AMS 597 - Statistical Computing

GROUP1: Varsha, Swati, Dhruv, Kabir, Iftekhar

2025-04-11

```r
# Load libraries for data manipulation, visualization, modeling, and evaluation

library(tidyverse,quietly = T)
```

```
## -- 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 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- 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 error
```

```r
library(RColorBrewer,quietly = T)
library(dlookr,quietly = T)
```

```
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
```

```
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

```
## Registered S3 methods overwritten by 'dlookr':
##   method          from
##   plot.transform  scales
##   print.transform scales
##
## Attaching package: 'dlookr'
##
## The following object is masked from 'package:tidyr':
##
##     extract
##
## The following object is masked from 'package:base':
##
##     transform
```

```
library(ggcorrplot,quietly = T)
library(plyr,quietly = T)
```

```
## ------------------------------------------------------------------------------
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
## ------------------------------------------------------------------------------
##
## Attaching package: 'plyr'
##
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
##
## The following object is masked from 'package:purrr':
##
##     compact
```

```
library(dplyr,quietly = T)
library(cowplot,quietly = T)
```

```
##
## Attaching package: 'cowplot'
##
## The following object is masked from 'package:lubridate':
##
##     stamp
```

```
library(ggplot2,quietly = T)
library(gridExtra,quietly = T)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(readr,quietly = T)
library(lattice,quietly = T)
library(magrittr,quietly = T)
```

```
##
## Attaching package: 'magrittr'
##
## The following object is masked from 'package:dlookr':
##
##     extract
##
## The following object is masked from 'package:purrr':
##
##     set_names
##
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(ggmap,quietly = T)
```

```
## i Google's Terms of Service: <https://mapsplatform.google.com>
##   Stadia Maps' Terms of Service: <https://stadiamaps.com/terms-of-service/>
##   OpenStreetMap's Tile Usage Policy: <https://operations.osmfoundation.org/policies/tiles/>
## i Please cite ggmap if you use it! Use `citation("ggmap")` for details.
##
## Attaching package: 'ggmap'
##
##
## The following object is masked from 'package:magrittr':
##
```

```
##     inset
##
##
## The following object is masked from 'package:cowplot':
##
##     theme_nothing
```

```
library(hexbin)
library(viridis)
```

```
## Loading required package: viridisLite
```

```
library(ggthemes)
```

```
##
## Attaching package: 'ggthemes'
##
## The following object is masked from 'package:cowplot':
##
##     theme_map
```

```
library(rsample,quietly = T)
library(rpart,quietly = T)
library(nnet,quietly = T)
library(caret,quietly = T)
```

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

```
library(party,quietly = T)
```

```
##
## Attaching package: 'modeltools'
##
## The following object is masked from 'package:plyr':
##
##     empty
##
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
##
## Attaching package: 'strucchange'
```

```
##
## The following object is masked from 'package:stringr':
##
##     boundary
##
##
## Attaching package: 'party'
##
## The following object is masked from 'package:dplyr':
##
##     where
```

```
library(pROC,quietly = T)
```

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

```
library(ROCR,quietly = T)
library(randomForest,quietly = T)
```

```
## randomForest 4.7-1.2
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:gridExtra':
##
##     combine
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(class,quietly = T)
library(SimDesign,quietly = T)
```

```
##
## Attaching package: 'SimDesign'
##
## The following objects are masked from 'package:mvtnorm':
##
##     rmvnorm, rmvt
##
```

```
## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
```

```
library(rpart.plot,quietly = T)
library(e1071,quietly = T)
```

```
##
## Attaching package: 'e1071'
##
## The following object is masked from 'package:rsample':
##
##     permutations
##
## The following objects are masked from 'package:dlookr':
##
##     kurtosis, skewness
```

```
#install.packages("corrplot")
library(corrplot)
```

```
## corrplot 0.95 loaded
```

PHASE 1

EXPLORATORY DATA ANALYSIS

```
# Load NYC crime dataset from CSV file into a data frame
nycCrimeData = read.csv("/Users/swati/Downloads/Project_nycCRIME/NYPD_Complaint_Data_Current_YTD.csv")
#head(nycCrimeData)
```

```
# Get the current working directory

getwd()
```

```
## [1] "/Users/swati/Desktop/Project_nycCRIME"
```

```
# Get the dimensions of the NYC crime dataset (number of rows and columns)

dim(nycCrimeData)
```

```
## [1] 361740     24
```

```
# Get the column names of the NYC crime dataset
colnames(nycCrimeData)
```

```
##  [1] "CMPLNT_NUM"        "CMPLNT_FR_DT"      "CMPLNT_FR_TM"
##  [4] "CMPLNT_TO_DT"      "CMPLNT_TO_TM"      "RPT_DT"
```

```
##  [7] "KY_CD"             "OFNS_DESC"         "PD_CD"
## [10] "PD_DESC"           "CRM_ATPT_CPTD_CD"  "LAW_CAT_CD"
## [13] "JURIS_DESC"        "BORO_NM"           "ADDR_PCT_CD"
## [16] "LOC_OF_OCCUR_DESC" "PREM_TYP_DESC"     "PARKS_NM"
## [19] "HADEVELOPT"        "X_COORD_CD"        "Y_COORD_CD"
## [22] "Latitude"          "Longitude"         "Lat_Lon"
```

```
#glimpse(nycCrimeData)
summary(nycCrimeData)
```

```
##    CMPLNT_NUM        CMPLNT_FR_DT       CMPLNT_FR_TM       CMPLNT_TO_DT
##  Min.   :100009724   Length:361740      Length:361740      Length:361740
##  1st Qu.:324169675   Class :character   Class :character   Class :character
##  Median :550299675   Mode  :character   Mode  :character   Mode  :character
##  Mean   :549940294
##  3rd Qu.:775703401
##  Max.   :999999422
##
##  CMPLNT_TO_TM          RPT_DT              KY_CD        OFNS_DESC
##  Length:361740      Length:361740      Min.   :101.0   Length:361740
##  Class :character   Class :character   1st Qu.:118.0   Class :character
##  Mode  :character   Mode  :character   Median :341.0   Mode  :character
##                                        Mean   :299.6
##                                        3rd Qu.:351.0
##                                        Max.   :685.0
##
##      PD_CD          PD_DESC          CRM_ATPT_CPTD_CD    LAW_CAT_CD
##  Min.   :101.0   Length:361740      Length:361740      Length:361740
##  1st Qu.:254.0   Class :character   Class :character   Class :character
##  Median :357.0   Mode  :character   Mode  :character   Mode  :character
##  Mean   :411.9
##  3rd Qu.:638.0
##  Max.   :922.0
##  NA's   :263
##   JURIS_DESC          BORO_NM           ADDR_PCT_CD     LOC_OF_OCCUR_DESC
##  Length:361740      Length:361740      Min.   :  1.00   Length:361740
##  Class :character   Class :character   1st Qu.: 40.00   Class :character
##  Mode  :character   Mode  :character   Median : 63.00   Mode  :character
##                                        Mean   : 63.03
##                                        3rd Qu.: 94.00
##                                        Max.   :123.00
##                                        NA's   :1
##  PREM_TYP_DESC        PARKS_NM          HADEVELOPT          X_COORD_CD
##  Length:361740      Length:361740      Length:361740      Min.   : 913357
##  Class :character   Class :character   Class :character   1st Qu.: 991945
##  Mode  :character   Mode  :character   Mode  :character   Median :1004550
##                                                           Mean   :1005074
##                                                           3rd Qu.:1016781
##                                                           Max.   :1067226
##                                                           NA's   :5854
##    Y_COORD_CD        Latitude       Longitude        Lat_Lon
##  Min.   :121250   Min.   :40.50   Min.   :-74.25   Length:361740
##  1st Qu.:184359   1st Qu.:40.67   1st Qu.:-73.97   Class :character
##  Median :206483   Median :40.73   Median :-73.93   Mode  :character
```

```
##  Mean   :207404   Mean   :40.74   Mean   :-73.92
##  3rd Qu.:235493   3rd Qu.:40.81   3rd Qu.:-73.88
##  Max.   :271820   Max.   :40.91   Max.   :-73.70
##  NA's   :5854     NA's   :5854    NA's   :5854
```

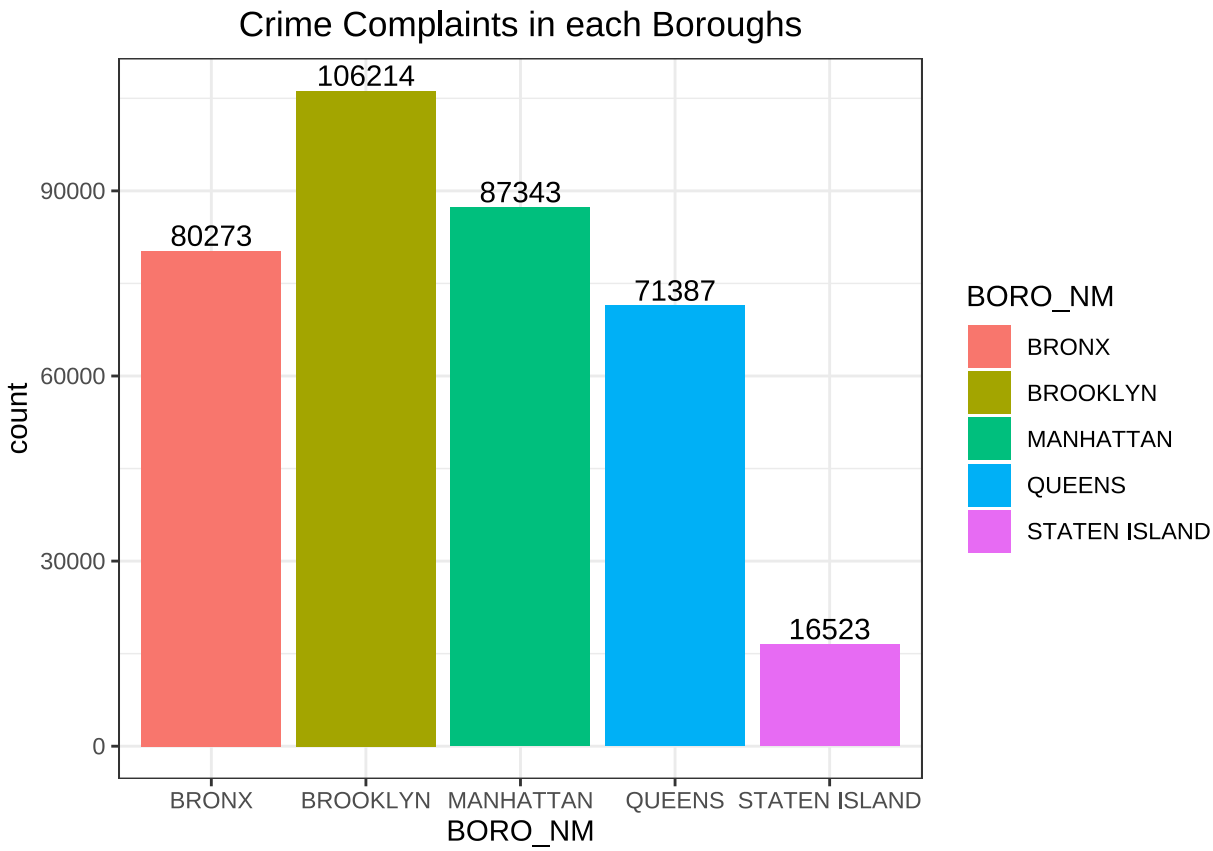
```
table(nycCrimeData$BORO_NM)
```

```
##
##         BRONX      BROOKLYN     MANHATTAN        QUEENS STATEN ISLAND
##         80273        106214         87343         71387         16523
```

```
#boro_table = table(nycCrimeData$BORO_NM)
#length(boro_table)
```

```
# Create a bar plot to visualize the distribution of crime complaints across boroughs

ggplot(nycCrimeData, aes(x = BORO_NM,fill = BORO_NM)) + geom_bar() + ggtitle("Crime Complaints in each
```

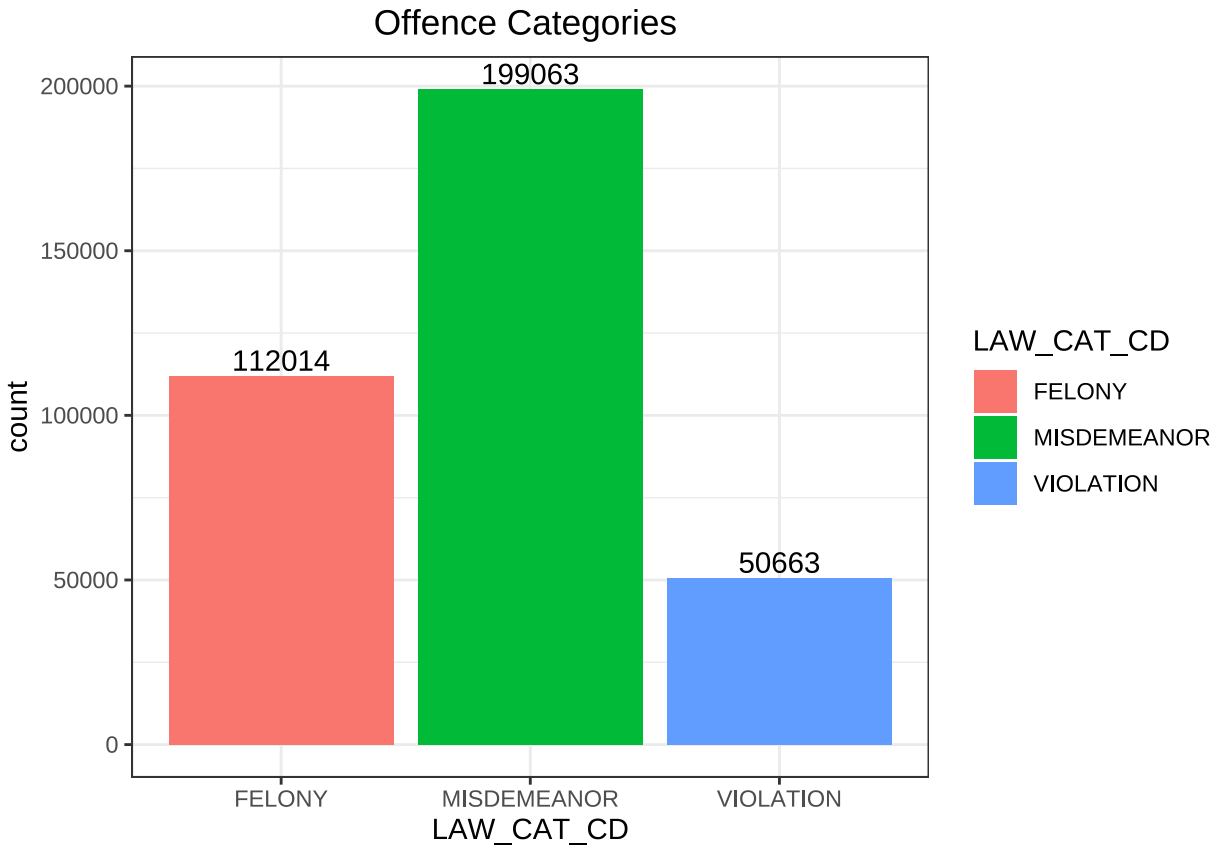
```
## Warning: The dot-dot notation (‘..count..‘) was deprecated in ggplot2 3.4.0.
## i Please use ‘after_stat(count)‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



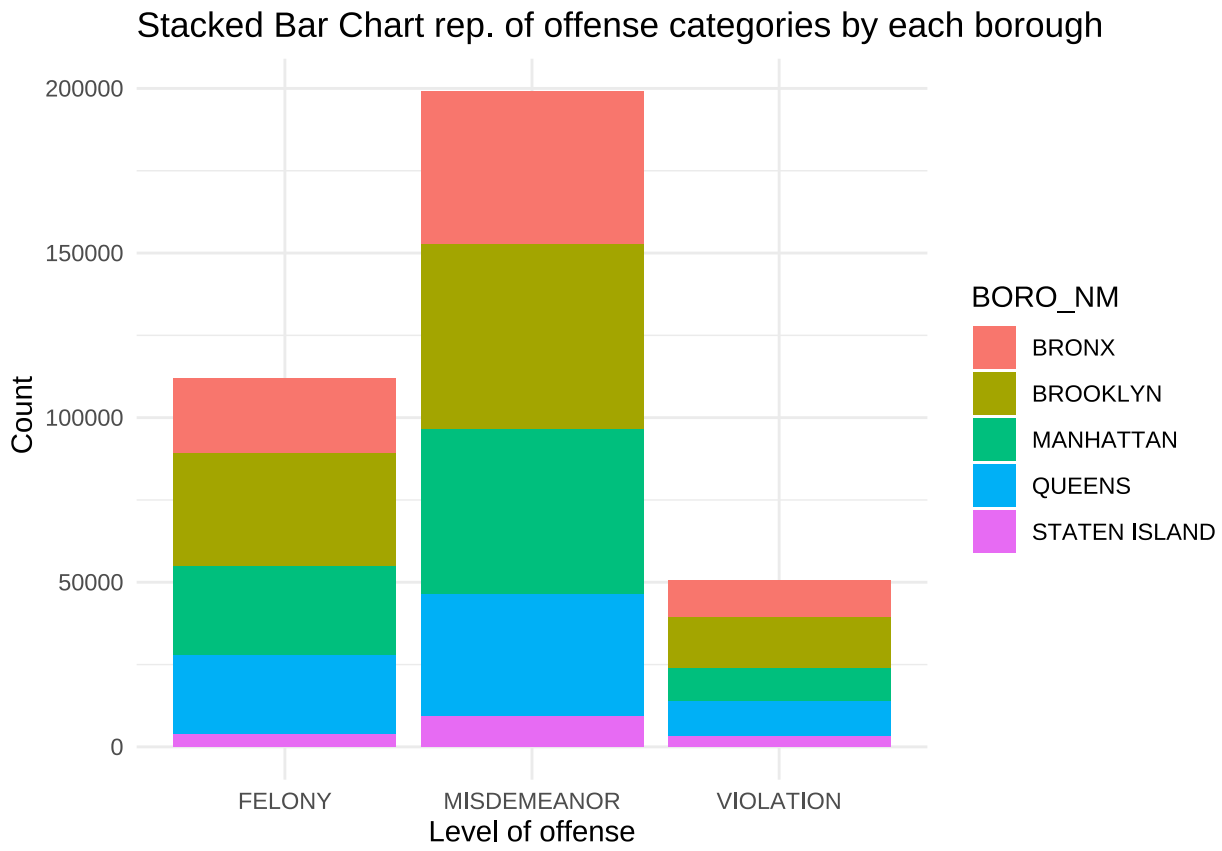
```r
# Create a frequency table for the offense categories (LAW_CAT_CD)

table(nycCrimeData$LAW_CAT_CD)
```

```
##
##      FELONY MISDEMEANOR   VIOLATION
##      112014      199063       50663
```

```r
# Create a bar plot to visualize the distribution of offense categories

ggplot(nycCrimeData, aes(x = LAW_CAT_CD,fill = LAW_CAT_CD)) + geom_bar() + ggtitle("Offence Categories
```

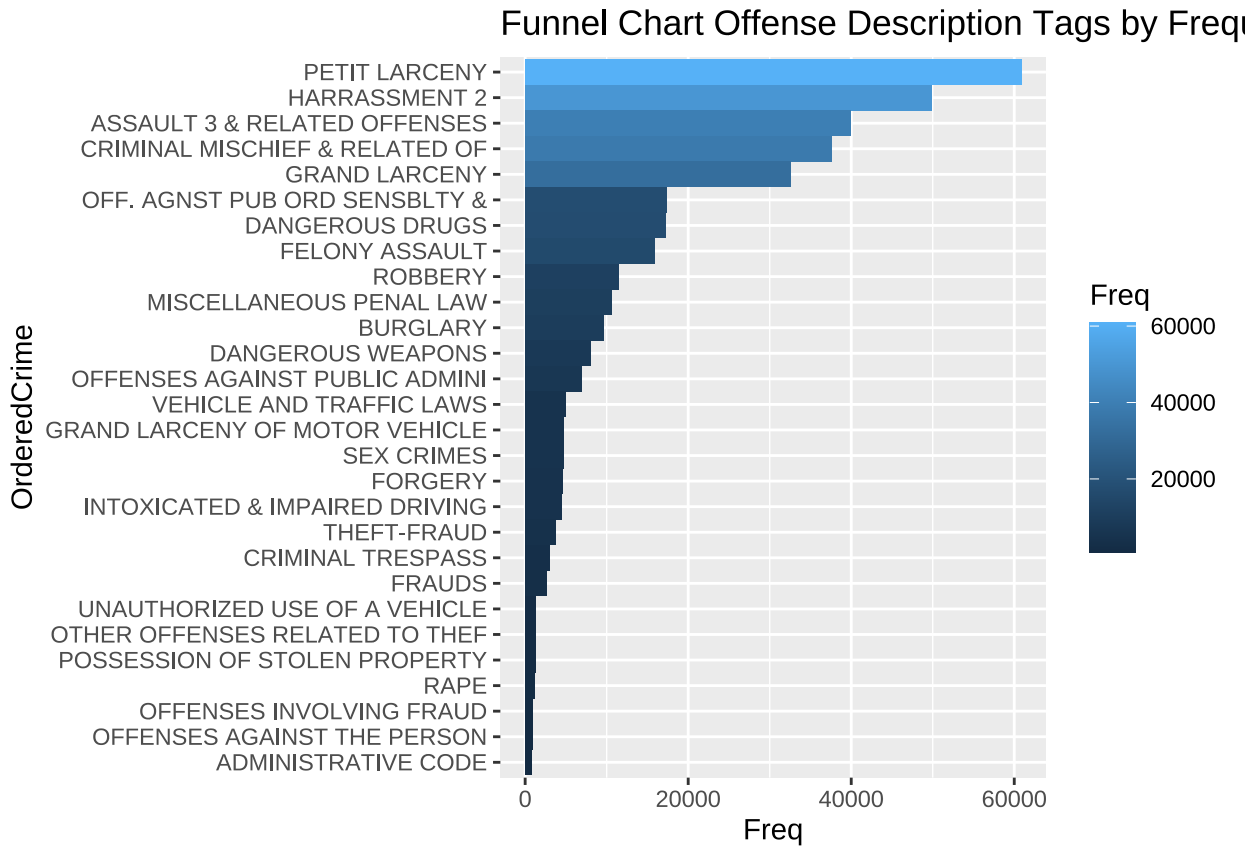



```r
# Create a stacked bar chart to visualize the distribution of offense categories by borough

ggplot(nycCrimeData, aes(x = LAW_CAT_CD,fill = BORO_NM)) + geom_bar() + ggtitle("Offense Categories") +
```




```
#table(nycCrimeData$OFNS_DESC)

#length(unique(nycCrimeData$OFNS_DESC))

#length(unique(nycCrimeData$OFNS_DESC))
```

```
# Create a funnel chart to visualize the most frequent crime descriptions with a frequency greater than

frequencies <- data.frame(table(nycCrimeData$OFNS_DESC))
names(frequencies)[1] <- "Crime"
frequencies <- frequencies[order(-frequencies$Freq),]

hifreq <- frequencies[which(frequencies$Freq > 659),]

hifreq$OrderedCrime <- reorder(hifreq$Crime, hifreq$Freq)

ofns.freqbp <- ggplot(hifreq, aes(x=OrderedCrime, y=Freq, fill=Freq)) +

  geom_bar(width=1, stat="identity") + ggtitle("Funnel Chart Offense Description Tags by Frequency") +

  coord_flip()

ofns.freqbp
```




```
# Analyze the frequency of police department descriptions for "DANGEROUS DRUGS" offense type

#length(unique(nycCrimeData$PD_DESC))

#head(nycCrimeData$PD_DESC)

drugs <- nycCrimeData[which(nycCrimeData$OFNS_DESC == "DANGEROUS DRUGS"),]
drugs.pd.summary <- data.frame(table(drugs$PD_DESC))
names(drugs.pd.summary)[1] <- "PD_DESC"
drugs.pd.summary <- drugs.pd.summary[which(drugs.pd.summary$Freq > 0),]
drugs.pd.summary <- drugs.pd.summary[order(-drugs.pd.summary$Freq),]
drugs.pd.summary
```

```
##                          PD_DESC Freq
## 14    MARIJUANA, POSSESSION 4 & 5 8121
## 2  CONTROLLED SUBSTANCE, POSSESSI 5186
## 5  CONTROLLED SUBSTANCE,INTENT TO 1403
## 6   CONTROLLED SUBSTANCE,POSSESS.  863
## 16          MARIJUANA, SALE 4 & 5  324
## 4    CONTROLLED SUBSTANCE, SALE 5  268
## 1  CONTROLLED SUBSTANCE, INTENT T  210
## 10    CONTROLLED SUBSTANCE,SALE 3  171
## 13 MARIJUANA, POSSESSION 1, 2 & 3  170
## 11 DRUG PARAPHERNALIA,   POSSESSE  162
## 8     CONTROLLED SUBSTANCE,SALE 1  141
## 15       MARIJUANA, SALE 1, 2 & 3   40
```

```
## 9     CONTROLLED SUBSTANCE,SALE 2   36
## 18 POSSESSION HYPODERMIC INSTRUME   36
## 19          SALE SCHOOL GROUNDS 4   26
## 3    CONTROLLED SUBSTANCE, SALE 4   23
## 7  CONTROLLED SUBSTANCE,POSSESS.-    5
## 12             DRUG, INJECTION OF    1
## 17    POSS METH MANUFACT MATERIAL    1
## 20          SALES OF PRESCRIPTION    1
```

```
# Filter the data for specific drug-related incidents (e.g., marijuana or alcohol),
# calculate the total frequency for the selected category, and compute its proportion
# relative to the total frequency of all drug-related incidents.

marijuana.pd <- drugs.pd.summary[grepl("MARIJUANA", drugs.pd.summary[,"PD_DESC"]),]
marijuana.sum <-  sum(marijuana.pd[,"Freq"])
drugs.sum <- sum(drugs.pd.summary[,"Freq"])
marijuana.sum / drugs.sum
```

```
## [1] 0.503549
```

```
marijuana.pd <- drugs.pd.summary[grepl("ALCOHOL", drugs.pd.summary[,"PD_DESC"]),]
marijuana.sum <-  sum(marijuana.pd[,"Freq"])
drugs.sum <- sum(drugs.pd.summary[,"Freq"])
marijuana.sum / drugs.sum
```

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

```
# Filter the unique descriptions of 'PD_DESC' for crimes categorized under "MISCELLANEOUS PENAL LAW"

# Combine complaint date and time into a single datetime object for start and end times
# Create additional time-based features like hour, day of month, month, and day of week for further ana

# Remove unnecessary columns from the dataset for a cleaner dataset

unique(nycCrimeData[which(nycCrimeData$OFNS_DESC == "MISCELLANEOUS PENAL LAW"),'PD_DESC'])
```

```
##  [1] "CRIMINAL CONTEMPT 1"
##  [2] "RECKLESS ENDANGERMENT 1"
##  [3] "BRIBERY,PUBLIC ADMINISTRATION"
##  [4] "AGGRAVATED CRIMINAL CONTEMPT"
##  [5] "PUBLIC ADMINISTRATION,UNCLASSI"
##  [6] "FALSE REPORT 1,FIRE"
##  [7] "IMPERSONATION 1, POLICE OFFICE"
##  [8] "UNAUTHORIZED USE VEHICLE 2"
##  [9] "MARIJUANA, POSSESSION"
## [10] "TRESPASS 4,CRIMINAL SUB 2"
## [11] "MAKING TERRORISTIC THREAT"
## [12] "THEFT OF SERVICES- CABLE TV SE"
## [13] "AGGRAVATED HARASSMENT 1"
## [14] "COERCION 1"
## [15] "BIGAMY"
## [16] "MENACING 1ST DEGREE (VICT PEAC"
```

```
## [17] "USURY,CRIMINAL"
## [18] "MENACING 1ST DEGREE (VICT NOT"
## [19] "TRESPASS 1,CRIMINAL"
## [20] "FORGERY-ILLEGAL POSSESSION,VEH"
## [21] "ESCAPE 2,1"
## [22] "SUPP. ACT TERR 2ND"
## [23] "HIND PROSEC. TERR 2"
## [24] "TERRORISM PROVIDE SUPPORT"
## [25] "BAIL JUMPING 1 & 2"
## [26] "CONSPIRACY 2, 1"
## [27] "EXPOSURE OF A PERSON"
## [28] "USE OF A CHILD IN A SEXUAL PER"
## [29] "BRIBERY, POLICE OFFICER"
## [30] "VEHICULAR ASSAULT (INTOX DRIVE"
## [31] "TAMPERING WITH A WITNESS"
## [32] "FIREWORKS, POSSESS/USE"
## [33] "SOLICITATION 3,2,1, CRIMINAL"
## [34] "FIREWORKS PREV CONV 5 YEARS"
## [35] "PROMOTING A SEXUAL PERFORMANCE"
## [36] "MANUFACTURE UNAUTHORIZED RECOR"
## [37] "RIOT 1"
## [38] "CONSPIRACY 4, 3"
## [39] "APPEARANCE TICKET FAIL TO RESP"
## [40] "END WELFARE VULNERABLE ELDERLY PERSON"
## [41] "EAVESDROPPING"
## [42] "PERJURY 2,1,ETC"
```

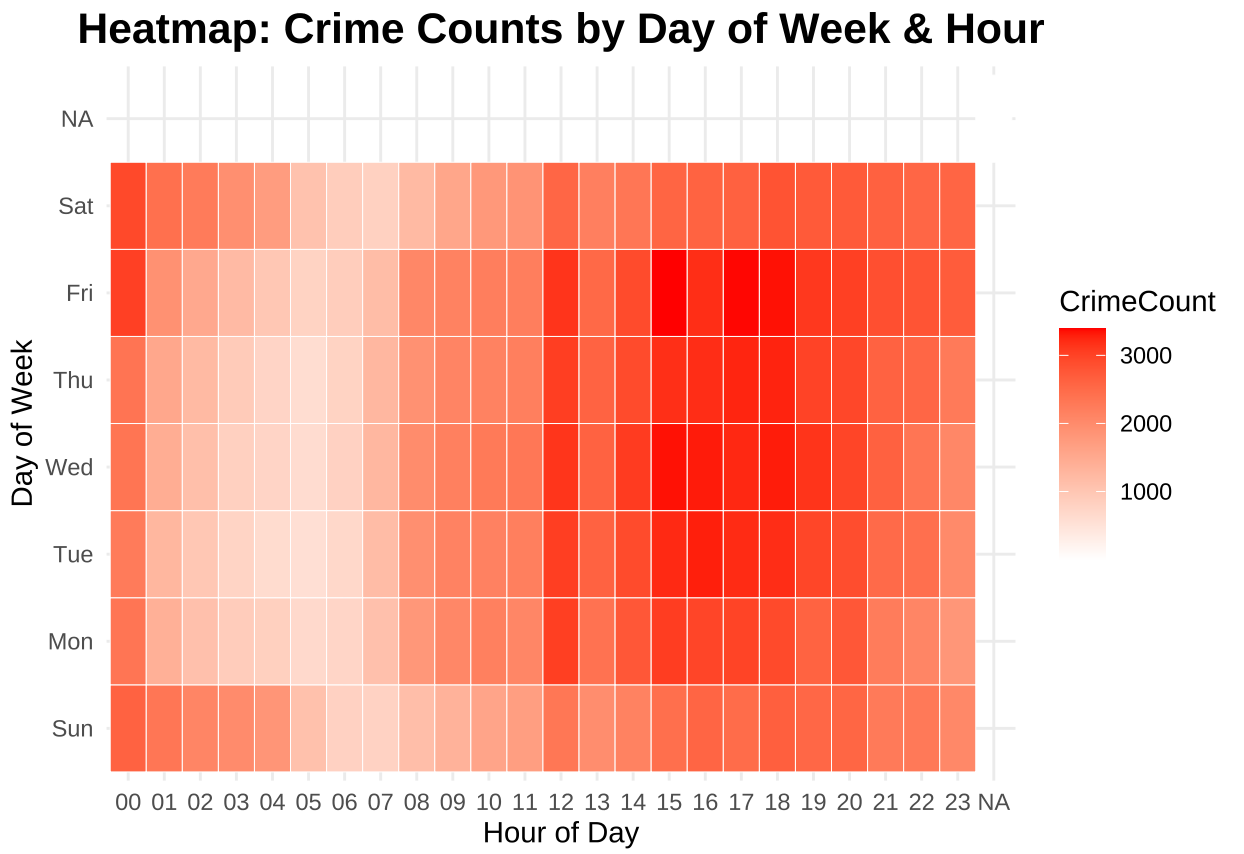
```
nycCrimeData$start <- as.POSIXlt(strptime(paste(nycCrimeData$CMPLNT_FR_DT,nycCrimeData$CMPLNT_FR_TM), "%
nycCrimeData$end <- as.POSIXlt(strptime(paste(nycCrimeData$CMPLNT_TO_DT,nycCrimeData$CMPLNT_TO_TM), "%m/
nycCrimeData$shr <-  as.factor(substr(nycCrimeData$start, 12, 13))
nycCrimeData$ehr <- as.factor(substr(nycCrimeData$end, 12, 13))
nycCrimeData$sdom <- as.factor(substr(nycCrimeData$start, 9, 10))
nycCrimeData$edom <- as.factor(substr(nycCrimeData$end, 9, 10))
nycCrimeData$mon <- as.factor(substr(nycCrimeData$start, 6, 7))
nycCrimeData$dow <- nycCrimeData$start$wday
nycCrimeData <- nycCrimeData[,c(-25,-26)]
#head(nycCrimeData)
```

```
#-----------------------------------------------------------------
# Convert Date and Time Columns into POSIXct Objects
#-----------------------------------------------------------------
nycCrimeData$start <- as.POSIXct(
  strptime(paste(nycCrimeData$CMPLNT_FR_DT, nycCrimeData$CMPLNT_FR_TM),
           "%m/%d/%Y %H:%M:%S")
)
# Extract the date and hour
nycCrimeData$day <- as.Date(nycCrimeData$start)
nycCrimeData$Hour <- as.factor(format(nycCrimeData$start, "%H"))
# Get day of week (0 = Sunday, ... 6 = Saturday)
nycCrimeData$dow <- as.POSIXlt(nycCrimeData$start)$wday
nycCrimeData$DayOfWeek <- factor(nycCrimeData$dow, levels = 0:6,
                                 labels = c("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"))
```

```r
#-------------------------------------------------------------
# Heatmap of Crime Counts by Day of Week and Hour
#-------------------------------------------------------------
heat_data <- nycCrimeData %>%
  group_by(DayOfWeek, Hour) %>%
  dplyr::summarise(CrimeCount = n())
```

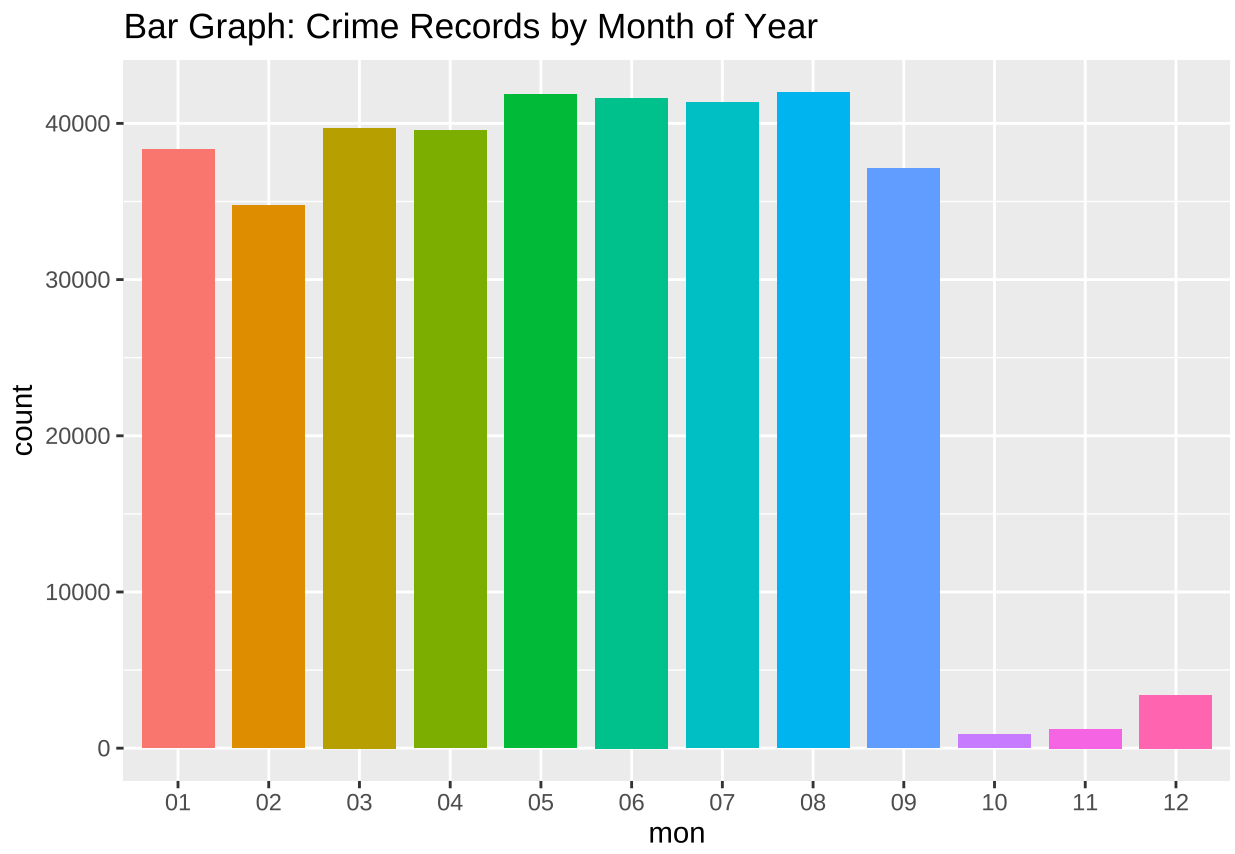
```
## `summarise()` has grouped output by 'DayOfWeek'. You can override using the
## `.groups` argument.
```

```r
p_heat <- ggplot(heat_data, aes(x = Hour, y = DayOfWeek, fill = CrimeCount)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "white", high = "red") +
  labs(title = "Heatmap: Crime Counts by Day of Week & Hour", x = "Hour of Day", y = "Day of Week") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, size = 16, face = "bold"))
print(p_heat)
```




```r
# Create a bar plot showing the number of crime records for each month.
# This helps identify seasonal trends or monthly patterns in crime occurrences.
# Crime records starting in each month

mon.bp <- ggplot(nycCrimeData, aes(x = mon, fill=as.factor(mon))) +
```

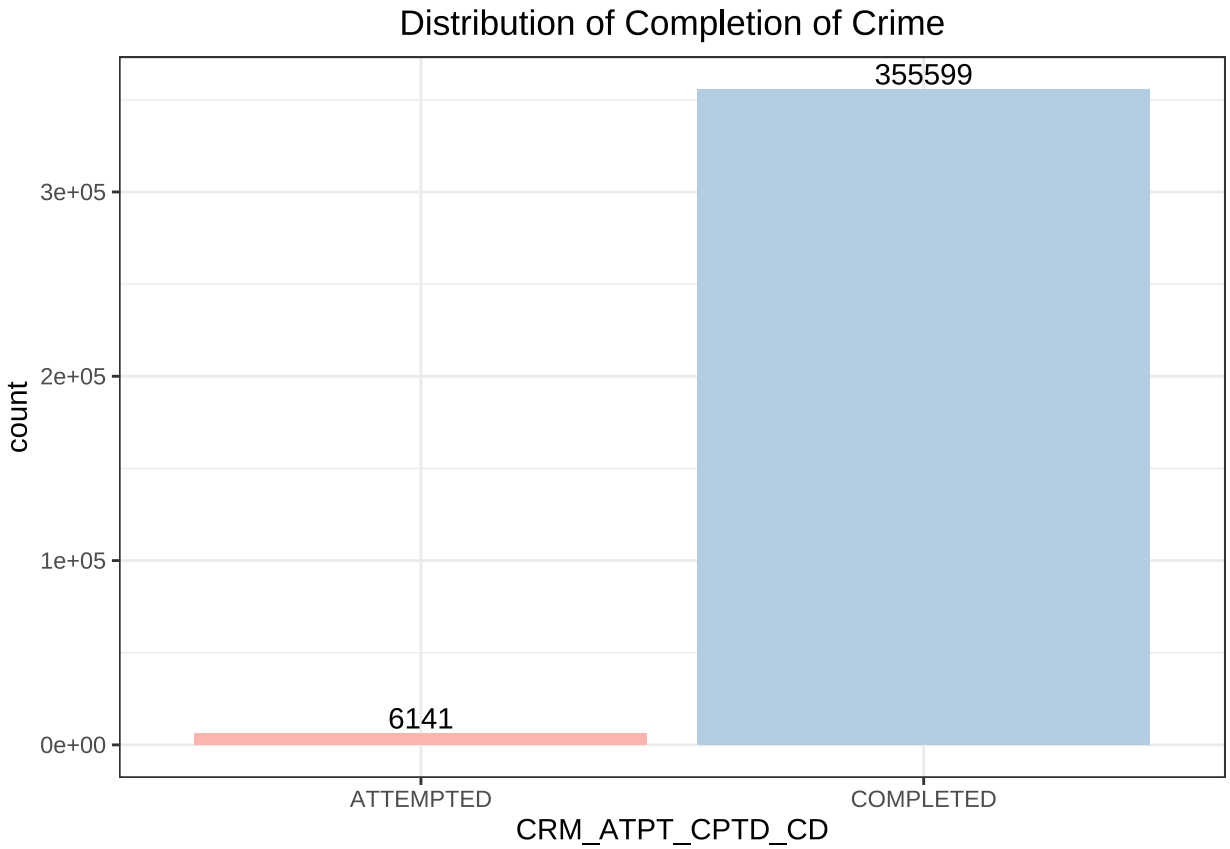
```
  geom_bar(width=0.8, stat="count") + theme(legend.position="none") +

  ggtitle("Bar Graph: Crime Records by Month of Year")

mon.bp
```




```
#count(nycCrimeData)
# Check the number of unique complaint records based on the complaint number.
# This helps ensure there are no duplicate crime reports.
n_distinct(nycCrimeData$CMPLNT_NUM)
```

```
## [1] 361740
```

```
# Create a bar plot to show the distribution of crimes based on whether they were
# completed or attempted. The bars are colored using a pastel palette, and counts
# are displayed above each bar for clarity.
ggplot(nycCrimeData, aes(x = CRM_ATPT_CPTD_CD,fill = CRM_ATPT_CPTD_CD )) + geom_bar(show.legend = FALSE
```




```
length(which(is.na(nycCrimeData$Latitude))) / length(nycCrimeData$Latitude)
```

```
## [1] 0.01618289
```

```
# Perform a Chi-squared test (with simulated p-value) to assess the association
# between boroughs and law categories of crimes.

c_test_2 <- chisq.test(table(nycCrimeData$BORO_NM, nycCrimeData$LAW_CAT_CD), simulate.p.value = TRUE)
c_test_2
```

```
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  table(nycCrimeData$BORO_NM, nycCrimeData$LAW_CAT_CD)
## X-squared = 1815.7, df = NA, p-value = 0.0004998
```

```
# Perform a Chi-squared test (with simulated p-value) to check for association
# between boroughs and the described locations where crimes occurred.
c_test_3 <- chisq.test(table(nycCrimeData$BORO_NM, nycCrimeData$LOC_OF_OCCUR_DESC), simulate.p.value = T
c_test_3
```

```
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
```

```
##  replicates)
##
## data:  table(nycCrimeData$BORO_NM, nycCrimeData$LOC_OF_OCCUR_DESC)
## X-squared = 3110.8, df = NA, p-value = 0.0004998
```

```
#####----------DATA CLEANING--------------------####
# Clean the dataset by handling missing values: removing unnecessary columns and rows with missing data

sum(is.na(nycCrimeData))
```

```
## [1] 148516
```

```
colSums(is.na(nycCrimeData))
```

```
##        CMPLNT_NUM      CMPLNT_FR_DT      CMPLNT_FR_TM      CMPLNT_TO_DT
##                 0                 0                 0                 0
##      CMPLNT_TO_TM            RPT_DT             KY_CD         OFNS_DESC
##                 0                 0                 0                 0
##             PD_CD           PD_DESC  CRM_ATPT_CPTD_CD        LAW_CAT_CD
##               263                 0                 0                 0
##        JURIS_DESC           BORO_NM       ADDR_PCT_CD LOC_OF_OCCUR_DESC
##                 0                 0                 1                 0
##     PREM_TYP_DESC          PARKS_NM        HADEVELOPT        X_COORD_CD
##                 0                 0                 0              5854
##        Y_COORD_CD          Latitude         Longitude           Lat_Lon
##              5854              5854              5854                 0
##               shr               ehr              sdom              edom
##                 0             62388                 0             62388
##               mon               dow             start               day
##                 0                12                12                12
##              Hour         DayOfWeek
##                12                12
```

```
#table(nycCrimeData$HADEVELOPT)

nyc_crime_drop = subset(nycCrimeData, select = -c(PARKS_NM,HADEVELOPT))



nyc_crime_clean = na.omit(nyc_crime_drop)
dim(nyc_crime_clean)
```

```
## [1] 294586     32
```

```
colSums(is.na(nyc_crime_clean))
```

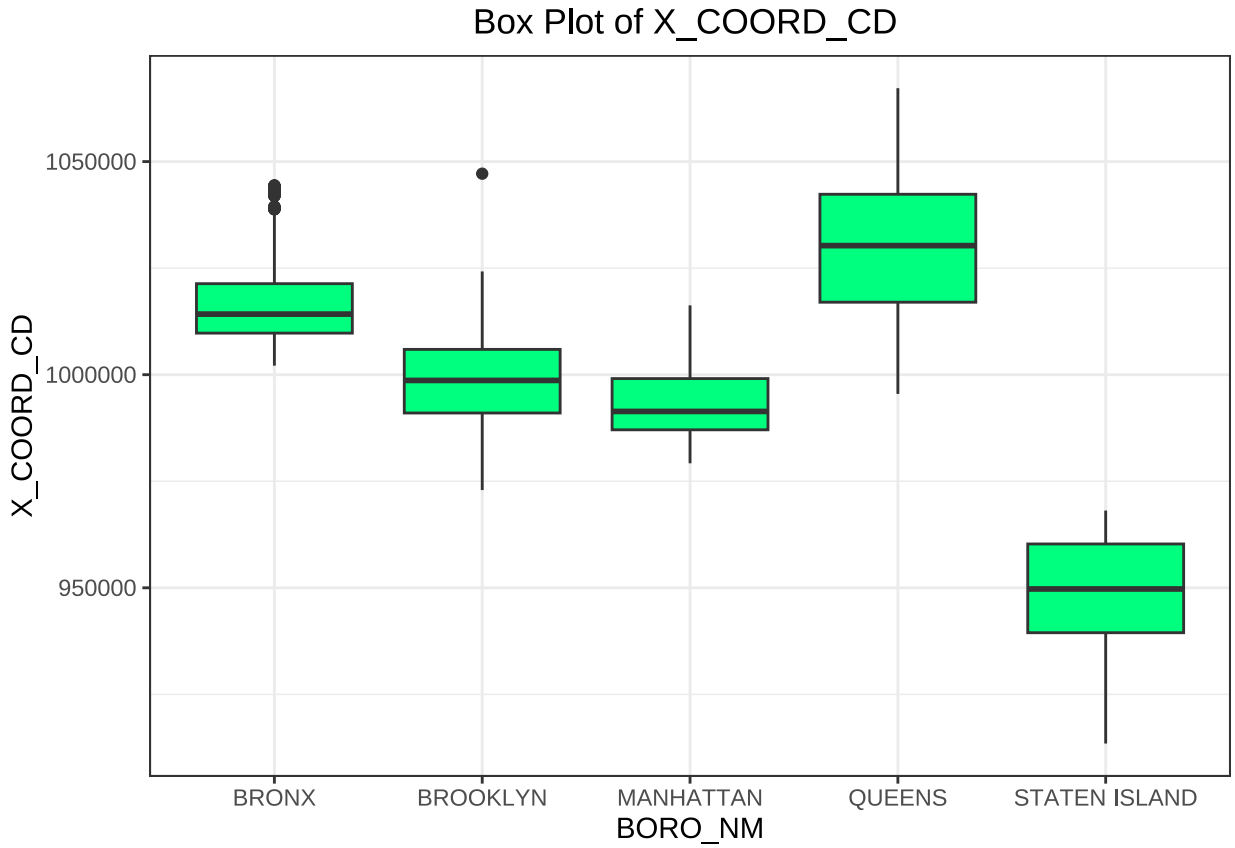
```
##        CMPLNT_NUM      CMPLNT_FR_DT      CMPLNT_FR_TM      CMPLNT_TO_DT
##                 0                 0                 0                 0
##      CMPLNT_TO_TM            RPT_DT             KY_CD         OFNS_DESC
##                 0                 0                 0                 0
##             PD_CD           PD_DESC  CRM_ATPT_CPTD_CD        LAW_CAT_CD
```

```
##                0                0                0                0
##        JURIS_DESC          BORO_NM     ADDR_PCT_CD LOC_OF_OCCUR_DESC
##                0                0                0                0
##     PREM_TYP_DESC      X_COORD_CD      Y_COORD_CD         Latitude
##                0                0                0                0
##         Longitude         Lat_Lon              shr              ehr
##                0                0                0                0
##              sdom             edom              mon              dow
##                0                0                0                0
##             start              day             Hour        DayOfWeek
##                0                0                0                0
```

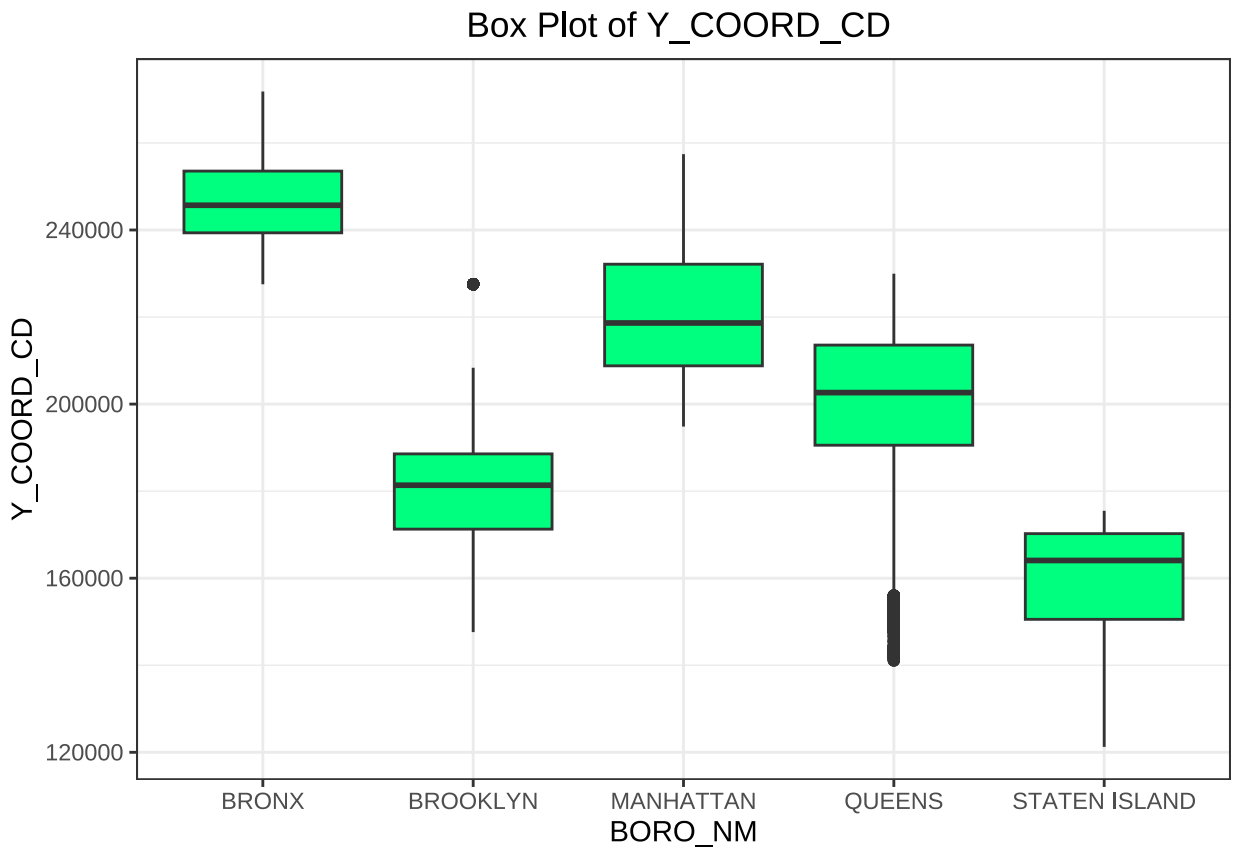
```
dim(nyc_crime_clean)
```

```
## [1] 294586     32
```

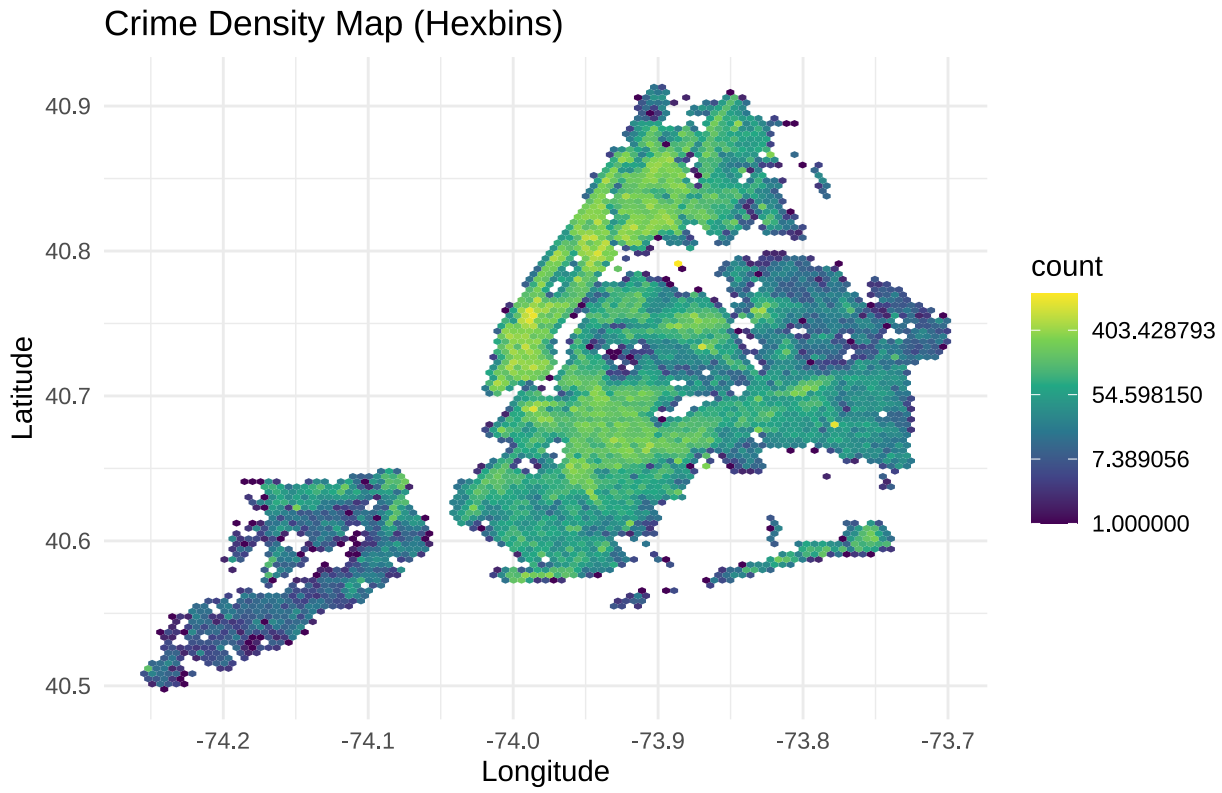
```
# Create a boxplot to visualize the distribution of X-coordinate values (longitude)
# across different NYC boroughs. This helps in understanding the east-west spread
# of crime incidents within each borough and detecting any spatial outliers.
ggplot(data = nyc_crime_clean, mapping = aes(x = BORO_NM, y = X_COORD_CD)) + geom_boxplot(fill="springg
```



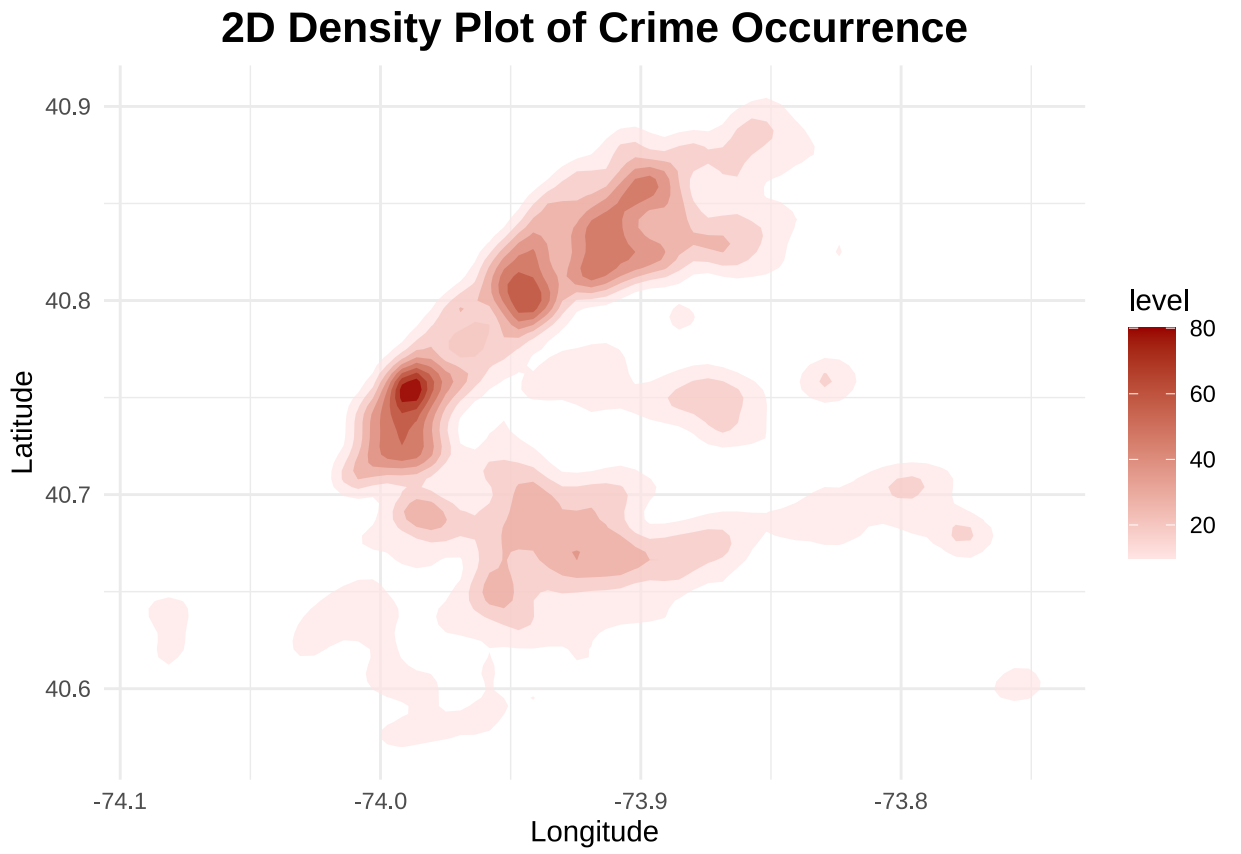

```
# Generate a boxplot to visualize the distribution of Y-coordinate values (latitude)
# across different NYC boroughs. This helps identify the spatial spread and outliers
# in the north-south direction within each borough.
ggplot(data = nyc_crime_clean, mapping = aes(x = BORO_NM, y = Y_COORD_CD)) + geom_boxplot(fill="springg
```




```r
# Create a hexbin density map of crime incidents in NYC,
# using longitude and latitude to show spatial distribution.
# Color intensity represents the (log-transformed) number of crimes in each hexbin.
ggplot(nyc_crime_clean, aes(x = Longitude, y = Latitude)) +
  stat_binhex(bins = 100) +
  scale_fill_viridis_c(trans = "log", option = "D") +
  coord_fixed() +
  labs(title = "Crime Density Map (Hexbins)", x = "Longitude", y = "Latitude") +
  theme_minimal()
```




```
#-----------------------------------------------------------------
# 2D Density Plot of Crime Occurrence with Red Shades
#-----------------------------------------------------------------
# Remove rows with missing Latitude or Longitude
nycCrimeData_clean <- nycCrimeData[!is.na(nycCrimeData$Latitude) & !is.na(nycCrimeData$Longitude), ]

p_density <- ggplot(nycCrimeData_clean, aes(x = Longitude, y = Latitude)) +
  stat_density2d(aes(fill = ..level..), geom = "polygon", contour = TRUE, alpha = 0.7) +
  scale_fill_gradient(low = "#FFE5E5", high = "#990000") +  # light red to dark red
  labs(title = "2D Density Plot of Crime Occurrence", x = "Longitude", y = "Latitude") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, size = 16, face = "bold"))

print(p_density)
```



PHASE 2

TRADITIONAL MACHINE LEARNING IN R

Data Preparation

```
#Loading the data-------------------------------------------------------------
nycCrimeData = read.csv("/Users/swati/Downloads/NYPD_Complaint_Data_Current_YTD.csv")
head(nycCrimeData)
```

```
##   CMPLNT_NUM CMPLNT_FR_DT CMPLNT_FR_TM CMPLNT_TO_DT CMPLNT_TO_TM     RPT_DT
## 1  736216184   09/30/2016     23:25:00   09/30/2016     23:25:00 09/30/2016
## 2  294332956   09/30/2016     23:16:00   09/30/2016     23:21:00 09/30/2016
## 3  852981427   09/30/2016     23:00:00   09/30/2016     23:05:00 09/30/2016
## 4  369976063   09/30/2016     23:00:00                          09/30/2016
## 5  117213771   09/30/2016     23:00:00   09/30/2016     23:10:00 09/30/2016
## 6  535504374   09/30/2016     22:55:00   09/30/2016     23:15:00 09/30/2016
##   KY_CD                    OFNS_DESC PD_CD                     PD_DESC
## 1   236            DANGEROUS WEAPONS   782    WEAPONS, POSSESSION, ETC
## 2   344 ASSAULT 3 & RELATED OFFENSES   101                     ASSAULT 3
```

```
## 3   235             DANGEROUS DRUGS   567 MARIJUANA, POSSESSION 4 & 5
## 4   118           DANGEROUS WEAPONS   793        WEAPONS POSSESSION 3
## 5   578              HARRASSMENT 2    637  HARASSMENT,SUBD 1,CIVILIAN
## 6   118           DANGEROUS WEAPONS   792    WEAPONS POSSESSION 1 & 2
##   CRM_ATPT_CPTD_CD  LAW_CAT_CD          JURIS_DESC  BORO_NM ADDR_PCT_CD
## 1        COMPLETED MISDEMEANOR N.Y. TRANSIT POLICE    BRONX          42
## 2        COMPLETED MISDEMEANOR    N.Y. POLICE DEPT BROOKLYN          71
## 3        COMPLETED MISDEMEANOR N.Y. HOUSING POLICE    BRONX          43
## 4        COMPLETED      FELONY    N.Y. POLICE DEPT   QUEENS         103
## 5        COMPLETED   VIOLATION    N.Y. POLICE DEPT   QUEENS         110
## 6        COMPLETED      FELONY    N.Y. POLICE DEPT BROOKLYN          75
##   LOC_OF_OCCUR_DESC              PREM_TYP_DESC PARKS_NM  HADEVELOPT X_COORD_CD
## 1                          TRANSIT - NYC SUBWAY                        1015308
## 2       OPPOSITE OF                     STREET                         997932
## 3            INSIDE RESIDENCE - PUBLIC HOUSING          CASTLE HILL    1025580
## 4                                       STREET                        1038464
## 5          FRONT OF                     STREET                        1016301
## 6                                       STREET                        1016123
##   Y_COORD_CD Latitude Longitude                        Lat_Lon
## 1     244373 40.83738 -73.88776 (40.837376359, -73.887760929)
## 2     180172 40.66120 -73.95069 (40.661204871, -73.950686652)
## 3     236918 40.81687 -73.85068 (40.816872438, -73.850684927)
## 4     192970 40.69618 -73.80449 (40.696177006, -73.804492266)
## 5     209428 40.74146 -73.88434 (40.741458245, -73.884339073)
## 6     180753 40.66275 -73.88512 (40.662752793, -73.885117129)
```

Data Cleaning

```
# Cleaning the dataset by handling missing values: removing unnecessary columns and rows with missing d

#table(nycCrimeData$PARKS_NM)

#table(nycCrimeData$HADEVELOPT)

colSums(is.na(nycCrimeData))
```

```
##        CMPLNT_NUM      CMPLNT_FR_DT      CMPLNT_FR_TM      CMPLNT_TO_DT
##                 0                 0                 0                 0
##      CMPLNT_TO_TM           RPT_DT             KY_CD         OFNS_DESC
##                 0                 0                 0                 0
##             PD_CD           PD_DESC  CRM_ATPT_CPTD_CD        LAW_CAT_CD
##               263                 0                 0                 0
##        JURIS_DESC           BORO_NM       ADDR_PCT_CD LOC_OF_OCCUR_DESC
##                 0                 0                 1                 0
##     PREM_TYP_DESC          PARKS_NM        HADEVELOPT        X_COORD_CD
##                 0                 0                 0              5854
##        Y_COORD_CD          Latitude         Longitude           Lat_Lon
##              5854              5854              5854                 0
```

```
nyc_crime_drop = subset(nycCrimeData, select = -c(PARKS_NM,HADEVELOPT,Latitude,Longitude,Lat_Lon,ADDR_P
```

```
nyc_crime_clean = na.omit(nyc_crime_drop)


colSums(is.na(nyc_crime_clean))
```

```
##        CMPLNT_NUM      CMPLNT_FR_DT      CMPLNT_FR_TM      CMPLNT_TO_DT
##                 0                 0                 0                 0
##      CMPLNT_TO_TM           RPT_DT             KY_CD         OFNS_DESC
##                 0                 0                 0                 0
##             PD_CD           PD_DESC  CRM_ATPT_CPTD_CD        LAW_CAT_CD
##                 0                 0                 0                 0
##        JURIS_DESC           BORO_NM LOC_OF_OCCUR_DESC     PREM_TYP_DESC
##                 0                 0                 0                 0
##        X_COORD_CD        Y_COORD_CD
##                 0                 0
```

```
dim(nyc_crime_clean)
```

```
## [1] 355625     18
```

```
crime_type <- c("LARCENY","ASSAULT","HARASSMENT,SUBD","THEFT","ADMINISTRATIVE CODE","HOMICIDE","INTOXICA
crime_type
```

```
##  [1] "LARCENY"             "ASSAULT"             "HARASSMENT,SUBD"
##  [4] "THEFT"               "ADMINISTRATIVE CODE" "HOMICIDE"
##  [7] "INTOXICATED"         "LOITERING"           "OTHERSTATE LAW"
## [10] "OFFENSES"            "CRIMINAL MISCHIEF"
```

```
for(i in 1:length(crime_type)){
  nyc_crime_clean$PD_DESC[grep(crime_type[i],nyc_crime_clean$PD_DESC)] <- crime_type[i]
  }

nyc_crime_clean$PD_DESC[nyc_crime_clean$PD_DESC == "HARASSMENT,SUBD"] <- "HARASSMENT"

#table(nycCrimeData$PD_DESC)
```

Feature Engineering and Train-Test split

```
#Feature Engineering and Train-Test split
# Prepare and split the cleaned crime dataset into training and test sets,
# convert character variables to numeric, and generate summary statistics for precincts by borough.

set.seed(43)

str(nyc_crime_clean)
```

```
## 'data.frame':    355625 obs. of  18 variables:
##  $ CMPLNT_NUM       : int  736216184 294332956 852981427 369976063 117213771 535504374 457718282 169
```

```
##  $ CMPLNT_FR_DT     : chr  "09/30/2016" "09/30/2016" "09/30/2016" "09/30/2016" ...
##  $ CMPLNT_FR_TM     : chr  "23:25:00" "23:16:00" "23:00:00" "23:00:00" ...
##  $ CMPLNT_TO_DT     : chr  "09/30/2016" "09/30/2016" "09/30/2016" "" ...
##  $ CMPLNT_TO_TM     : chr  "23:25:00" "23:21:00" "23:05:00" "" ...
##  $ RPT_DT           : chr  "09/30/2016" "09/30/2016" "09/30/2016" "09/30/2016" ...
##  $ KY_CD            : int  236 344 235 118 578 118 236 235 117 117 ...
##  $ OFNS_DESC        : chr  "DANGEROUS WEAPONS" "ASSAULT 3 & RELATED OFFENSES" "DANGEROUS DRUGS" "DANG
##  $ PD_CD            : int  782 101 567 793 637 792 782 511 568 501 ...
##  $ PD_DESC          : chr  "WEAPONS, POSSESSION, ETC" "ASSAULT" "MARIJUANA, POSSESSION 4 & 5" "WEAPO
##  $ CRM_ATPT_CPTD_CD : chr  "COMPLETED" "COMPLETED" "COMPLETED" "COMPLETED" ...
##  $ LAW_CAT_CD       : chr  "MISDEMEANOR" "MISDEMEANOR" "MISDEMEANOR" "FELONY" ...
##  $ JURIS_DESC       : chr  "N.Y. TRANSIT POLICE" "N.Y. POLICE DEPT" "N.Y. HOUSING POLICE" "N.Y. POLIC
##  $ BORO_NM          : chr  "BRONX" "BROOKLYN" "BRONX" "QUEENS" ...
##  $ LOC_OF_OCCUR_DESC: chr  "" "OPPOSITE OF" "INSIDE" "" ...
##  $ PREM_TYP_DESC    : chr  "TRANSIT - NYC SUBWAY" "STREET" "RESIDENCE - PUBLIC HOUSING" "STREET" ...
##  $ X_COORD_CD       : int  1015308 997932 1025580 1038464 1016301 1016123 1021316 1001446 1000481 101
##  $ Y_COORD_CD       : int  244373 180172 236918 192970 209428 180753 253601 234215 243059 240437 ...
##  - attr(*, "na.action")= 'omit' Named int [1:6115] 38 84 166 207 343 376 479 542 563 573 ...
##   ..- attr(*, "names")= chr [1:6115] "38" "84" "166" "207" ...
```

```
nyc_crime_clean <- nyc_crime_clean %>% mutate_if(is.character, function(x) unclass(as.factor(x)))
dim(nyc_crime_clean)
```

```
## [1] 355625     18
```

```
index <- 1:ncol(nyc_crime_clean)
index
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
```

```
nyc_crime_clean[ , index] <- lapply(nyc_crime_clean[ , index], as.numeric)
str(nyc_crime_clean)
```

```
## 'data.frame':    355625 obs. of  18 variables:
##  $ CMPLNT_NUM       : num  7.36e+08 2.94e+08 8.53e+08 3.70e+08 1.17e+08 ...
##  $ CMPLNT_FR_DT     : num  1091 1091 1091 1091 1091 ...
##  $ CMPLNT_FR_TM     : num  1406 1397 1381 1381 1381 ...
##  $ CMPLNT_TO_DT     : num  629 629 629 1 629 629 1 629 1 629 ...
##  $ CMPLNT_TO_TM     : num  1407 1403 1387 1 1392 ...
##  $ RPT_DT           : num  274 274 274 274 274 274 274 274 274 274 ...
##  $ KY_CD            : num  236 344 235 118 578 118 236 235 117 117 ...
##  $ OFNS_DESC        : num  15 8 14 15 27 15 15 14 14 14 ...
##  $ PD_CD            : num  782 101 567 793 637 792 782 511 568 501 ...
##  $ PD_DESC          : num  249 17 131 248 103 247 249 54 130 58 ...
##  $ CRM_ATPT_CPTD_CD : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ LAW_CAT_CD       : num  2 2 2 1 3 1 2 2 1 1 ...
##  $ JURIS_DESC       : num  10 7 6 7 7 7 7 7 7 6 ...
##  $ BORO_NM          : num  1 2 1 4 4 2 1 3 3 1 ...
##  $ LOC_OF_OCCUR_DESC: num  1 4 3 1 2 1 2 1 2 2 ...
##  $ PREM_TYP_DESC    : num  67 60 52 60 60 60 60 60 60 52 ...
##  $ X_COORD_CD       : num  1015308 997932 1025580 1038464 1016301 ...
##  $ Y_COORD_CD       : num  244373 180172 236918 192970 209428 ...
```

```
##  - attr(*, "na.action")= 'omit' Named int [1:6115] 38 84 166 207 343 376 479 542 563 573 ...
##   ..- attr(*, "names")= chr [1:6115] "38" "84" "166" "207" ...
```

```
training_crime <- sample(1:nrow(nyc_crime_clean), nrow(nyc_crime_clean)*.70, replace = FALSE)
#training_crime

feature_names <- names(nyc_crime_clean)[1:18]
feature_names
```

```
##  [1] "CMPLNT_NUM"         "CMPLNT_FR_DT"       "CMPLNT_FR_TM"
##  [4] "CMPLNT_TO_DT"       "CMPLNT_TO_TM"       "RPT_DT"
##  [7] "KY_CD"              "OFNS_DESC"          "PD_CD"
## [10] "PD_DESC"            "CRM_ATPT_CPTD_CD"   "LAW_CAT_CD"
## [13] "JURIS_DESC"         "BORO_NM"            "LOC_OF_OCCUR_DESC"
## [16] "PREM_TYP_DESC"      "X_COORD_CD"         "Y_COORD_CD"
```

```
train_set <- nyc_crime_clean[training_crime, c(feature_names)]
#train_set
getwd()
```

```
## [1] "/Users/swati/Desktop/Project_nycCRIME"
```

```
dim(train_set)
```

```
## [1] 248937     18
```

```
#table(train_set$BORO_NM) %>% prop.table()

test_set <- nyc_crime_clean[-training_crime, c(feature_names)]
#test_set

dim(test_set)
```

```
## [1] 106688     18
```

```
#table(test_set$BORO_NM) %>% prop.table()



boro_ranges <- aggregate(ADDR_PCT_CD ~ BORO_NM, data = nycCrimeData,
                         FUN = function(x) c(min = min(x, na.rm = TRUE),
                                             max = max(x, na.rm = TRUE)))

# Convert the matrix output to a data frame
boro_ranges <- do.call(data.frame, boro_ranges)

# Rename columns for clarity
names(boro_ranges) <- c("Borough", "Min_Precinct", "Max_Precinct")

print(boro_ranges)
```

```
##         Borough Min_Precinct Max_Precinct
## 1         BRONX           40           52
## 2      BROOKLYN           60           94
## 3     MANHATTAN            1           34
## 4        QUEENS          100          115
## 5 STATEN ISLAND          120          123
```

Logistic Regression

```
# This code performs a multinomial logistic regression on a dataset, predicts the class labels for both
# evaluates the model using confusion matrices, plots multiclass ROC curves, and computes the AUC, vari

dim(test_set)
```

```
## [1] 106688     18
```

```
dim(train_set)
```

```
## [1] 248937     18
```

```
formula <- "BORO_NM ~ RPT_DT + KY_CD + PD_CD + CRM_ATPT_CPTD_CD + LAW_CAT_CD + X_COORD_CD + Y_COORD_CD"
"RPT_DT" %in% names(train_set)
```

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

```
multinom.model <- multinom(formula, data=train_set, MaxNWts =1000000)
```

```
## # weights:  45 (32 variable)
## initial  value 400648.645606
## iter  10 value 272864.212025
## iter  20 value 272676.966836
## iter  30 value 171572.703979
## iter  40 value 119447.254033
## iter  50 value 119390.467432
## iter  60 value 97956.291412
## iter  70 value 77606.392319
## iter  80 value 76626.641528
## iter  90 value 65180.142990
## iter 100 value 61921.566669
## final  value 61921.566669
## stopped after 100 iterations
```

```
#head(your_dataframe$RPT_DT)  # See if it exists

train.result <- predict(object=multinom.model, newdata=train_set,type="class")

#train.result

confusionMatrix(train.result, as.factor(train_set$BORO_NM))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2     3     4     5
##          1 49768    25  4020    68     0
##          2     0 70826   466  2736   338
##          3  4639   431 55574  4278    11
##          4   857  1881    35 42001     0
##          5     0     0     0     0 10983
##
## Overall Statistics
##
##                Accuracy : 0.9205
##                  95% CI : (0.9195, 0.9216)
##     No Information Rate : 0.2939
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8959
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.9006   0.9681   0.9248   0.8557  0.96920
## Specificity            0.9788   0.9799   0.9504   0.9861  1.00000
## Pos Pred Value         0.9237   0.9524   0.8559   0.9381  1.00000
## Neg Pred Value         0.9718   0.9866   0.9754   0.9653  0.99853
## Prevalence             0.2220   0.2939   0.2414   0.1972  0.04552
## Detection Rate         0.1999   0.2845   0.2232   0.1687  0.04412
## Detection Prevalence   0.2164   0.2987   0.2608   0.1799  0.04412
## Balanced Accuracy      0.9397   0.9740   0.9376   0.9209  0.98460
```

```
test.result <- predict(object=multinom.model, newdata=test_set, type="class")

confusionMatrix(test.result, as.factor(test_set$BORO_NM))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2     3     4     5
##          1 21477    10  1756    19     0
##          2     0 30158   198  1160   119
##          3  1929   214 23643  1788     1
##          4   343   798    21 18178     0
##          5     0     0     0     0  4876
##
## Overall Statistics
##
##                Accuracy : 0.9217
##                  95% CI : (0.92, 0.9233)
##     No Information Rate : 0.2923
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

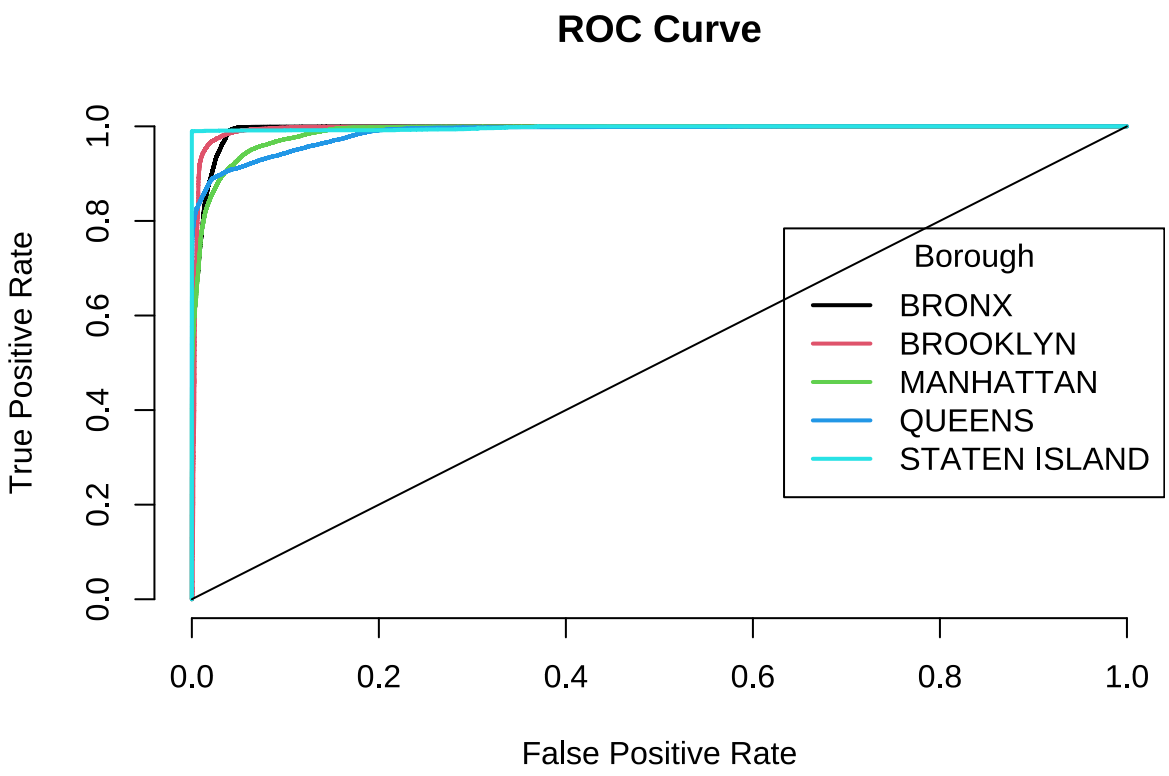
```
##                  Kappa : 0.8976
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.9043   0.9672   0.9229   0.8597  0.97598
## Specificity            0.9785   0.9804   0.9515   0.9864  1.00000
## Pos Pred Value         0.9233   0.9533   0.8574   0.9399  1.00000
## Neg Pred Value         0.9728   0.9864   0.9750   0.9660  0.99882
## Prevalence             0.2226   0.2923   0.2401   0.1982  0.04683
## Detection Rate         0.2013   0.2827   0.2216   0.1704  0.04570
## Detection Prevalence   0.2180   0.2965   0.2585   0.1813  0.04570
## Balanced Accuracy      0.9414   0.9738   0.9372   0.9230  0.98799
```

```
multiclass_roc_plot <- function(df, probs) {
     class.0.probs <- probs[,1]
     class.1.probs <- probs[,2]
     class.2.probs <- probs[,3]
     class.3.probs <- probs[,4]
     class.4.probs <- probs[,5]


     actual.0.class <- as.integer(df$BORO_NM == 1)
     actual.1.class <- as.integer(df$BORO_NM == 2)
     actual.2.class <- as.integer(df$BORO_NM == 3)
     actual.3.class <- as.integer(df$BORO_NM == 4)
     actual.4.class <- as.integer(df$BORO_NM == 5)



     plot(x=NA, y=NA, xlim=c(0,1), ylim=c(0,1),
          ylab='True Positive Rate',
          xlab='False Positive Rate',
          bty='n')

     legend(x = "right",
            title = "Borough",
            legend = c("BRONX", "BROOKLYN", "MANHATTAN", "QUEENS", "STATEN ISLAND"),
            col = c(1, 2, 3, 4 ,5),
            lwd = 2)

     title("ROC Curve")

     pred.0 = prediction(class.0.probs, actual.0.class)
     nbperf.0 = performance(pred.0, "tpr", "fpr")

     roc.x = unlist(nbperf.0@x.values)
     roc.y = unlist(nbperf.0@y.values)
     lines(roc.y ~ roc.x, col=0+1, lwd=2)


     pred.1 = prediction(class.1.probs, actual.1.class)
```

```
    nbperf.1 = performance(pred.1, "tpr", "fpr")

    roc.x = unlist(nbperf.1@x.values)
    roc.y = unlist(nbperf.1@y.values)
    lines(roc.y ~ roc.x, col=1+1, lwd=2)

    pred.2 = prediction(class.2.probs, actual.2.class)
    nbperf.2 = performance(pred.2, "tpr", "fpr")

    roc.x = unlist(nbperf.2@x.values)
    roc.y = unlist(nbperf.2@y.values)
    lines(roc.y ~ roc.x, col=2+1, lwd=2)

    pred.3 = prediction(class.3.probs, actual.3.class)
    nbperf.3 = performance(pred.3, "tpr", "fpr")

    roc.x = unlist(nbperf.3@x.values)
    roc.y = unlist(nbperf.3@y.values)
    lines(roc.y ~ roc.x, col=3+1, lwd=2)

    pred.4 = prediction(class.4.probs, actual.4.class)
    nbperf.4 = performance(pred.4, "tpr", "fpr")

    roc.x = unlist(nbperf.4@x.values)
    roc.y = unlist(nbperf.4@y.values)
    lines(roc.y ~ roc.x, col=4+1, lwd=2)



    lines(x=c(0,1), c(0,1))

 }

test.result.probs <- predict(multinom.model, test_set, type='probs')

roc.multi <- multiclass.roc(test_set$BORO_NM,test.result.probs)
auc(roc.multi)
```

```
## Multi-class area under the curve: 0.993
```

```
multiclass_roc_plot(test_set,test.result.probs)
```




```r
var(as.numeric(test.result), as.numeric(test_set$BORO_NM))
```

```
## [1] 1.208989
```

```r
bias(as.numeric(test.result), as.numeric(test_set$BORO_NM))
```

```
## [1] -0.01430339
```

Nearest Neighbour

```r
# This code applies the k-Nearest Neighbors (KNN) algorithm with different values of k (3, 5, and 11) t
# evaluates model performance using confusion matrices, and calculates the variance and bias of the pre

train_knn <- train_set
test_knn <- test_set

str(train_knn)
```

```
## 'data.frame':    248937 obs. of  18 variables:
##  $ CMPLNT_NUM       : num  5.44e+08 2.33e+08 1.69e+08 9.50e+08 2.09e+08 ...
##  $ CMPLNT_FR_DT     : num  318 356 775 779 825 61 905 500 273 414 ...
##  $ CMPLNT_FR_TM     : num  811 961 211 16 1231 ...
```

```
##  $ CMPLNT_TO_DT     : num  173 197 1 440 468 32 517 280 148 228 ...
##  $ CMPLNT_TO_TM     : num  827 1022 1 602 1242 ...
##  $ RPT_DT           : num  82 93 199 201 212 12 231 128 69 108 ...
##  $ KY_CD            : num  361 341 348 109 344 344 578 235 351 344 ...
##  $ OFNS_DESC        : num  39 49 59 25 8 8 27 14 12 8 ...
##  $ PD_CD            : num  661 343 916 421 101 101 637 511 254 101 ...
##  $ PD_DESC          : num  124 122 123 122 17 17 103 54 143 17 ...
##  $ CRM_ATPT_CPTD_CD : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ LAW_CAT_CD       : num  2 2 2 1 2 2 3 2 2 2 ...
##  $ JURIS_DESC       : num  7 7 7 7 7 7 7 7 7 7 ...
##  $ BORO_NM          : num  4 3 1 1 2 1 4 2 2 1 ...
##  $ LOC_OF_OCCUR_DESC: num  1 2 2 1 3 3 3 1 2 2 ...
##  $ PREM_TYP_DESC    : num  45 42 60 60 27 51 35 60 60 29 ...
##  $ X_COORD_CD       : num  1061282 996002 1025350 1009562 990563 ...
##  $ Y_COORD_CD       : num  210733 232214 241990 254560 187446 ...
```

```r
#train_knn

knn.model <- knn(train_knn, test_knn, cl = train_knn$BORO_NM, k = 3)

confusionMatrix(knn.model, as.factor(test_knn$BORO_NM))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2     3     4     5
##          1 19264    17  4851  1240     0
##          2    10 27253  2828  4482  2978
##          3  3928  2251 16750  2284   237
##          4   547  1491  1188 13139     1
##          5     0   168     1     0  1780
##
## Overall Statistics
##
##                Accuracy : 0.7328
##                  95% CI : (0.7302, 0.7355)
##     No Information Rate : 0.2923
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6465
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.8111   0.8741   0.6538   0.6214  0.35629
## Specificity            0.9264   0.8636   0.8927   0.9623  0.99834
## Pos Pred Value         0.7593   0.7258   0.6582   0.8028  0.91329
## Neg Pred Value         0.9448   0.9432   0.8908   0.9114  0.96930
## Prevalence             0.2226   0.2923   0.2401   0.1982  0.04683
## Detection Rate         0.1806   0.2554   0.1570   0.1232  0.01668
## Detection Prevalence   0.2378   0.3520   0.2385   0.1534  0.01827
## Balanced Accuracy      0.8688   0.8688   0.7733   0.7918  0.67731
```

```r
knn.model <- knn(train_knn, test_knn, cl = train_knn$BORO_NM, k = 5)

confusionMatrix(knn.model, as.factor(test_knn$BORO_NM))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2     3     4     5
##          1 18759    13  5129  1520     0
##          2     2 27152  3468  5311  3712
##          3  4362  2430 16068  2632   303
##          4   626  1499   952 11682     1
##          5     0    86     1     0   980
##
## Overall Statistics
##
##                Accuracy : 0.6996
##                  95% CI : (0.6969, 0.7024)
##     No Information Rate : 0.2923
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6008
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.7899   0.8708   0.6272   0.5525 0.196157
## Specificity            0.9197   0.8345   0.8800   0.9640 0.999144
## Pos Pred Value         0.7379   0.6849   0.6229   0.7915 0.918463
## Neg Pred Value         0.9386   0.9399   0.8819   0.8971 0.961977
## Prevalence             0.2226   0.2923   0.2401   0.1982 0.046828
## Detection Rate         0.1758   0.2545   0.1506   0.1095 0.009186
## Detection Prevalence   0.2383   0.3716   0.2418   0.1383 0.010001
## Balanced Accuracy      0.8548   0.8527   0.7536   0.7582 0.597651
```

```r
knn.model <- knn(train_knn, test_knn, cl = train_knn$BORO_NM, k = 11)

confusionMatrix(knn.model, as.factor(test_knn$BORO_NM))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2     3     4     5
##          1 16603    26  5399  1912     0
##          2    36 26455  4873  7102  4538
##          3  6220  2899 14684  3434   335
##          4   890  1782   662  8697     2
##          5     0    18     0     0   121
##
## Overall Statistics
##
```

```
##                Accuracy : 0.6239
##                  95% CI : (0.621, 0.6268)
##     No Information Rate : 0.2923
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.497
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.6991   0.8485   0.5732  0.41130 0.024219
## Specificity            0.9115   0.7808   0.8410  0.96100 0.999823
## Pos Pred Value         0.6935   0.6152   0.5326  0.72276 0.870504
## Neg Pred Value         0.9136   0.9258   0.8618  0.86849 0.954246
## Prevalence             0.2226   0.2923   0.2401  0.19819 0.046828
## Detection Rate         0.1556   0.2480   0.1376  0.08152 0.001134
## Detection Prevalence   0.2244   0.4031   0.2584  0.11279 0.001303
## Balanced Accuracy      0.8053   0.8146   0.7071  0.68615 0.512021
```

```
var(as.numeric(knn.model), as.numeric(test_knn$BORO_NM))
```

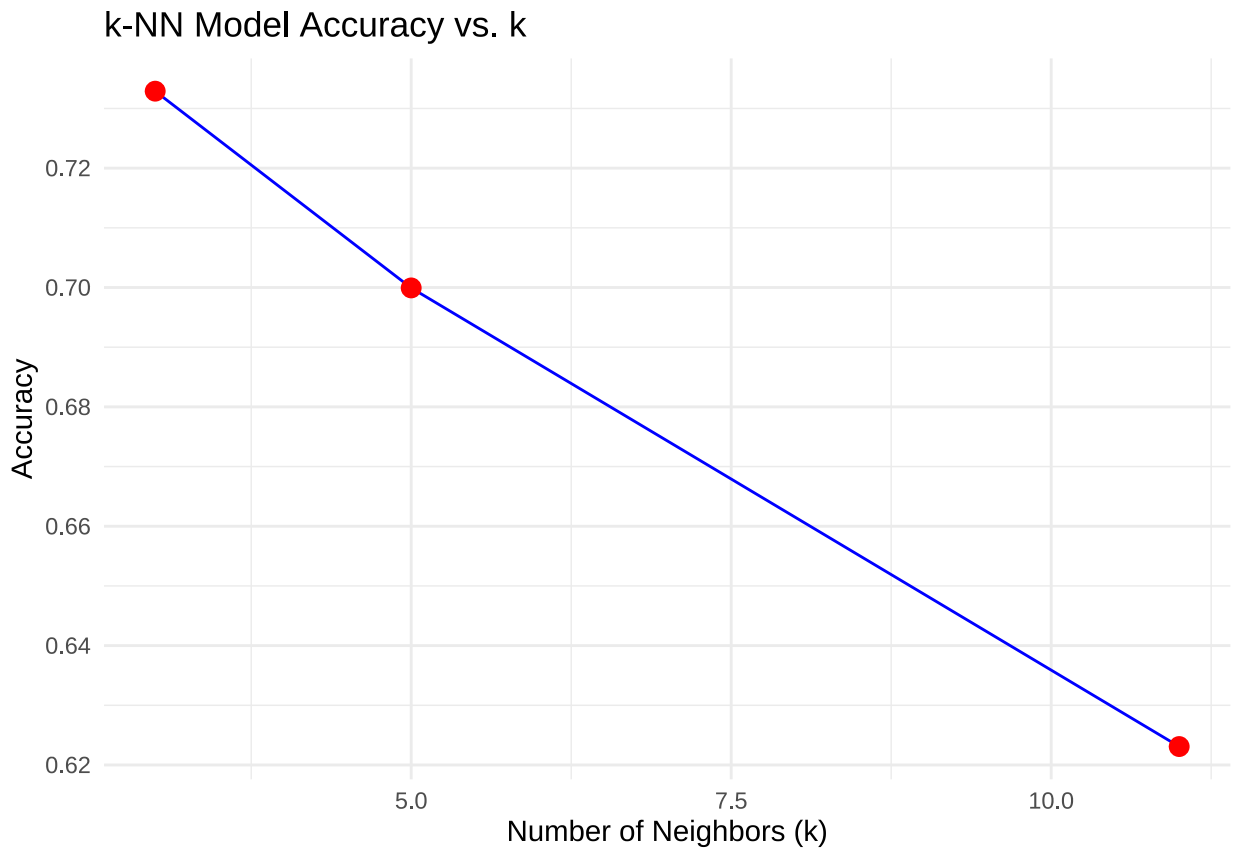
```
## [1] 0.4083277
```

```
bias(as.numeric(knn.model), as.numeric(test_knn$BORO_NM))
```

```
## [1] -0.2908668
```

```
library(class)
library(caret)
library(ggplot2)

train_knn <- train_set
test_knn <- test_set

k_values <- c(3, 5, 11)
accuracies <- numeric(length(k_values))

for (i in 1:length(k_values)) {
  k <- k_values[i]
  pred <- knn(train = train_knn[, -which(names(train_knn) == "BORO_NM")],
              test = test_knn[, -which(names(test_knn) == "BORO_NM")],
              cl = train_knn$BORO_NM, k = k)

  cm <- confusionMatrix(pred, as.factor(test_knn$BORO_NM))
  accuracies[i] <- cm$overall['Accuracy']
}

# Create a data frame for plotting
results_df <- data.frame(k = k_values, Accuracy = accuracies)
```

```r
# Plotting
ggplot(results_df, aes(x = k, y = Accuracy)) +
  geom_line(color = "blue") +
  geom_point(size = 3, color = "red") +
  labs(title = "k-NN Model Accuracy vs. k",
       x = "Number of Neighbors (k)",
       y = "Accuracy") +
  theme_minimal()
```



Decision Tree

```r
# This code builds a decision tree model using the rpart function, evaluates its performance on both th
# plots the decision tree, and calculates model metrics such as variance, bias, AUC, and multiclass ROC


dim(train_set)
```

```
## [1] 248937     18
```

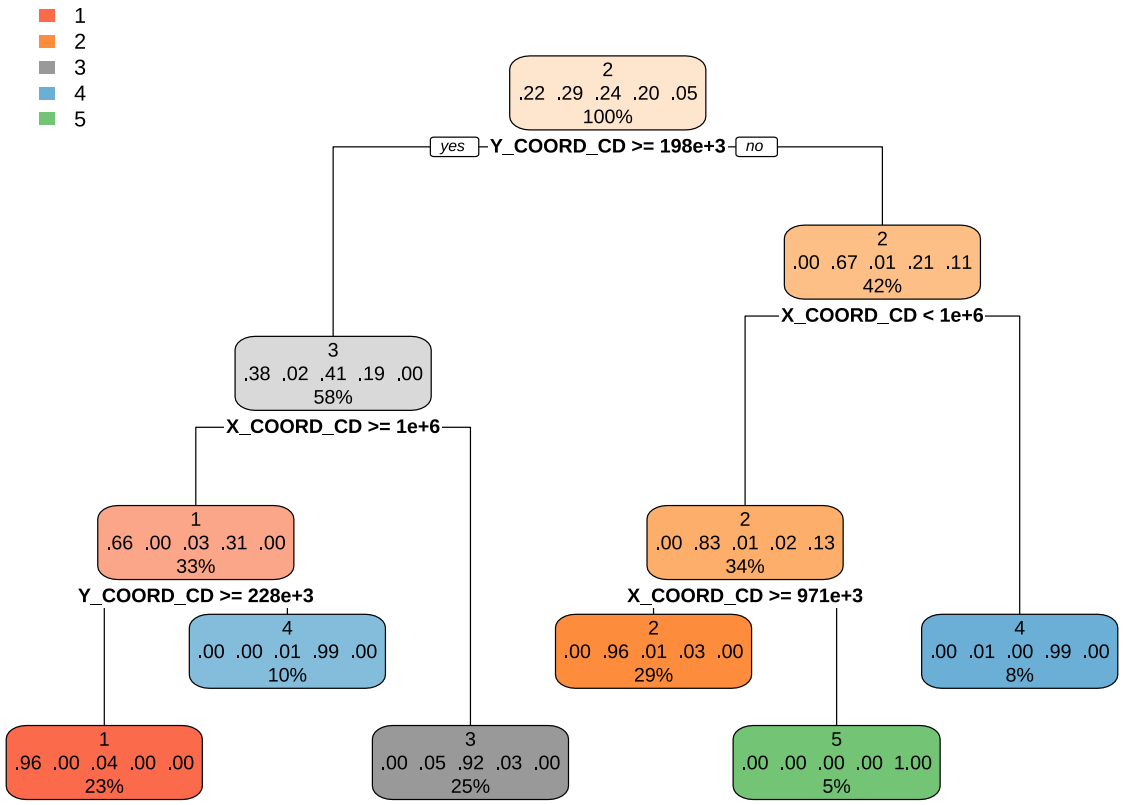
```r
dim(test_set)
```

```
## [1] 106688     18
```

```
# Check if target accidentally exists in predictors
dim(train_set)
```

```
## [1] 248937     18
```

```
#model_dt <- rpart(BORO_NM ~ ., data = train_set,method='class')

model_dt <- rpart(BORO_NM~., data=train_set, method = "class", parms=list(split=c("information","gini"))


rpart.plot(model_dt)
```




```
tr_predict_dt <- predict(object = model_dt, newdata = train_set, type="class")

#tr_predict_dt

confusionMatrix(tr_predict_dt, as.factor(train_set$BORO_NM))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2     3     4     5
##          1 55260    77  2203   134     0
##          2     0 69799   973  2025     0
```

```
##          3     4  3118 56768  1963     0
##          4     0   169   151 44961     0
##          5     0     0     0     0 11332
##
## Overall Statistics
##
##                Accuracy : 0.9565
##                  95% CI : (0.9557, 0.9573)
##     No Information Rate : 0.2939
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9432
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.9999   0.9540   0.9446   0.9160  1.00000
## Specificity            0.9875   0.9829   0.9731   0.9984  1.00000
## Pos Pred Value         0.9581   0.9588   0.9178   0.9929  1.00000
## Neg Pred Value         1.0000   0.9809   0.9822   0.9798  1.00000
## Prevalence             0.2220   0.2939   0.2414   0.1972  0.04552
## Detection Rate         0.2220   0.2804   0.2280   0.1806  0.04552
## Detection Prevalence   0.2317   0.2924   0.2485   0.1819  0.04552
## Balanced Accuracy      0.9937   0.9685   0.9589   0.9572  1.00000
```

```
tst_predict_dt <- predict(object = model_dt, newdata = test_set, type="class")

confusionMatrix(tst_predict_dt, as.factor(test_set$BORO_NM))
```

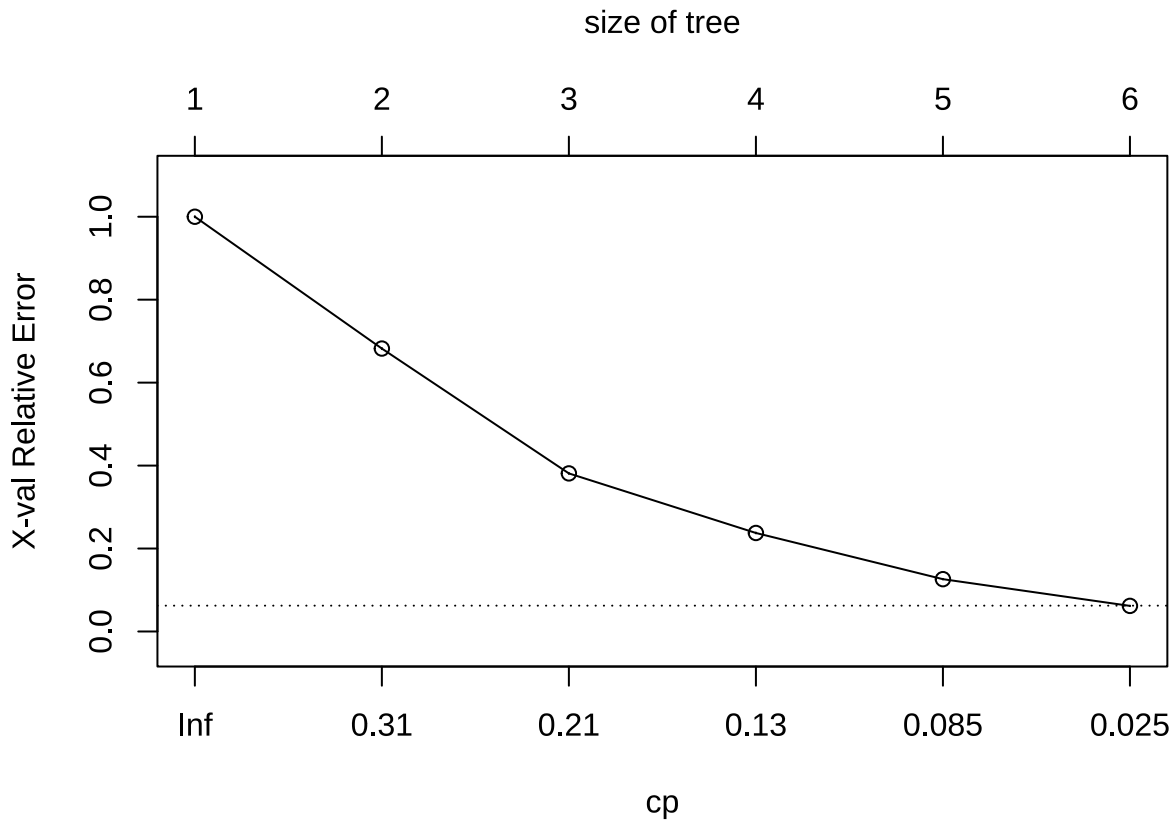
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2     3     4     5
##          1 23745    26   953    50     0
##          2     0 29724   389   898     0
##          3     4  1349 24198   807     0
##          4     0    81    78 19390     0
##          5     0     0     0     0  4996
##
## Overall Statistics
##
##                Accuracy : 0.9566
##                  95% CI : (0.9553, 0.9578)
##     No Information Rate : 0.2923
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9432
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.9998   0.9533   0.9446   0.9170  1.00000
## Specificity            0.9876   0.9830   0.9734   0.9981  1.00000
## Pos Pred Value         0.9585   0.9585   0.9181   0.9919  1.00000
## Neg Pred Value         1.0000   0.9808   0.9823   0.9799  1.00000
## Prevalence             0.2226   0.2923   0.2401   0.1982  0.04683
## Detection Rate         0.2226   0.2786   0.2268   0.1817  0.04683
## Detection Prevalence   0.2322   0.2907   0.2471   0.1832  0.04683
## Balanced Accuracy      0.9937   0.9681   0.9590   0.9576  1.00000
```

```
test.dtree.probs <- predict(model_dt,test_set, type="prob")

auc(roc.multi)
```

```
## Multi-class area under the curve: 0.993
```

```
plotcp(model_dt)
```



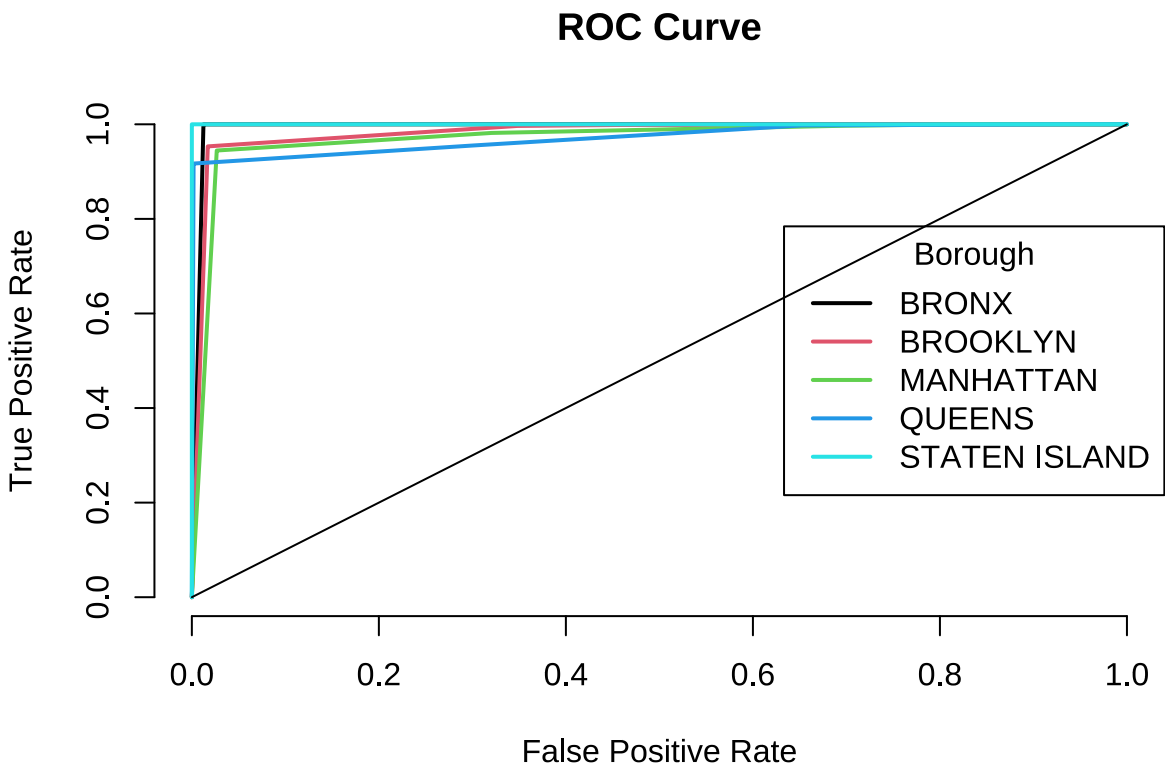

```
var(as.numeric(tst_predict_dt), unclass(as.factor(test_set$BORO_NM)))
```

```
## [1] 1.315221
```

```
bias(as.numeric(tst_predict_dt), as.vector(unclass(as.factor(test_set$BORO_NM)),'numeric'))
```

```
## [1] -0.03259036
```

```
multiclass_roc_plot(test_set, test.dtree.probs)
```



Support Vector Machines(SVM)

```
dim(train_set)
```

```
## [1] 248937     18
```

```
# Reducing the dataset to 62,000 rows (keeping all columns as it is taking more than 3 hours  for the l
reduced_train_set_trial1 <- train_set[1:62000, ]
dim(reduced_train_set_trial1)
```

```
## [1] 62000    18
```

```r
svmfit <- svm(BORO_NM~., data = reduced_train_set_trial1 , kernel = "radial", probability = TRUE)


pred.svm.train <- predict(svmfit, reduced_train_set_trial1 ,type="class")
#pred.svm.train
pred.svm.train <- round(pred.svm.train,digits=0)
#pred.svm.train
pred.svm.train[pred.svm.train == 0] <- 1
#pred.svm.train

common_levels_train <- levels(factor(reduced_train_set_trial1$BORO_NM))
predicted_train <- factor(pred.svm.train, levels = common_levels_train)
actual_train <- factor(reduced_train_set_trial1$BORO_NM, levels = common_levels_train)
# Confusion matrix
confusionMatrix(predicted_train, actual_train)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2     3     4     5
##          1  8096    22   146     0     0
##          2  5706 16365  2949  1647     0
##          3    69  1744 11639  4642   153
##          4     0     1   138  5958  1026
##          5     0     0     0    41  1560
##
## Overall Statistics
##
##                Accuracy : 0.7046
##                  95% CI : (0.701, 0.7082)
##     No Information Rate : 0.2929
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6057
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.5837   0.9025   0.7826  0.48486  0.56955
## Specificity            0.9965   0.7646   0.8595  0.97652  0.99931
## Pos Pred Value         0.9797   0.6137   0.6379  0.83645  0.97439
## Neg Pred Value         0.8923   0.9499   0.9259  0.88444  0.98045
## Prevalence             0.2241   0.2929   0.2403  0.19851  0.04425
## Detection Rate         0.1308   0.2644   0.1880  0.09625  0.02520
## Detection Prevalence   0.1335   0.4308   0.2948  0.11507  0.02586
## Balanced Accuracy      0.7901   0.8336   0.8211  0.73069  0.78443
```

```r
#confusionMatrix(as.factor(pred.svm.train), as.factor(reduced_train_set_trial1$BORO_NM))


#-------------------------------------------------------------------------------
```

```r
dim(test_set)
```

```
## [1] 106688     18
```

```r
reduced_test_set_trial1 <- test_set[1:18600, ]
dim(reduced_test_set_trial1)
```

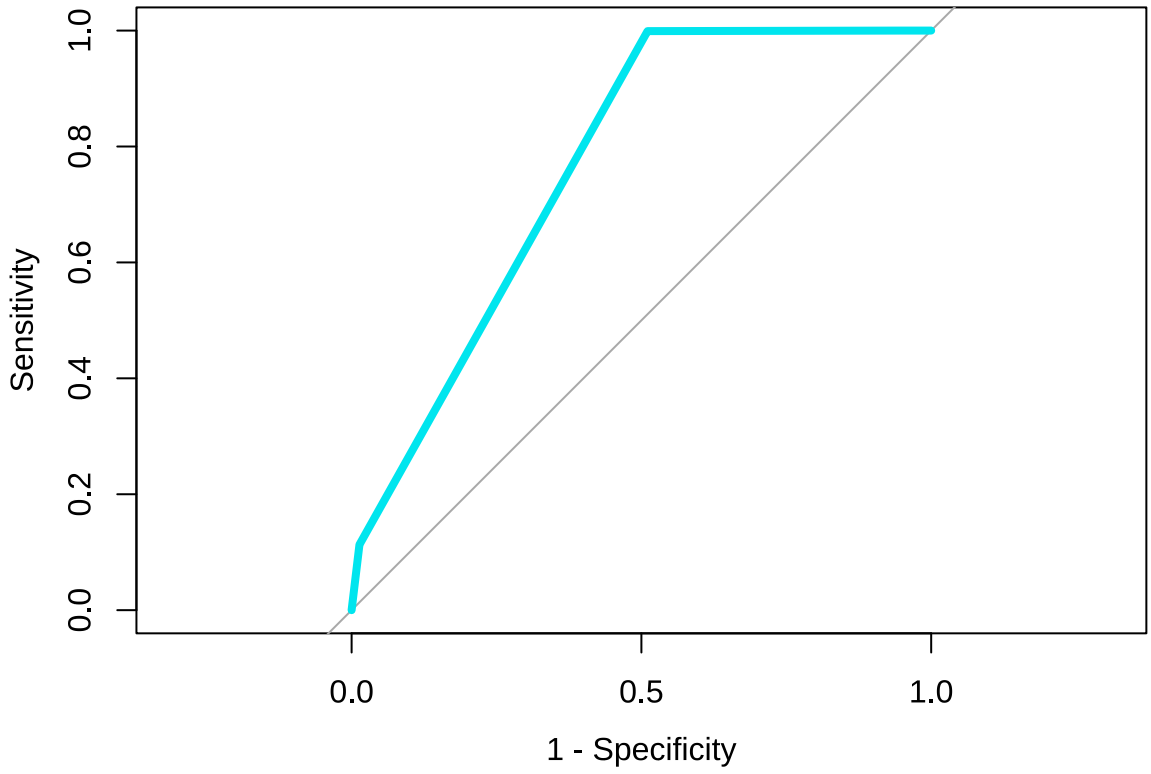
```
## [1] 18600    18
```

```r
pred.svm.test <- predict(svmfit, reduced_test_set_trial1)
pred.svm.test <- round(pred.svm.test, digits = 0)


common_levels_test <- levels(factor(reduced_test_set_trial1$BORO_NM))
predicted_test <- factor(pred.svm.test, levels = common_levels_test)
actual_test <- factor(reduced_test_set_trial1$BORO_NM, levels = common_levels_test)
confusionMatrix(predicted_test, actual_test)
```

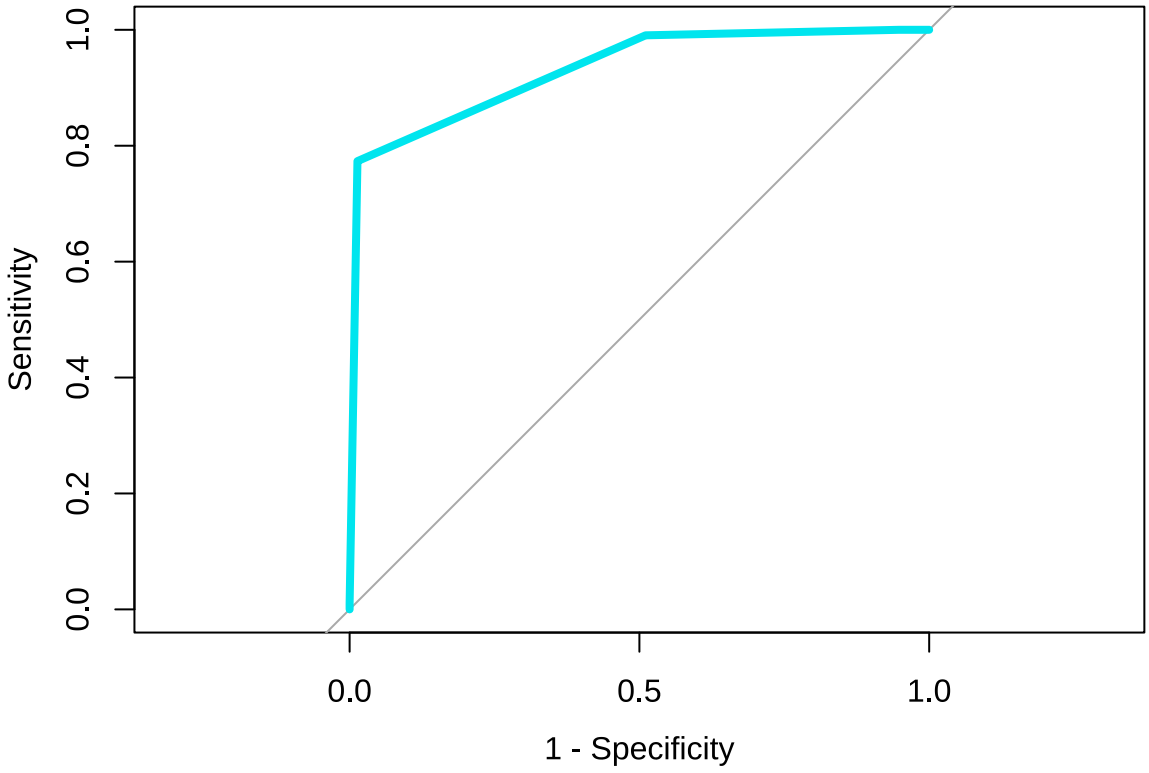
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2    3    4    5
##          1 1750    5   43    0    0
##          2 1990 4970  980  427    1
##          3   54  633 3449 1469   79
##          4    0    1   43 1697  303
##          5    0    0    0   33  432
##
## Overall Statistics
##
##                Accuracy : 0.6699
##                  95% CI : (0.663, 0.6767)
##     No Information Rate : 0.3055
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5546
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity           0.46125   0.8861   0.7639  0.46801  0.53006
## Specificity           0.99670   0.7335   0.8386  0.97645  0.99812
## Pos Pred Value        0.97330   0.5939   0.6068  0.83023  0.92903
## Neg Pred Value        0.87658   0.9360   0.9159  0.88177  0.97860
## Prevalence            0.20666   0.3055   0.2459  0.19751  0.04439
## Detection Rate        0.09532   0.2707   0.1879  0.09243  0.02353
## Detection Prevalence  0.09794   0.4558   0.3096  0.11134  0.02533
## Balanced Accuracy     0.72898   0.8098   0.8012  0.72223  0.76409
```

```r
test.svm.probs <- predict(svmfit,reduced_test_set_trial1, probability = TRUE)
common_levels_test <- levels(factor(reduced_test_set_trial1$BORO_NM))
predicted <- factor(test.svm.probs, levels = common_levels_test)
actual <- factor(reduced_test_set_trial1$BORO_NM, levels = common_levels_test)

roc <- multiclass.roc(reduced_test_set_trial1$BORO_NM, pred.svm.test, plot=TRUE,col="turquoise2",lwd = 4
```

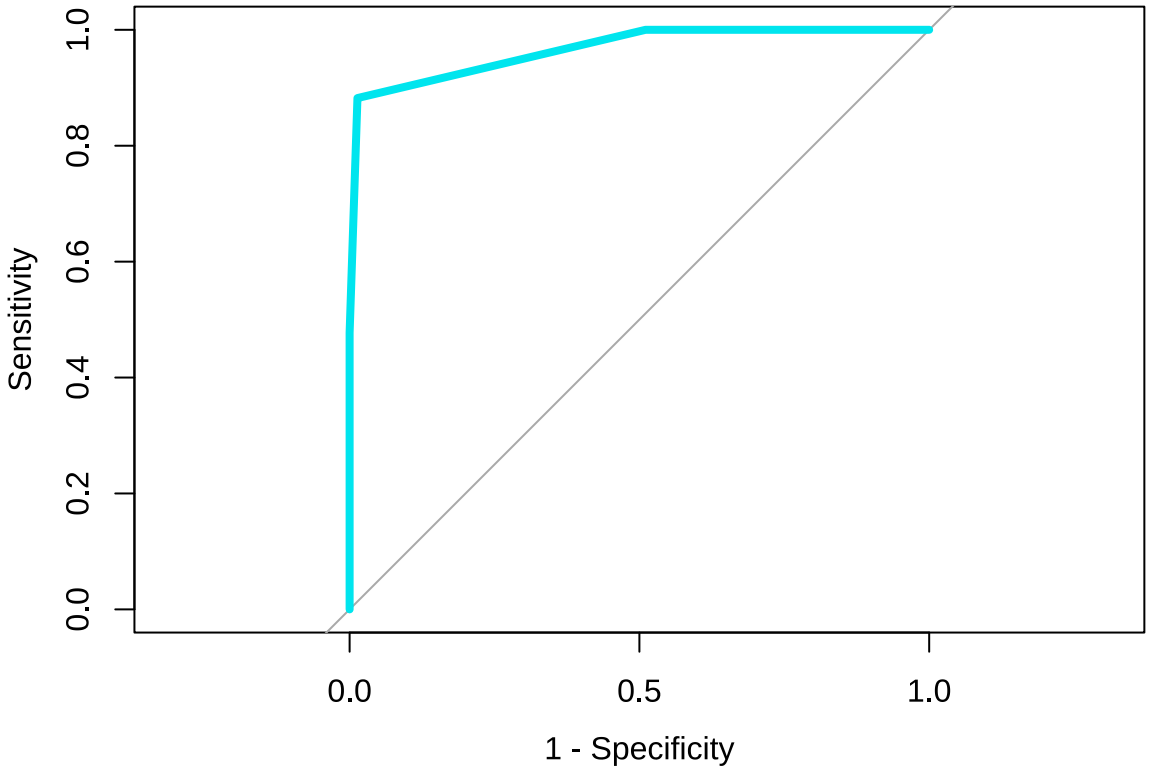
```
## Setting direction: controls < cases
```



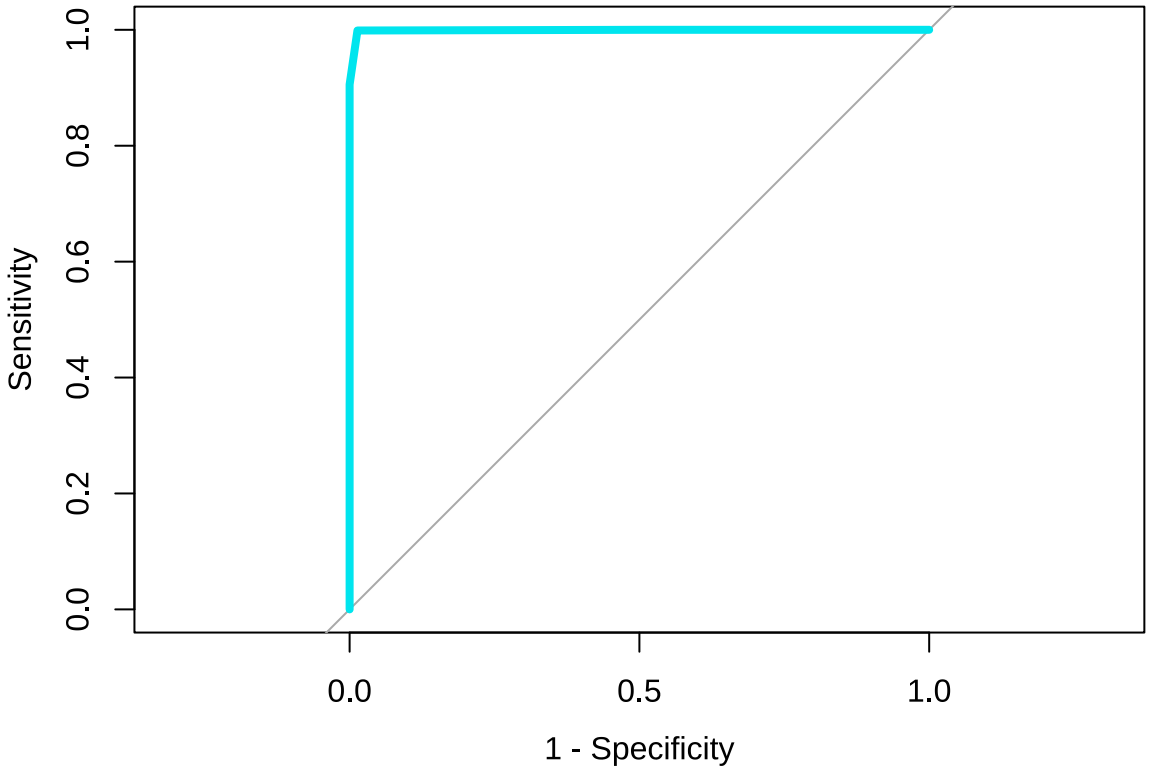

```
## Setting direction: controls < cases
```



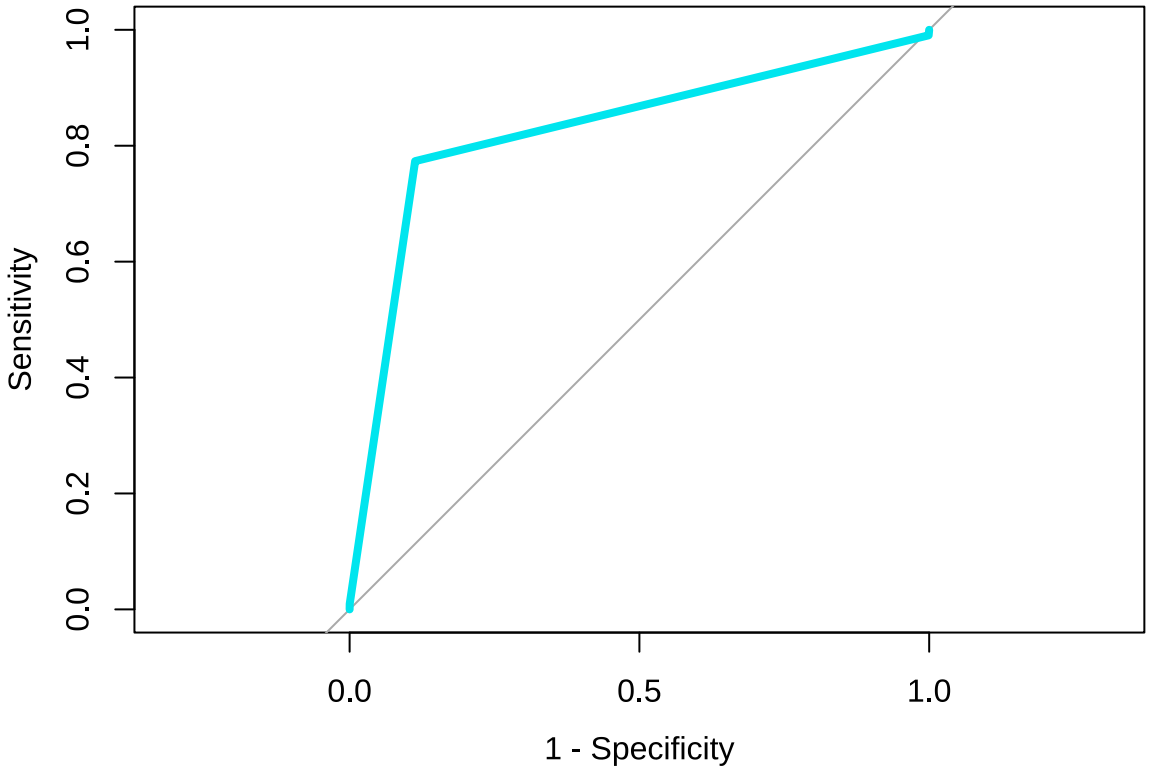

```
## Setting direction: controls < cases
```



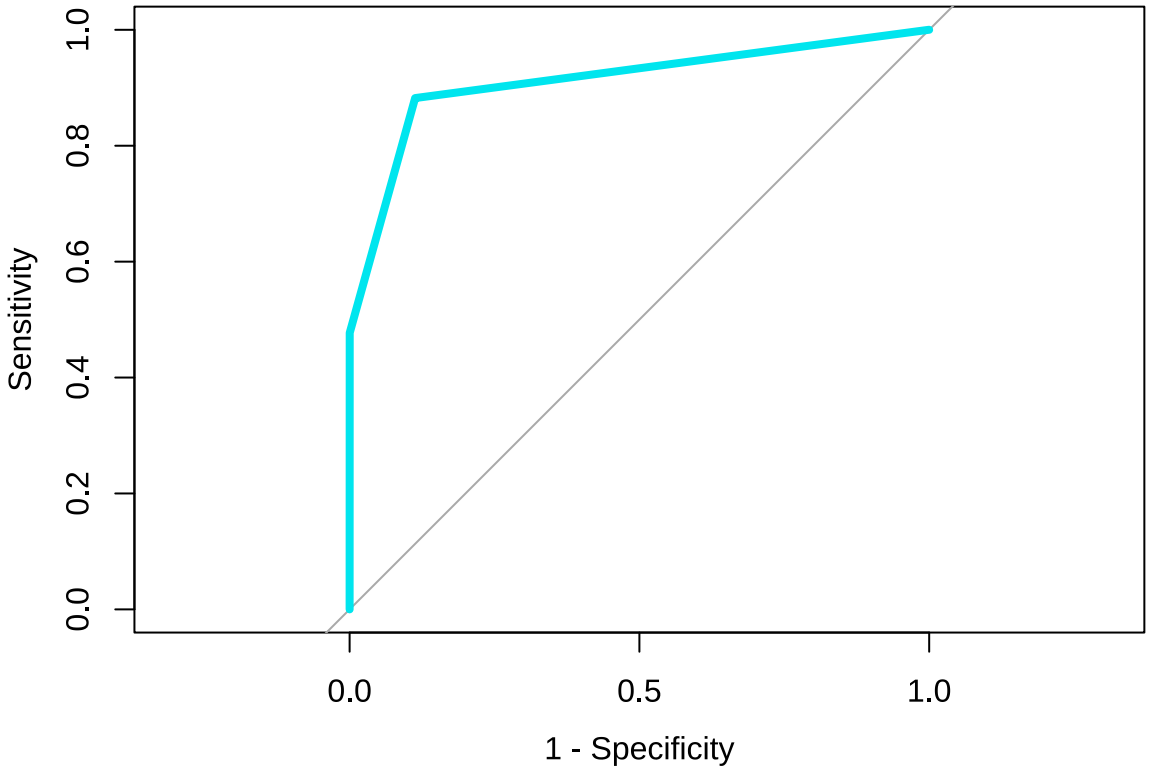

```
## Setting direction: controls < cases
```



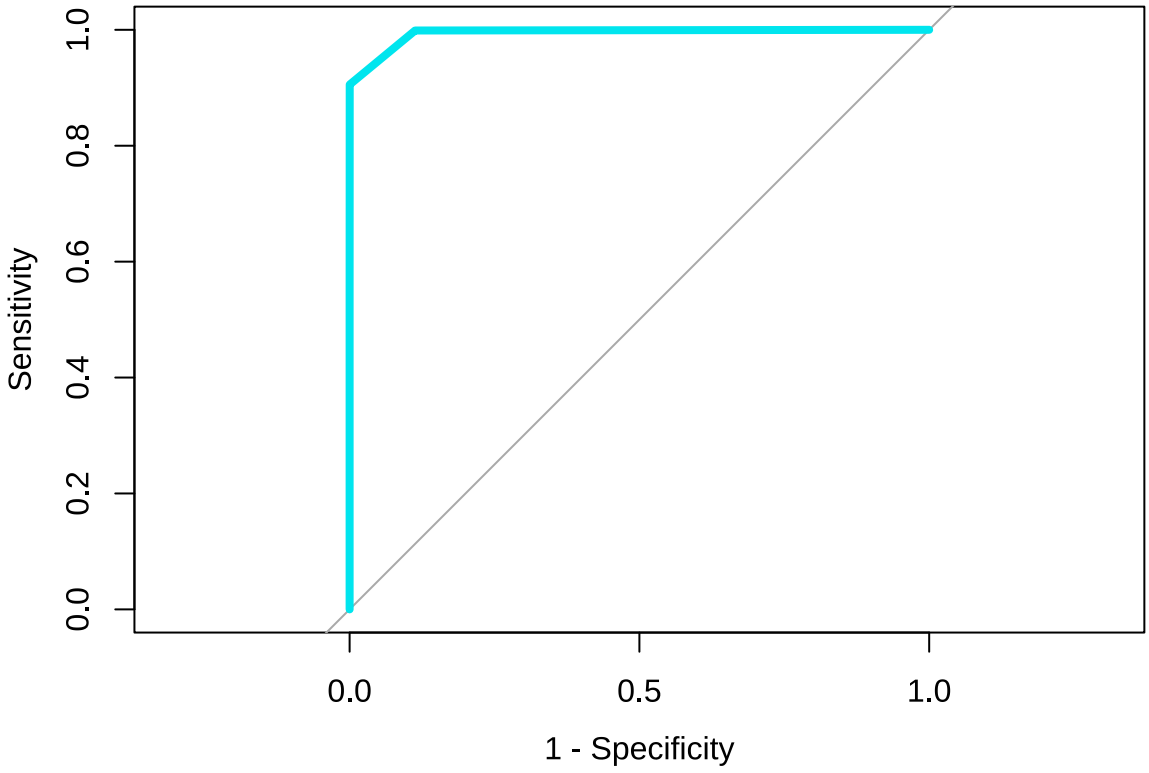

```
## Setting direction: controls < cases
```



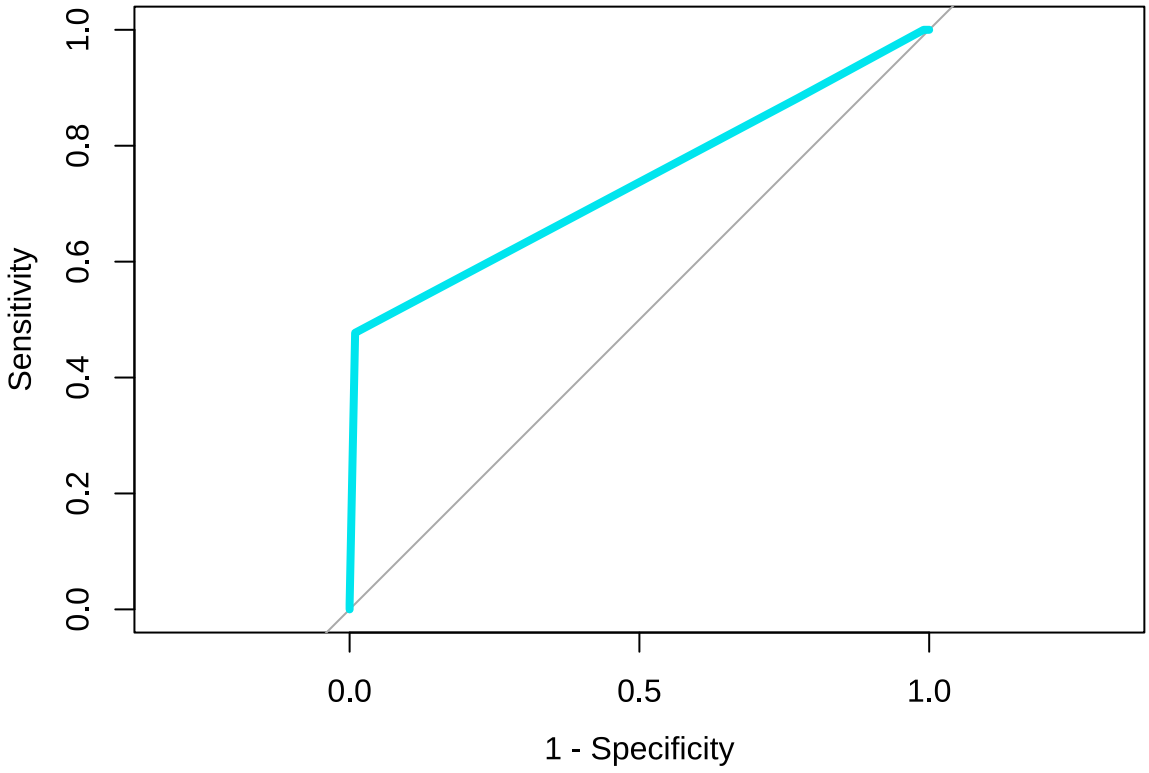

```
## Setting direction: controls < cases
```



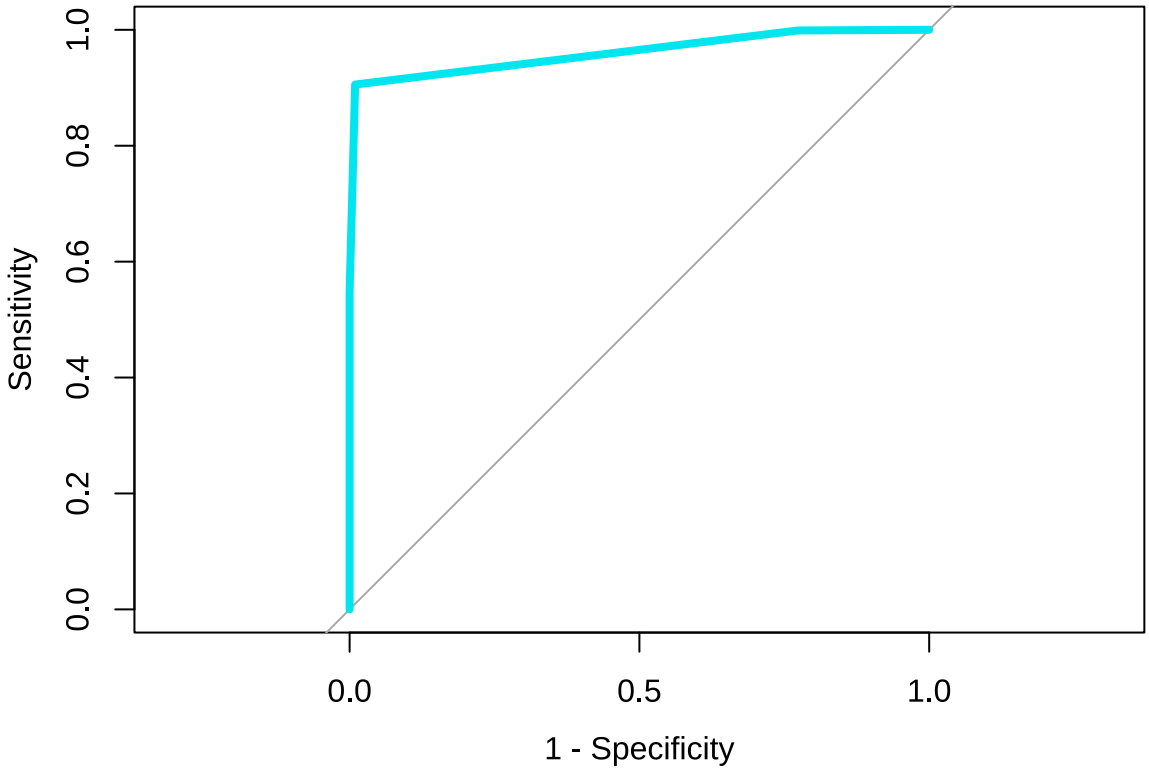

```
## Setting direction: controls < cases
```




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

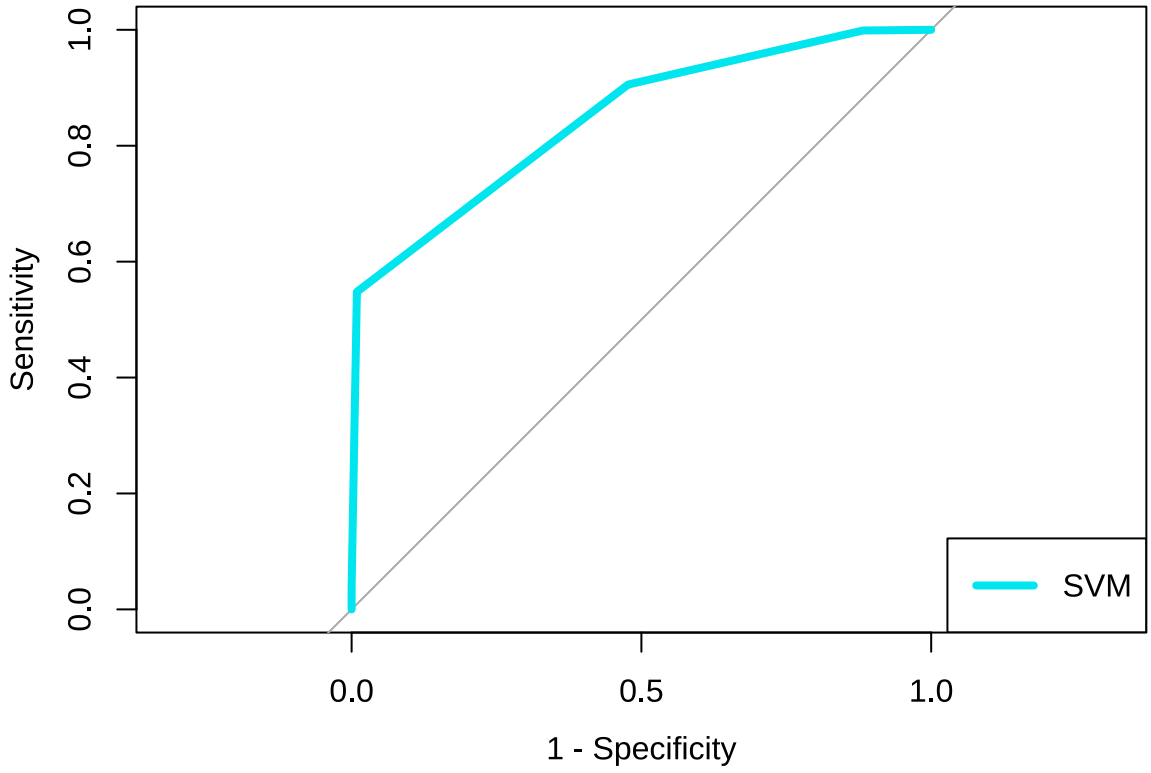



```
## Setting direction: controls < cases
```




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

```
legend("bottomright",legend=c("SVM"),col=c("turquoise2"),lwd=4)
```




```
roc.multi <- multiclass.roc(reduced_test_set_trial1$BORO_NM, test.svm.probs)
```

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

```
roc.multi
```

```
##
## Call:
## multiclass.roc.default(response = reduced_test_set_trial1$BORO_NM,     predictor = test.svm.probs)
##
## Data: test.svm.probs with 5 levels of reduced_test_set_trial1$BORO_NM: 1, 2, 3, 4, 5.
## Multi-class area under the curve: 0.9277
```

```
auc(roc.multi)
```

```
## Multi-class area under the curve: 0.9277
```

```r
var(as.numeric(pred.svm.test), as.numeric(reduced_test_set_trial1$BORO_NM))
```

```
## [1] 0.9048064
```

```r
bias(as.numeric(pred.svm.test), as.numeric(reduced_test_set_trial1$BORO_NM))
```

```
## [1] -0.06596774
```

Random Forest(RF)

```r
# This code trains a random forest classifier on the training dataset, evaluates feature importance, pr
# and assesses model performance using confusion matrices, variance, and bias calculations.

dim(train_set)
```

```
## [1] 248937     18
```

```r
dim(test_set)
```

```
## [1] 106688     18
```

```r
model_rf <- randomForest(as.factor(BORO_NM) ~ ., ntree = 3 , importance = TRUE, data = train_set, tuneG

importance(model_rf)
```

```
##                             1          2         3          4          5
## CMPLNT_NUM          -0.5881758   1.4880083 -4.1532787   0.5726817 -1.38221352
## CMPLNT_FR_DT         4.8400015   1.6390384  4.8013686   2.4050375  0.57699878
## CMPLNT_FR_TM         7.9208891   2.0977627  4.3175188   2.8148224 -0.00530051
## CMPLNT_TO_DT         7.9557051   1.4909255  5.7095708   4.2048506  1.56004207
## CMPLNT_TO_TM         3.1327646   3.1402199  2.4245789   4.2813847  0.62556668
## RPT_DT               5.2862539   2.2931355  5.4373164   2.4392325  0.41931550
## KY_CD                1.9442802   1.6273333  3.2364883   1.8667054  1.61547345
## OFNS_DESC            1.9809531   1.5505312  3.2377453   3.1095243  0.06675239
## PD_CD                2.7630630   2.0158978  3.2059972   1.9242063  0.37313778
## PD_DESC              3.6341616   2.5565776  1.9890914   2.2913417  1.38388850
## CRM_ATPT_CPTD_CD    -2.4493946   1.2247449  0.6982697   0.8853226  1.22474487
## LAW_CAT_CD           1.5430611   0.4618702  2.3176391   1.9341012 -0.67097149
## JURIS_DESC           6.0931487   5.2265535  6.8147242  12.2691942  2.92857485
## LOC_OF_OCCUR_DESC    2.2581347   1.9191803  3.8953156   1.8834540  1.42080074
## PREM_TYP_DESC        1.7373655   4.4068106  2.8041457   4.3630753  2.04515327
## X_COORD_CD        1388.0285472  36.3251253 58.5487127 138.4517881 37.08669442
## Y_COORD_CD          66.5964261 177.9073924 87.3068514  35.6363580 32.90517439
##                   MeanDecreaseAccuracy MeanDecreaseGini
## CMPLNT_NUM                    -1.470858        794.07161
## CMPLNT_FR_DT                   4.861002        716.36876
## CMPLNT_FR_TM                   5.490428        829.10441
## CMPLNT_TO_DT                   5.305452        599.10823
```

```
## CMPLNT_TO_TM                  3.189946       1247.57978
## RPT_DT                        4.539801        710.22018
## KY_CD                        24.900204        453.38935
## OFNS_DESC                     3.796987        478.91723
## PD_CD                         8.411519        613.80291
## PD_DESC                       2.247332        348.62566
## CRM_ATPT_CPTD_CD              1.306686         21.97099
## LAW_CAT_CD                    5.726252         80.31616
## JURIS_DESC                    8.430546       1631.90071
## LOC_OF_OCCUR_DESC             4.727024        279.64940
## PREM_TYP_DESC                12.316629       1381.30592
## X_COORD_CD                   84.103601      87501.87774
## Y_COORD_CD                   82.293810      92435.57403
```

```
rf_train <- predict(model_rf, train_set)

#rf_train

confusionMatrix(rf_train, as.factor(train_set$BORO_NM))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2     3     4     5
##          1 55243    32    86     2     0
##          2     5 73050    28    63     4
##          3    14    34 59966    50     0
##          4     2    46    15 48968     0
##          5     0     1     0     0 11328
##
## Overall Statistics
##
##                Accuracy : 0.9985
##                  95% CI : (0.9983, 0.9986)
##     No Information Rate : 0.2939
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.998
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.9996   0.9985   0.9979   0.9977  0.99965
## Specificity            0.9994   0.9994   0.9995   0.9997  1.00000
## Pos Pred Value         0.9978   0.9986   0.9984   0.9987  0.99991
## Neg Pred Value         0.9999   0.9994   0.9993   0.9994  0.99998
## Prevalence             0.2220   0.2939   0.2414   0.1972  0.04552
## Detection Rate         0.2219   0.2934   0.2409   0.1967  0.04551
## Detection Prevalence   0.2224   0.2938   0.2413   0.1970  0.04551
## Balanced Accuracy      0.9995   0.9989   0.9987   0.9987  0.99982
```

```r
rf_test <- predict(model_rf, test_set)

#rf_test

confusionMatrix(rf_test, as.factor(test_set$BORO_NM))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2     3     4     5
##          1 23678    24   174    11     0
##          2     9 30923    72   186    11
##          3    59   115 25327   116     0
##          4     3   109    42 20832     0
##          5     0     9     3     0  4985
##
## Overall Statistics
##
##                Accuracy : 0.9912
##                  95% CI : (0.9906, 0.9917)
##     No Information Rate : 0.2923
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9885
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.9970   0.9918   0.9886   0.9852  0.99780
## Specificity            0.9975   0.9963   0.9964   0.9982  0.99988
## Pos Pred Value         0.9913   0.9911   0.9887   0.9927  0.99760
## Neg Pred Value         0.9991   0.9966   0.9964   0.9963  0.99989
## Prevalence             0.2226   0.2923   0.2401   0.1982  0.04683
## Detection Rate         0.2219   0.2898   0.2374   0.1953  0.04673
## Detection Prevalence   0.2239   0.2925   0.2401   0.1967  0.04684
## Balanced Accuracy      0.9972   0.9940   0.9925   0.9917  0.99884
```

```r
var(as.numeric(rf_test), as.numeric(test_set$BORO_NM))
```

```
## [1] 1.35652
```

```r
bias(as.numeric(rf_test), as.numeric(test_set$BORO_NM))
```

```
## [1] -0.004255399
```