked
head(data_clean[, c("DATETIME", "year")])
```

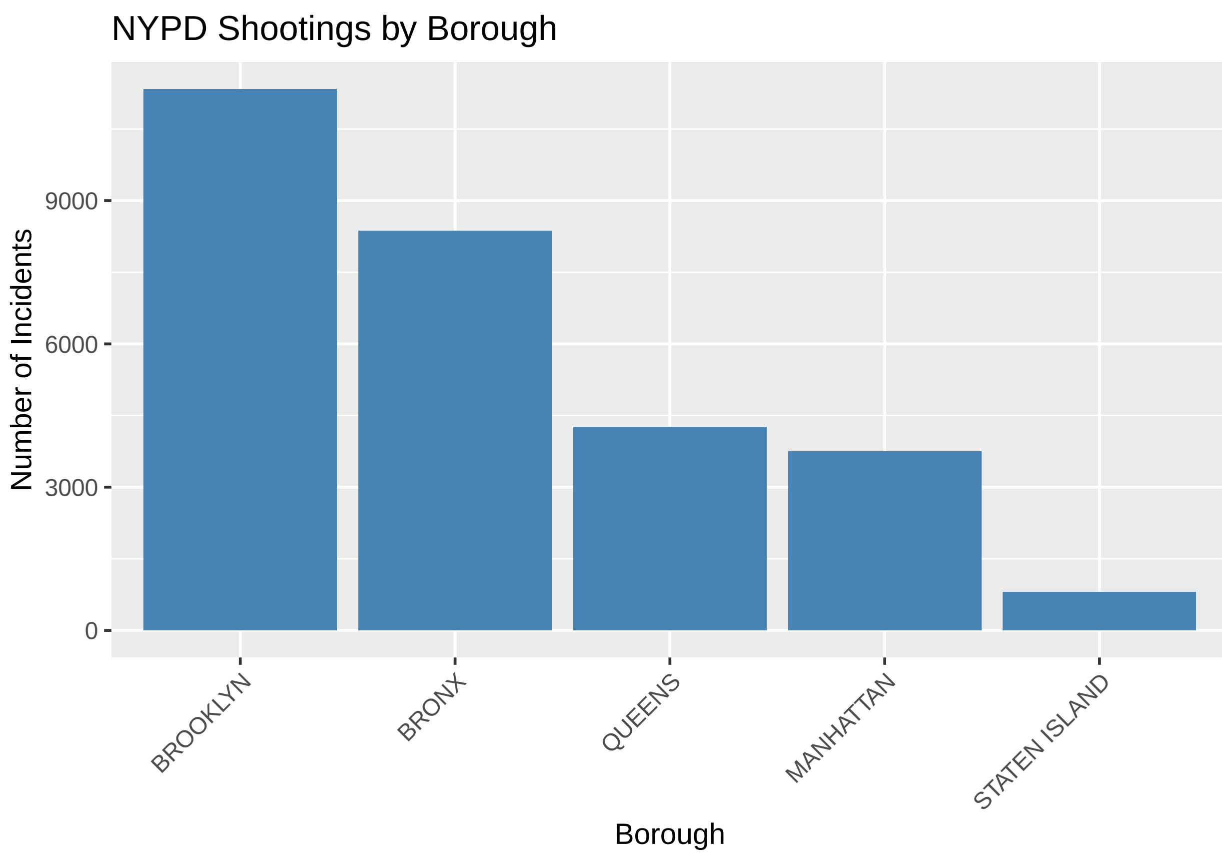
```
##              DATETIME year
## 1 2021-08-09 01:06:00 2021
## 2 2018-04-07 19:48:00 2018
## 3 2022-12-02 22:57:00 2022
## 4 2006-11-19 01:50:00 2006
## 5 2010-05-09 01:58:00 2010
## 6 2012-07-22 21:35:00 2012
```

Exploratory Data Analysis

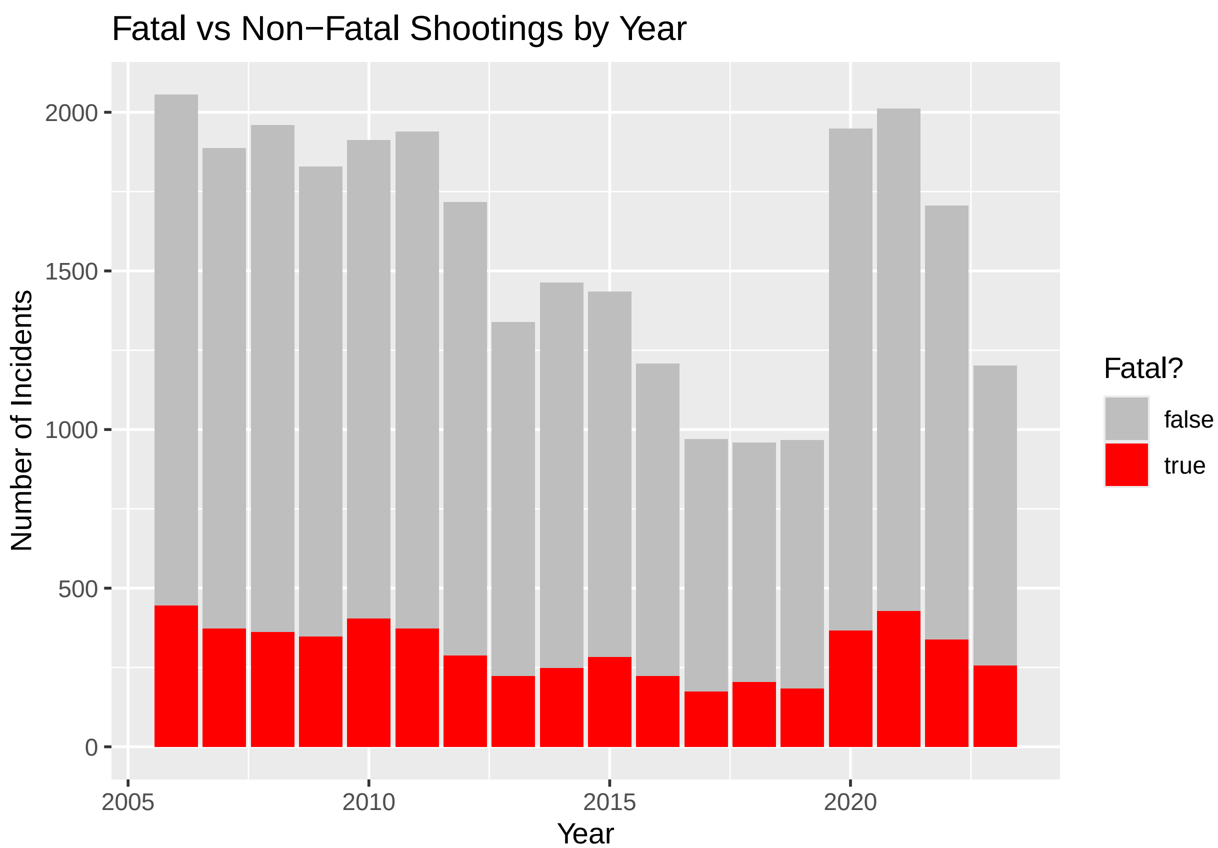
```r
# Summarize shootings by year
trends = data_clean %>%
  group_by(year) %>%
  summarise(n_shootings = n())

# Plot
ggplot(trends, aes(x = year, y = n_shootings)) +
  geom_line() + geom_point() +
  labs(title = "NYPD Shootings by Year (2006-2022)",
       x = "Year",
       y = "Number of Incidents") +
  theme_minimal()
```



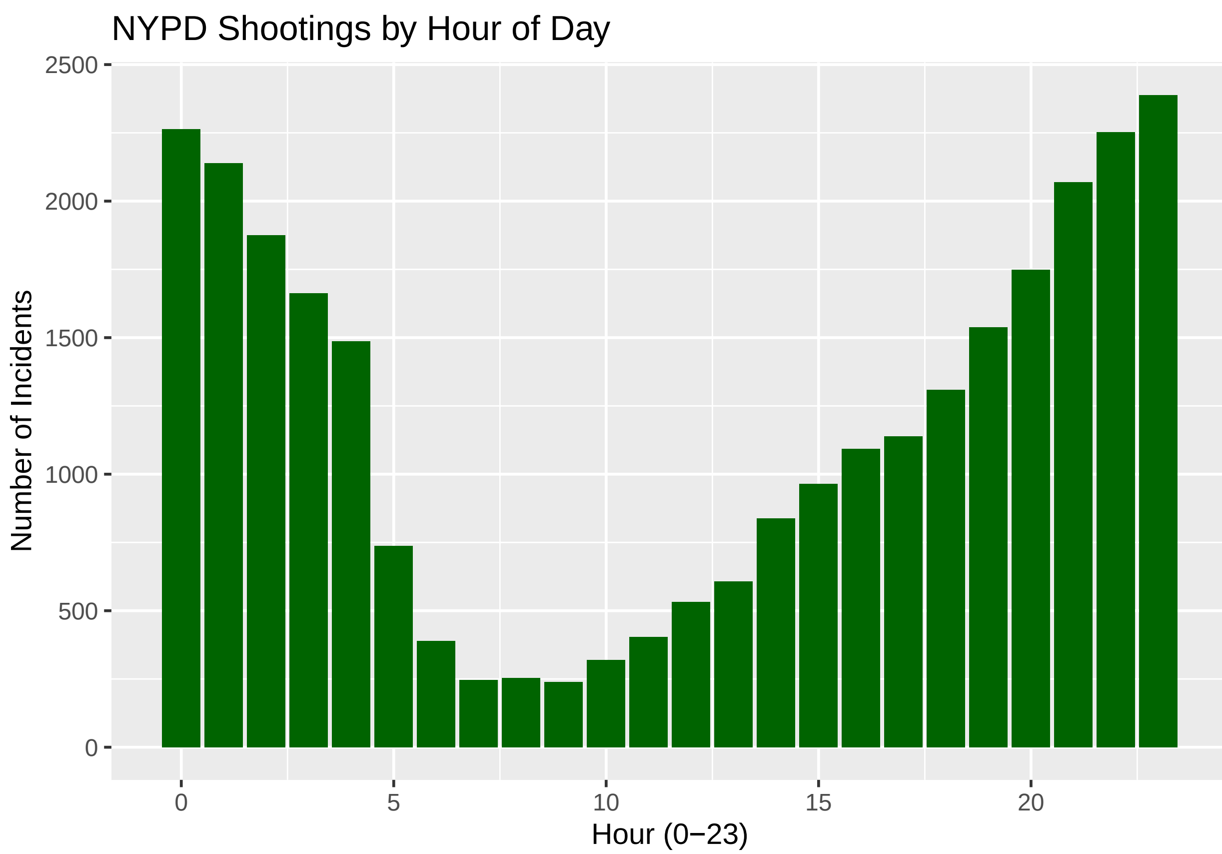

```r
# Summarize by borough
boro_trends = data_clean %>%
  group_by(BORO) %>%
  summarise(n_shootings = n())

# Bar plot
ggplot(boro_trends, aes(x = reorder(BORO, -n_shootings), y = n_shootings)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "NYPD Shootings by Borough",
       x = "Borough",
       y = "Number of Incidents") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
# Summarize fatalities by year
fatality_trends = data_clean %>%
  group_by(year, STATISTICAL_MURDER_FLAG) %>%
  summarise(n = n(), .groups = "drop")

# Plot
ggplot(fatality_trends, aes(x = year, y = n, fill = STATISTICAL_MURDER_FLAG)) +
  geom_bar(stat = "identity", position = "stack") +
  labs(title = "Fatal vs Non-Fatal Shootings by Year",
       x = "Year",
       y = "Number of Incidents",
       fill = "Fatal?") +
  scale_fill_manual(values = c("grey", "red"))
```




```r
# Hour
data_clean$hour = hour(data_clean$DATETIME)

# Summarize by hour
hourly = data_clean %>%
  group_by(hour) %>%
  summarise(n_shootings = n())

# Plot
ggplot(hourly, aes(x = hour, y = n_shootings)) +
  geom_bar(stat = "identity", fill = "darkgreen") +
  labs(title = "NYPD Shootings by Hour of Day",
       x = "Hour (0-23)",
       y = "Number of Incidents")
```



Data Modeling

```r
# Regression: Predict fatality bases on year, borough
# Ensure STATISTICAL_MURDER_FLAG is binary before splitting
data_clean$STATISTICAL_MURDER_FLAG = ifelse(data_clean$STATISTICAL_MURDER_FLAG == "true", 1, 0)

# Ensure no missing values in the dependent variable
data_clean = na.omit(data_clean)

# Split the data
set.seed(123)
trainIndex = createDataPartition(data_clean$STATISTICAL_MURDER_FLAG, p = 0.8, list = FALSE)
train_data = data_clean[trainIndex, ]

# Check unique values to confirm binary encoding
unique(train_data$STATISTICAL_MURDER_FLAG)
```

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

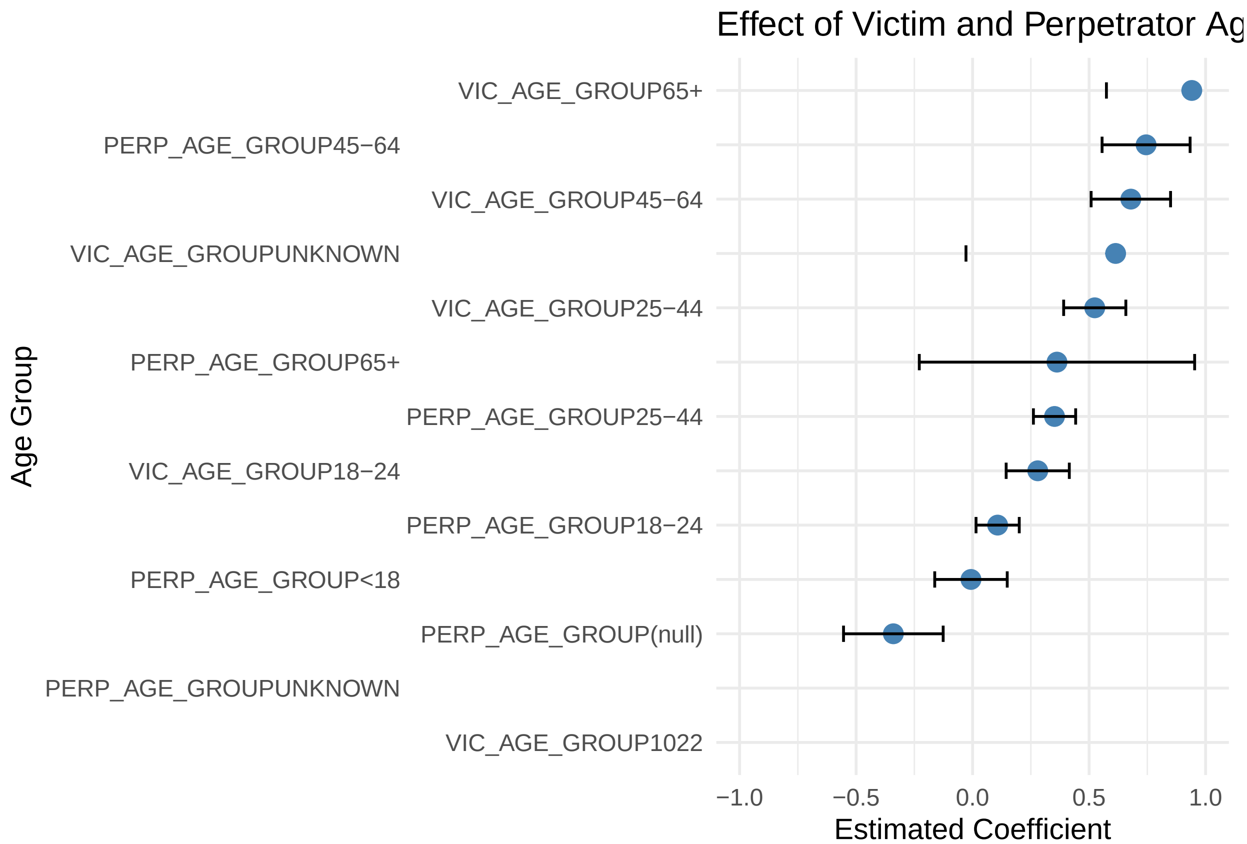
```r
# Fit logistic regression model
model = glm(STATISTICAL_MURDER_FLAG ~ year + I(year^2) + BORO + VIC_AGE_GROUP +
              PERP_AGE_GROUP + JURISDICTION_CODE,
              family = "binomial", data = train_data)

# Display model summary
summary(model)
```

```
##
## Call:
## glm(formula = STATISTICAL_MURDER_FLAG ~ year + I(year^2) + BORO +
##     VIC_AGE_GROUP + PERP_AGE_GROUP + JURISDICTION_CODE, family = "binomial",
##     data = train_data)
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)            2.713e+04  3.176e+03   8.543  < 2e-16 ***
## year                  -2.691e+01  3.153e+00  -8.533  < 2e-16 ***
## I(year^2)              6.671e-03  7.827e-04   8.523  < 2e-16 ***
## BOROBROOKLYN           5.298e-02  4.195e-02   1.263  0.20657
## BOROMANHATTAN         -1.111e-01  5.835e-02  -1.904  0.05696 .
## BOROQUEENS             4.781e-02  5.408e-02   0.884  0.37659
## BOROSTATEN ISLAND     -1.339e-02  1.047e-01  -0.128  0.89818
## VIC_AGE_GROUP1022     -8.707e+00  1.195e+02  -0.073  0.94190
## VIC_AGE_GROUP18-24     2.796e-01  6.916e-02   4.043 5.28e-05 ***
## VIC_AGE_GROUP25-44     5.243e-01  6.807e-02   7.702 1.34e-14 ***
## VIC_AGE_GROUP45-64     6.792e-01  8.700e-02   7.807 5.84e-15 ***
## VIC_AGE_GROUP65+       9.408e-01  1.869e-01   5.033 4.83e-07 ***
## VIC_AGE_GROUPUNKNOWN   6.138e-01  3.277e-01   1.873  0.06103 .
## PERP_AGE_GROUP(null)  -3.401e-01  1.091e-01  -3.117  0.00182 **
## PERP_AGE_GROUP<18     -6.974e-03  7.941e-02  -0.088  0.93002
## PERP_AGE_GROUP18-24    1.075e-01  4.732e-02   2.272  0.02308 *
## PERP_AGE_GROUP25-44    3.516e-01  4.625e-02   7.604 2.88e-14 ***
## PERP_AGE_GROUP45-64    7.447e-01  9.638e-02   7.726 1.11e-14 ***
## PERP_AGE_GROUP65+      3.621e-01  3.015e-01   1.201  0.22976
## PERP_AGE_GROUPUNKNOWN -2.211e+00  1.182e-01 -18.701  < 2e-16 ***
## JURISDICTION_CODE     -1.117e-01  2.483e-02  -4.497 6.88e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22457  on 22800  degrees of freedom
## Residual deviance: 21309  on 22780  degrees of freedom
## AIC: 21351
##
## Number of Fisher Scoring iterations: 9
```

```r
# plot of victim by age
coefficients = summary(model)$coefficients
age_coeffs = coefficients[grepl("VIC_AGE_GROUP|PERP_AGE_GROUP", rownames(coefficients)), ]
age_coeffs = as.data.frame(age_coeffs)
age_coeffs$Variable = rownames(age_coeffs)
ggplot(age_coeffs, aes(x = reorder(Variable, Estimate), y = Estimate)) +
  geom_point(color = "steelblue", size = 3) +
  geom_errorbar(aes(ymin = Estimate - 1.96 * `Std. Error`, ymax = Estimate + 1.96 * `Std. Error`),
              width = 0.3, color = "black") +
  labs(title = "Effect of Victim and Perpetrator Age Groups on Probability of Murder",
       x = "Age Group",
       y = "Estimated Coefficient") +
  theme_minimal() +
  coord_flip() +
```

```r
  scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
  scale_y_continuous(limits = c(-1, 1))  # Adjust limit as needed
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```r
# Machine Learning: XGBoost
# Prepare data: Convert categorical variables to factors
train_data = data_clean[trainIndex, ]
data_clean$VIC_SEX = as.factor(data_clean$VIC_SEX)
data_clean$PERP_AGE_GROUP = as.factor(data_clean$PERP_AGE_GROUP)
data_clean$PERP_SEX = as.factor(data_clean$PERP_SEX)
data_clean$JURISDICTION_CODE = as.factor(data_clean$JURISDICTION_CODE)

# Convert STATISTICAL_MURDER_FLAG to 0/1 (already verified)
data_clean$STATISTICAL_MURDER_FLAG = as.numeric(data_clean$STATISTICAL_MURDER_FLAG)

# Create train-test split
set.seed(123)
trainIndex = createDataPartition(data_clean$STATISTICAL_MURDER_FLAG, p = 0.8, list = FALSE)
train_data = data_clean[trainIndex, ]
test_data = data_clean[-trainIndex, ]

# Combine data to ensure consistent dummy variables
```

```r
combined_data = rbind(train_data[, c("year", "BORO", "hour", "VIC_AGE_GROUP", "VIC_SEX",
                                     "PERP_AGE_GROUP", "JURISDICTION_CODE")],
                      test_data[, c("year", "BORO", "hour", "VIC_AGE_GROUP", "VIC_SEX",
                                    "PERP_AGE_GROUP", "JURISDICTION_CODE")])
combined_matrix = model.matrix(~ . - 1, data = combined_data)

# Split back into train and test matrices
train_matrix = combined_matrix[1:nrow(train_data), ]
test_matrix = combined_matrix[(nrow(train_data) + 1):nrow(combined_matrix), ]

# Prepare XGBoost data
dtrain = xgb.DMatrix(data = train_matrix, label = train_data$STATISTICAL_MURDER_FLAG)
dtest = xgb.DMatrix(data = test_matrix, label = test_data$STATISTICAL_MURDER_FLAG)

# Set class weights to handle imbalance
scale_pos_weight = sum(train_data$STATISTICAL_MURDER_FLAG == 0) / sum(train_data$STATISTICAL_MURDER_FLA
print(paste("Scale pos weight:", scale_pos_weight))
```

```
## [1] "Scale pos weight: 4.14579101782893"
```

```r
# Define parameters
params <- list(
  objective = "binary:logistic",
  eval_metric = "logloss",
  eta = 0.3,
  max_depth = 6,
  subsample = 0.8,
  colsample_bytree = 0.8
)

# Train the model with early stopping
xgb_model = xgb.train(
  params = params,
  data = dtrain,
  nrounds = 100,
  watchlist = list(train = dtrain, test = dtest),
  scale_pos_weight = scale_pos_weight,
  early_stopping_rounds = 10,
  verbose = 1
)
```

```
## [1]  train-logloss:0.675733  test-logloss:0.678883
## Multiple eval metrics are present. Will use test_logloss for early stopping.
## Will train until test_logloss hasn't improved in 10 rounds.
##
## [2]  train-logloss:0.666825  test-logloss:0.671584
## [3]  train-logloss:0.654770  test-logloss:0.662192
## [4]  train-logloss:0.652059  test-logloss:0.661748
## [5]  train-logloss:0.644174  test-logloss:0.656148
## [6]  train-logloss:0.636689  test-logloss:0.650073
## [7]  train-logloss:0.634146  test-logloss:0.648824
## [8]  train-logloss:0.630032  test-logloss:0.646163
## [9]  train-logloss:0.628750  test-logloss:0.646424
```

```
## [10] train-logloss:0.626646  test-logloss:0.646427
## [11] train-logloss:0.624879  test-logloss:0.645863
## [12] train-logloss:0.622302  test-logloss:0.644709
## [13] train-logloss:0.620850  test-logloss:0.644638
## [14] train-logloss:0.619872  test-logloss:0.644812
## [15] train-logloss:0.618211  test-logloss:0.643833
## [16] train-logloss:0.614812  test-logloss:0.640925
## [17] train-logloss:0.613929  test-logloss:0.641387
## [18] train-logloss:0.610917  test-logloss:0.639666
## [19] train-logloss:0.608302  test-logloss:0.638213
## [20] train-logloss:0.606756  test-logloss:0.637423
## [21] train-logloss:0.603591  test-logloss:0.635507
## [22] train-logloss:0.601887  test-logloss:0.635161
## [23] train-logloss:0.601174  test-logloss:0.635130
## [24] train-logloss:0.600230  test-logloss:0.635185
## [25] train-logloss:0.599589  test-logloss:0.635208
## [26] train-logloss:0.598014  test-logloss:0.634963
## [27] train-logloss:0.596615  test-logloss:0.633659
## [28] train-logloss:0.595588  test-logloss:0.633168
## [29] train-logloss:0.594380  test-logloss:0.633148
## [30] train-logloss:0.592612  test-logloss:0.632398
## [31] train-logloss:0.590773  test-logloss:0.631041
## [32] train-logloss:0.590598  test-logloss:0.631496
## [33] train-logloss:0.590248  test-logloss:0.631868
## [34] train-logloss:0.588931  test-logloss:0.631134
## [35] train-logloss:0.588333  test-logloss:0.631221
## [36] train-logloss:0.586298  test-logloss:0.630685
## [37] train-logloss:0.586877  test-logloss:0.631991
## [38] train-logloss:0.586585  test-logloss:0.632616
## [39] train-logloss:0.585317  test-logloss:0.632325
## [40] train-logloss:0.581430  test-logloss:0.629156
## [41] train-logloss:0.580470  test-logloss:0.629077
## [42] train-logloss:0.579305  test-logloss:0.629814
## [43] train-logloss:0.578079  test-logloss:0.628993
## [44] train-logloss:0.578260  test-logloss:0.629783
## [45] train-logloss:0.576872  test-logloss:0.629240
## [46] train-logloss:0.574644  test-logloss:0.628255
## [47] train-logloss:0.573067  test-logloss:0.628077
## [48] train-logloss:0.571003  test-logloss:0.626830
## [49] train-logloss:0.570043  test-logloss:0.626442
## [50] train-logloss:0.569036  test-logloss:0.625966
## [51] train-logloss:0.568005  test-logloss:0.625625
## [52] train-logloss:0.567050  test-logloss:0.625382
## [53] train-logloss:0.566667  test-logloss:0.624970
## [54] train-logloss:0.565393  test-logloss:0.624880
## [55] train-logloss:0.565330  test-logloss:0.625965
## [56] train-logloss:0.565997  test-logloss:0.627305
## [57] train-logloss:0.565371  test-logloss:0.627062
## [58] train-logloss:0.564129  test-logloss:0.626424
## [59] train-logloss:0.562979  test-logloss:0.626637
## [60] train-logloss:0.560940  test-logloss:0.626579
## [61] train-logloss:0.561075  test-logloss:0.627239
## [62] train-logloss:0.559041  test-logloss:0.625905
## [63] train-logloss:0.559803  test-logloss:0.627883
```

```
## [64] train-logloss:0.557904  test-logloss:0.626666
## Stopping. Best iteration:
## [54] train-logloss:0.565393  test-logloss:0.624880
```

```
# Predict on test set
test_data$pred = predict(xgb_model, dtest)
test_data$pred_class = ifelse(test_data$pred > 0.5, 1, 0)  # Default threshold

# Confusion matrix
confusionMatrix(as.factor(test_data$pred_class), as.factor(test_data$STATISTICAL_MURDER_FLAG), positive
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2869  423
##          1 1740  668
##
##                Accuracy : 0.6205
##                  95% CI : (0.6078, 0.6331)
##     No Information Rate : 0.8086
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1607
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6123
##             Specificity : 0.6225
##          Pos Pred Value : 0.2774
##          Neg Pred Value : 0.8715
##              Prevalence : 0.1914
##          Detection Rate : 0.1172
##    Detection Prevalence : 0.4225
##       Balanced Accuracy : 0.6174
##
##        'Positive' Class : 1
##
```

```
xgb_cv = xgb.cv(
  params = params,
  data = dtrain,
  nrounds = 100,
  nfold = 5,
  scale_pos_weight = scale_pos_weight,
  early_stopping_rounds = 10,
  verbose = 1
)
```

```
## [1]  train-logloss:0.668044+0.001271 test-logloss:0.669759+0.001842
## Multiple eval metrics are present. Will use test_logloss for early stopping.
## Will train until test_logloss hasn't improved in 10 rounds.
##
```

```
## [2]  train-logloss:0.656160+0.002206 test-logloss:0.659499+0.002007
## [3]  train-logloss:0.648392+0.002358 test-logloss:0.653640+0.002866
## [4]  train-logloss:0.642394+0.002561 test-logloss:0.649482+0.003611
## [5]  train-logloss:0.638391+0.002112 test-logloss:0.646977+0.003799
## [6]  train-logloss:0.633263+0.002071 test-logloss:0.643629+0.002946
## [7]  train-logloss:0.629940+0.003612 test-logloss:0.641498+0.003004
## [8]  train-logloss:0.626769+0.003096 test-logloss:0.639906+0.002636
## [9]  train-logloss:0.623367+0.003259 test-logloss:0.637887+0.003139
## [10] train-logloss:0.620890+0.003104 test-logloss:0.636729+0.003846
## [11] train-logloss:0.618382+0.003402 test-logloss:0.635410+0.004266
## [12] train-logloss:0.616184+0.003616 test-logloss:0.634769+0.004460
## [13] train-logloss:0.614724+0.003046 test-logloss:0.634595+0.003337
## [14] train-logloss:0.612282+0.002641 test-logloss:0.633887+0.002970
## [15] train-logloss:0.610081+0.001703 test-logloss:0.633028+0.002987
## [16] train-logloss:0.607991+0.002176 test-logloss:0.631922+0.002867
## [17] train-logloss:0.605720+0.001852 test-logloss:0.630864+0.002859
## [18] train-logloss:0.603041+0.001857 test-logloss:0.629891+0.002547
## [19] train-logloss:0.601431+0.002516 test-logloss:0.629614+0.002770
## [20] train-logloss:0.599424+0.002737 test-logloss:0.629382+0.003005
## [21] train-logloss:0.597792+0.003035 test-logloss:0.629544+0.003264
## [22] train-logloss:0.595692+0.002640 test-logloss:0.629355+0.002820
## [23] train-logloss:0.593089+0.001920 test-logloss:0.627489+0.002719
## [24] train-logloss:0.592062+0.001762 test-logloss:0.627672+0.002597
## [25] train-logloss:0.591125+0.001751 test-logloss:0.627878+0.003224
## [26] train-logloss:0.589927+0.001427 test-logloss:0.627966+0.003205
## [27] train-logloss:0.587796+0.001709 test-logloss:0.627086+0.003390
## [28] train-logloss:0.586005+0.001579 test-logloss:0.627118+0.003706
## [29] train-logloss:0.584566+0.001916 test-logloss:0.626834+0.003696
## [30] train-logloss:0.582939+0.002019 test-logloss:0.625986+0.004375
## [31] train-logloss:0.582247+0.002455 test-logloss:0.626060+0.004400
## [32] train-logloss:0.581209+0.001998 test-logloss:0.626450+0.004391
## [33] train-logloss:0.579136+0.002502 test-logloss:0.626063+0.004201
## [34] train-logloss:0.577199+0.002116 test-logloss:0.625375+0.004161
## [35] train-logloss:0.575747+0.001841 test-logloss:0.625332+0.004220
## [36] train-logloss:0.574503+0.001322 test-logloss:0.625573+0.003986
## [37] train-logloss:0.573527+0.001497 test-logloss:0.625484+0.003558
## [38] train-logloss:0.572612+0.001168 test-logloss:0.625742+0.003585
## [39] train-logloss:0.571610+0.001361 test-logloss:0.625815+0.003312
## [40] train-logloss:0.569853+0.000704 test-logloss:0.625258+0.004068
## [41] train-logloss:0.568360+0.000989 test-logloss:0.624659+0.004004
## [42] train-logloss:0.567782+0.001089 test-logloss:0.625038+0.004062
## [43] train-logloss:0.566235+0.001148 test-logloss:0.624512+0.004135
## [44] train-logloss:0.565375+0.001334 test-logloss:0.624911+0.005173
## [45] train-logloss:0.564384+0.000564 test-logloss:0.624873+0.004531
## [46] train-logloss:0.563677+0.000790 test-logloss:0.624867+0.004030
## [47] train-logloss:0.563176+0.001387 test-logloss:0.625502+0.004465
## [48] train-logloss:0.561248+0.001534 test-logloss:0.624508+0.004498
## [49] train-logloss:0.559601+0.001193 test-logloss:0.623822+0.003966
## [50] train-logloss:0.558677+0.001485 test-logloss:0.623492+0.004381
## [51] train-logloss:0.558061+0.001399 test-logloss:0.623512+0.004196
## [52] train-logloss:0.557339+0.001272 test-logloss:0.623654+0.004868
## [53] train-logloss:0.556575+0.000941 test-logloss:0.624089+0.004199
## [54] train-logloss:0.555600+0.001010 test-logloss:0.624110+0.004338
## [55] train-logloss:0.554251+0.001792 test-logloss:0.623352+0.004719
```

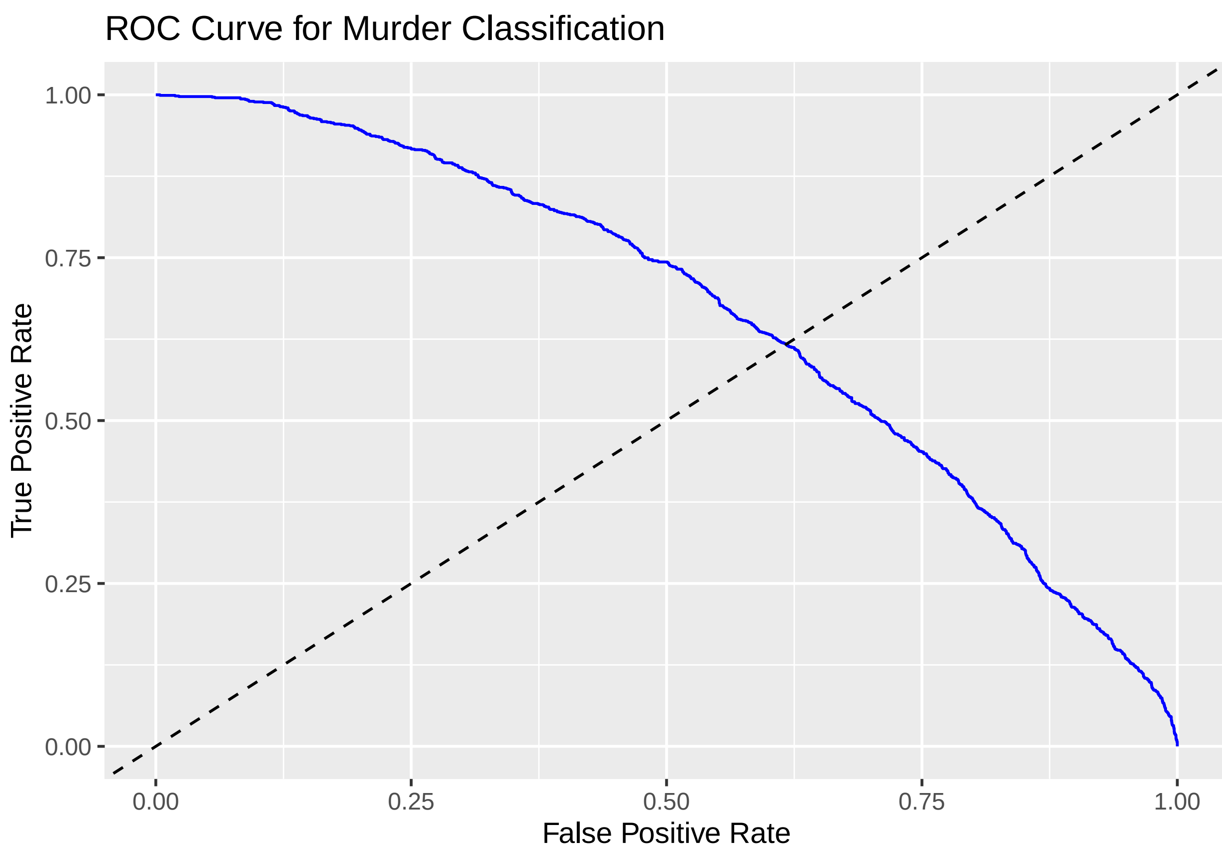
```
## [56] train-logloss:0.553375+0.001549 test-logloss:0.623244+0.004801
## [57] train-logloss:0.552733+0.001228 test-logloss:0.623312+0.004516
## [58] train-logloss:0.551936+0.002220 test-logloss:0.623331+0.004313
## [59] train-logloss:0.550993+0.002247 test-logloss:0.623405+0.004640
## [60] train-logloss:0.549735+0.002291 test-logloss:0.623323+0.003872
## [61] train-logloss:0.548272+0.002006 test-logloss:0.623056+0.004569
## [62] train-logloss:0.547648+0.001801 test-logloss:0.623353+0.004885
## [63] train-logloss:0.546388+0.001518 test-logloss:0.622785+0.005293
## [64] train-logloss:0.545435+0.001121 test-logloss:0.622778+0.005185
## [65] train-logloss:0.545094+0.001057 test-logloss:0.622979+0.005550
## [66] train-logloss:0.543879+0.000529 test-logloss:0.622625+0.005689
## [67] train-logloss:0.543174+0.000806 test-logloss:0.622629+0.004535
## [68] train-logloss:0.542574+0.001254 test-logloss:0.622679+0.004470
## [69] train-logloss:0.541529+0.001191 test-logloss:0.622590+0.004379
## [70] train-logloss:0.540091+0.000720 test-logloss:0.621949+0.005168
## [71] train-logloss:0.539424+0.000904 test-logloss:0.622382+0.004806
## [72] train-logloss:0.538602+0.001036 test-logloss:0.622434+0.004818
## [73] train-logloss:0.537959+0.000973 test-logloss:0.622402+0.005060
## [74] train-logloss:0.537524+0.000721 test-logloss:0.623007+0.005579
## [75] train-logloss:0.537346+0.000859 test-logloss:0.623456+0.006187
## [76] train-logloss:0.536427+0.001192 test-logloss:0.623523+0.006885
## [77] train-logloss:0.535418+0.001567 test-logloss:0.623329+0.007434
## [78] train-logloss:0.534305+0.001283 test-logloss:0.623099+0.006967
## [79] train-logloss:0.533654+0.001335 test-logloss:0.622985+0.006618
## [80] train-logloss:0.532632+0.000926 test-logloss:0.622388+0.006014
## Stopping. Best iteration:
## [70] train-logloss:0.540091+0.000720 test-logloss:0.621949+0.005168
```

```r
# Assuming test_data contains actual and predicted values
test_data$pred_prob <- predict(xgb_model, dtest)
test_data$pred_class <- ifelse(test_data$pred_prob > 0.5, 1, 0)

# ROC Curve
roc_obj <- roc(test_data$STATISTICAL_MURDER_FLAG, test_data$pred_prob)
```

```
## Setting levels: control = 0, case = 1
```

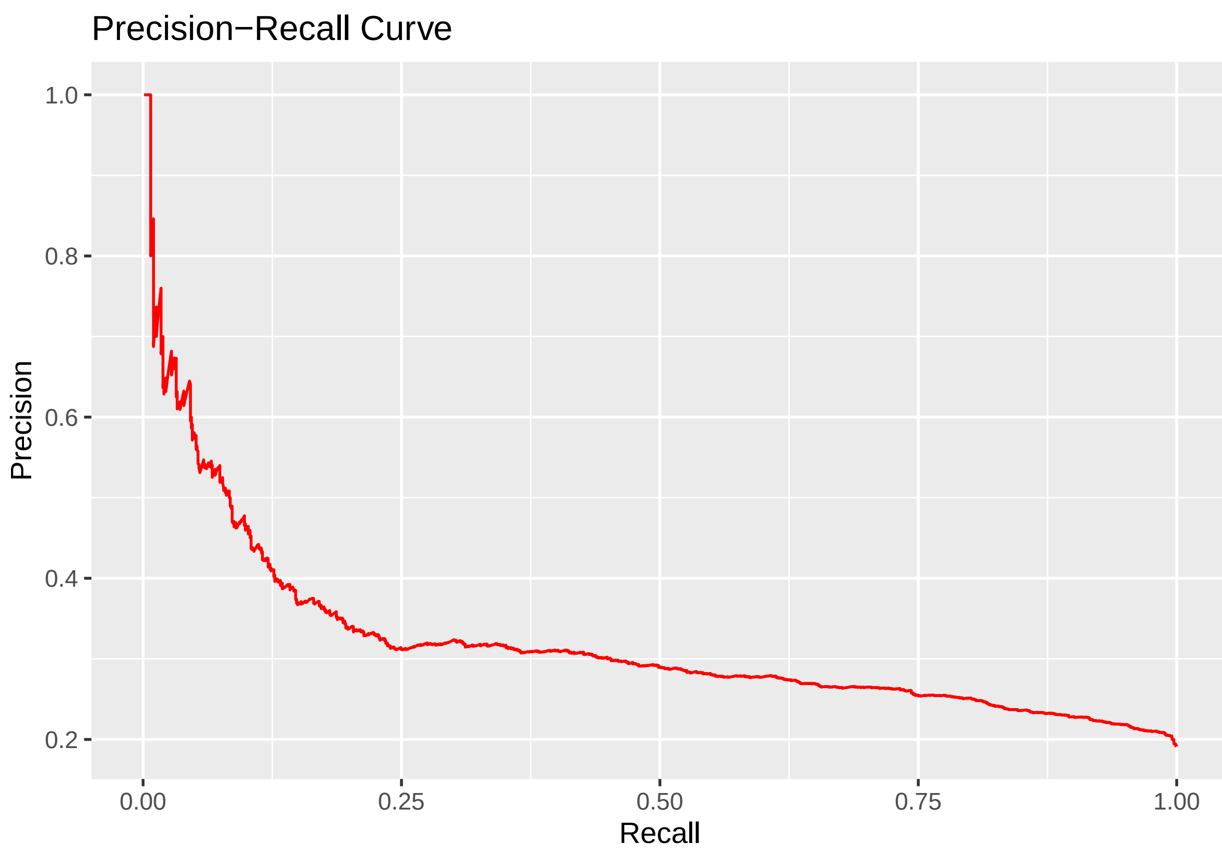
```
## Setting direction: controls < cases
```

```r
ggplot() +
  geom_line(aes(x = roc_obj$specificities, y = roc_obj$sensitivities), color = 'blue') +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed") +
  labs(title = "ROC Curve for Murder Classification", x = "False Positive Rate", y = "True Positive Rate
```

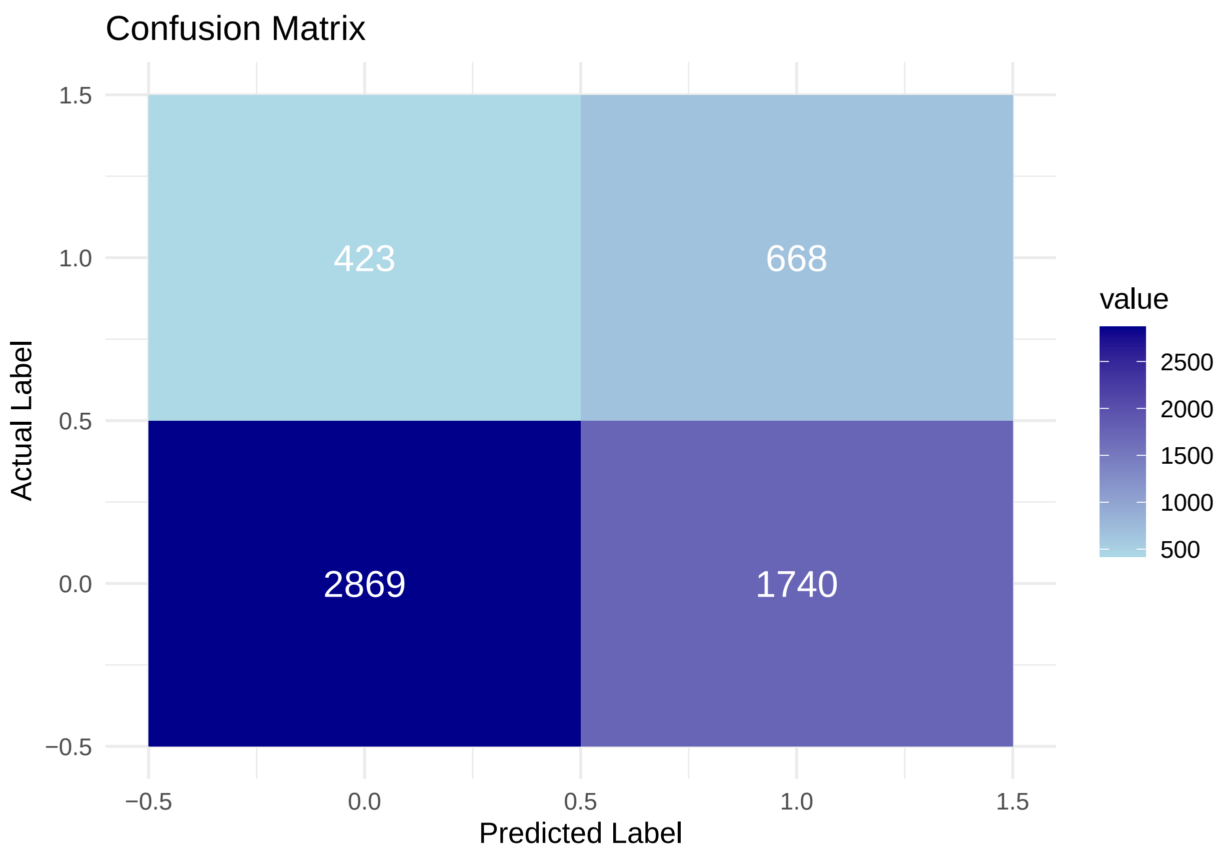



```
# Precision-Recall Curve
pr <- prediction(test_data$pred_prob, test_data$STATISTICAL_MURDER_FLAG)
pr_curve <- performance(pr, "prec", "rec")
pr_df <- data.frame(recall = unlist(pr_curve@x.values), precision = unlist(pr_curve@y.values))

ggplot(pr_df, aes(x = recall, y = precision)) +
  geom_line(color = "red") +
  labs(title = "Precision-Recall Curve", x = "Recall", y = "Precision")
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```




```
# Confusion Matrix
conf_matrix <- table(Predicted = test_data$pred_class, Actual = test_data$STATISTICAL_MURDER_FLAG)
conf_matrix_melt <- melt(conf_matrix)

ggplot(conf_matrix_melt, aes(x = Predicted, y = Actual, fill = value)) +
  geom_tile() +
  geom_text(aes(label = value), color = "white", size = 5) +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(title = "Confusion Matrix", x = "Predicted Label", y = "Actual Label") +
  theme_minimal()
```



Evaluate / Interpret

NYPD Shootings by Borough

Brooklyn: exceeds 9,000 incidents, Bronx: around 7,000-8,000 incidents, Queens: around 4,000-5,000 incidents, Manhattan: around 3,000-4,000 incidents, Staten Island: below 2,000 incidents.

Brooklyn and Bronx together account for the majority of shootings. Brooklyn and Bronx have larger populations, which might contribute to higher incident counts. Staten Island's low count could indicate safer conditions or fewer reported incidents, possibly due to its smaller size

Limitations: The chart shows total incidents over the dataset's timeframe (2006-2022). It doesn't account for population density or time trends.

Fatal vs Non-Fatal Shootings by Year

2006-2009: fatal shootings around 500-600 per year, 2010-2016: fatal shootings 200-300 per year, 2017-2020: fatial shootings 400-500, 2021-2022: fatal shooting around 400.

Fatal shootings consistently make up a smaller portion of total incidents, roughly 20-30% each year. The proportion of fatal shootings appears relatively stable over time, despite fluctuations in total incidents.

Limitations: While the stacked bars show raw counts, the proportion of fatal shootings isn't immediately clea

NYPD Shootings by Hour of Day

0-3 AM: 2000-2500 incidents, 4-7 AM: 500 incidents, 8 AM - noon: 1000-1500 incidents, noon-5 PM: 1500 incidents, 6PM-11PM: 2000-2500 incidents.

The majority of shootings occur during nighttime and early morning hours, with the lowest activity during the early morning (4-7 AM), likely corresponding to lower population activity. The two peaks (0-3 AM and 8-11 PM) suggest times of higher social activity or vulnerability, possibly linked to nightlife, late-night gatherings, or reduced visibility/policing.

Limitations: The chart shows total incidents but doesn't differentiate by factors like borough, fatality, or day of the week, which could provide deeper insights

Logistic Regression

Age Effect: Older victims (65+) are 3.18 (0.9408, $p < 0.001$) times more likely to die, possibly due to physical vulnerability or delayed medical response. Age 18-24: Increases the log-odds by 0.2796 ($p < 0.001$). Age 25-44: Increases the log-odds by 0.5243 ($p < 0.001$). Age 45-64: Stronger effect (0.6792, $p < 0.001$).Time Trend: The U-shaped trend (lowest ~2014-2015) suggests external factors (e.g., policing, social unrest) influenced fatality rates post-2015. There did not appear to be any significant effect of death based on borough. Perpetrators aged 18-24 and 25-44 are more likely to be involved in shootings that result in murder ($p < 0.05$).Perpetrators aged 45-64 have the strongest effect (0.7447, $p < 0.001$).Null deviance vs. Residual deviance: The model explains some variation in the data, but there is still room for improvement.AIC: Lower AIC values indicate a better model.

Summary

The evaluation of the NYPD Shooting Data visualizations reveals distinct patterns across the charts. For NYPD Shootings by Borough, Brooklyn exceeds 9,000 incidents, followed by Bronx with 7,000-8,000, Queens with 4,000-5,000, Manhattan with 3,000-4,000, and Staten Island below 2,000. Brooklyn and Bronx together account for the majority of shootings, likely influenced by their larger populations, while Staten Island's low count may indicate safer conditions or fewer reported incidents, possibly due to its smaller size. However, the chart, covering total incidents from 2006 to 2022, does not account for population density or time trends, limiting its depth.

For Fatal vs Non-Fatal Shootings by Year, fatal shootings range from 500-600 per year in 2006-2009, drop to 200-300 in 2010-2016, rise to 400-500 in 2017-2020, and stabilize around 400 in 2021-2022, consistently making up 20-30% of total incidents with a stable proportion despite fluctuations. The limitation here is that raw counts obscure exact proportions, and the

inclusion of 2005 data may be incomplete. Interestingly enough, the borough didn't matter when evaluating for fatal vs non-fatal. Also, older age of the vitim and perpetrater were also associated with an increased odds of fatality.

Lastly, NYPD Shootings by Hour of Day shows peaks of 2,000-2,500 incidents at 0-3 AM and 6-11 PM, a dip to around 500 incidents at 4-7 AM, and 1,000-1,500 incidents midday, with nighttime peaks suggesting higher social activity or vulnerability and the low morning activity aligning with reduced population presence. The chart's limitation is its lack of differentiation by borough, fatality, or day of week, which could provide deeper insights.

Murder classification likelihood has decreased over time, but there may be a non-linear trend. Older victims and perpetrators significantly increase classification probability. Missing perpetrator age significantly decreases the probability of statistical murder classification.Borough is not a strong predictor.

Of note, the machine learning prediction did not significantly improve the model over linear regression models. The model correctly classifies ~62% of the cases.The model captures ~61% of the actual murders. The model captures ~62% of non-murder cases. Only 27.74% of cases predicted as murder are actually murders.87.15% of cases predicted as non-murder are correct. McNemar's Test ($p < 2e\text{-}16$) Suggests a significant difference between how the model predicts Class 0 vs. Class 1. Kappa = 0.1607 Indicates the model is only slightly better than random guessing.The model struggles with correctly identifying murder cases (Class 1), which is expected given the class imbalance.

Potential biases include population bias where higher counts in Brooklyn and Bronx may reflect population size rather than crime rate per capita, reporting bias where lower counts in Staten Island could result from underreporting or fewer police resources, and geographic bias where urban density differences are not normalized in the borough analysis. For fatal vs. non-fatal shootings, there may be data collection bias from variations in medical response or reporting standards affecting fatality classification, temporal bias from aggregated data masking yearly shifts like the 2020 COVID impact, and definition bias where the "fatal" definition may vary and skew trends. In the hourly analysis, activity bias may overrepresent nightlife areas at 0-3 AM and 8-11 PM while underrepresenting daytime crime, reporting bias might lower 4-7 AM counts due to fewer witnesses or patrols, and temporal aggregation bias averages over 2006-2022, ignoring seasonal or yearly variations like post-2020 changes. Overall, the data aggregation across long periods, absence of normalization, and lack of socioeconomic or policing context introduce potential confounding biases that could be mitigated with per-capita adjustments, faceting by additional factors, and validation with external data.