Chicago Crime

Xin Guan, Vera Hudak, Yuqi Zhang

2023-11-24

```r
# Packages required
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(magrittr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats 1.0.0     v stringr 1.5.0
## v ggplot2 3.4.4     v tibble  3.2.1
## v purrr   1.0.2     v tidyr   1.3.0
## v readr   2.1.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::extract()   masks magrittr::extract()
## x dplyr::filter()    masks stats::filter()
## x dplyr::lag()       masks stats::lag()
## x purrr::set_names() masks magrittr::set_names()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(ggplot2)
library(caret)
```

```
## 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(MLmetrics)
```

```
##
## Attaching package: 'MLmetrics'
##
## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
##
## The following object is masked from 'package:base':
##
##     Recall
```

```r
library(ROSE)
```

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

```r
library(knitr)
#library(sf)
library(spatstat)
```

```
## Loading required package: spatstat.data
## Loading required package: spatstat.geom
## spatstat.geom 3.2-7
##
## Attaching package: 'spatstat.geom'
##
## The following object is masked from 'package:pROC':
##
```

```
##     coords
##
## Loading required package: spatstat.random
## spatstat.random 3.2-2
## Loading required package: spatstat.explore
## Loading required package: nlme
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:dplyr':
##
##     collapse
##
## spatstat.explore 3.2-5
##
## Attaching package: 'spatstat.explore'
##
## The following objects are masked from 'package:pROC':
##
##     auc, roc
##
## The following object is masked from 'package:lattice':
##
##     panel.histogram
##
## Loading required package: spatstat.model
## Loading required package: rpart
## spatstat.model 3.2-8
##
## Attaching package: 'spatstat.model'
##
## The following object is masked from 'package:lattice':
##
##     panel.histogram
##
## Loading required package: spatstat.linnet
## spatstat.linnet 3.1-3
##
## spatstat 3.0-7
## For an introduction to spatstat, type 'beginner'
```

```r
library(ggplot2)
library(tidyr)
library(dplyr)
library(readr)
library(e1071)
library(doParallel)
```

```
## Loading required package: foreach
##
## Attaching package: 'foreach'
##
## The following objects are masked from 'package:purrr':
##
```

```
##     accumulate, when
##
## Loading required package: iterators
## Loading required package: parallel
```

```r
library(mltools)
```

```
##
## Attaching package: 'mltools'
##
## The following object is masked from 'package:e1071':
##
##     skewness
##
## The following object is masked from 'package:tidyr':
##
##     replace_na
```

```r
library(caret)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
##
## The following object is masked from 'package:purrr':
##
##     cross
##
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```r
library(ROCR)
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following object is masked from 'package:spatstat.geom':
##
##     shift
##
## The following object is masked from 'package:purrr':
##
##     transpose
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following objects are masked from 'package:lubridate':
##
```

```
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```r
library(DescTools)
```

```
##
## Attaching package: 'DescTools'
##
## The following object is masked from 'package:data.table':
##
##     %like%
##
## The following object is masked from 'package:foreach':
##
##     %:%
##
## The following objects are masked from 'package:MLmetrics':
##
##     AUC, Gini, MAE, MAPE, MSE, RMSE
##
## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
```

```r
library(scales)
```

```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:kernlab':
##
##     alpha
##
## The following object is masked from 'package:spatstat.geom':
##
##     rescale
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
library(geosphere)
```

```
##
## Attaching package: 'geosphere'
##
## The following object is masked from 'package:spatstat.geom':
##
##     perimeter
```

The repository for this project can be found at the following link: https://github.com/XinnGuan/Chicago-Crime.

Data Processing

```
# Read data
df <- read.csv("OriginalData.csv")
```

In this investigation, we basically separate the variables into three parts, namely *Time*, *Space*, and *Type*.

For the Time-related variables, we focus on `Date`. By dividing the information from the original observations, we aim to find the most variant and significant indicator to simplify our model.

For the Space-related features, we select two main variables to describe the spatial information, i.e., `District` and `X(Y).coordinate`. We shall use one of them in our exploration process.

For the Type-related one, we have the variable `Primary.Type` as our indicator. We shall check the crime frequency among all types to have an initial understanding of the dataset.

Processing variable 'Date'

```
data <- df %>% select(ID,Date, Primary.Type, Location.Description, Arrest, District)
data %<>% filter(!Primary.Type %in% c("NON-CRIMINAL", "OTHER NARCOTIC VIOLATION", "PUBLIC INDECENCY", "
data$Date <- parse_datetime(data$Date, format = "%m/%d/%Y %I:%M:%S %p")
data %<>%
  mutate(Date = as.POSIXct(Date, format = "%m/%d/%Y %I:%M:%S %p"),
         time = format(as.POSIXct(Date), format = "%H"),
         date = as.Date(Date),
         month = format(as.Date(date, format = "%m/%d/%Y"), "%m"),
         DayofYear = yday(date),
         weekday = weekdays(date))
```

Our dataset has been filtered to exclude certain crime types, namely "NON-CRIMINAL," "OTHER NARCOTIC VIOLATION," "PUBLIC INDECENCY," "HUMAN TRAFFICKING," and "OBSCENITY" - which allows for a more focused analysis on specific criminal activities.

Subsequently, the `Date` column has been parsed and manipulated to extract various components. This includes the creation of new columns such as `time` (hour), `date` (date), `month` (month), `DayofYear` (day of the year), and `weekday` (day of the week).

Processing missing data

```
colSums(is.na(data))
```

```
##                   ID                 Date         Primary.Type
##                    0                    0                    0
## Location.Description               Arrest             District
##                    0                    0                    0
##                 time                 date                month
##                    0                    0                    0
##            DayofYear              weekday
##                    0                    0
```

We can see that the number of missing values for the variables `X.Coordinate`, `Y.Coordinate`, `Latitude`, and `Longtitude` are exactly the same, we can deduce that the coordinates are calculated from the latitude and longitude, so we don't need to include both pairs of location information. Also since the number of missing value is small compare to the number of data size, we will just eliminate the rows with missing values.

```
data %<>%  na.omit()
```

Expalnatory Data Analysis

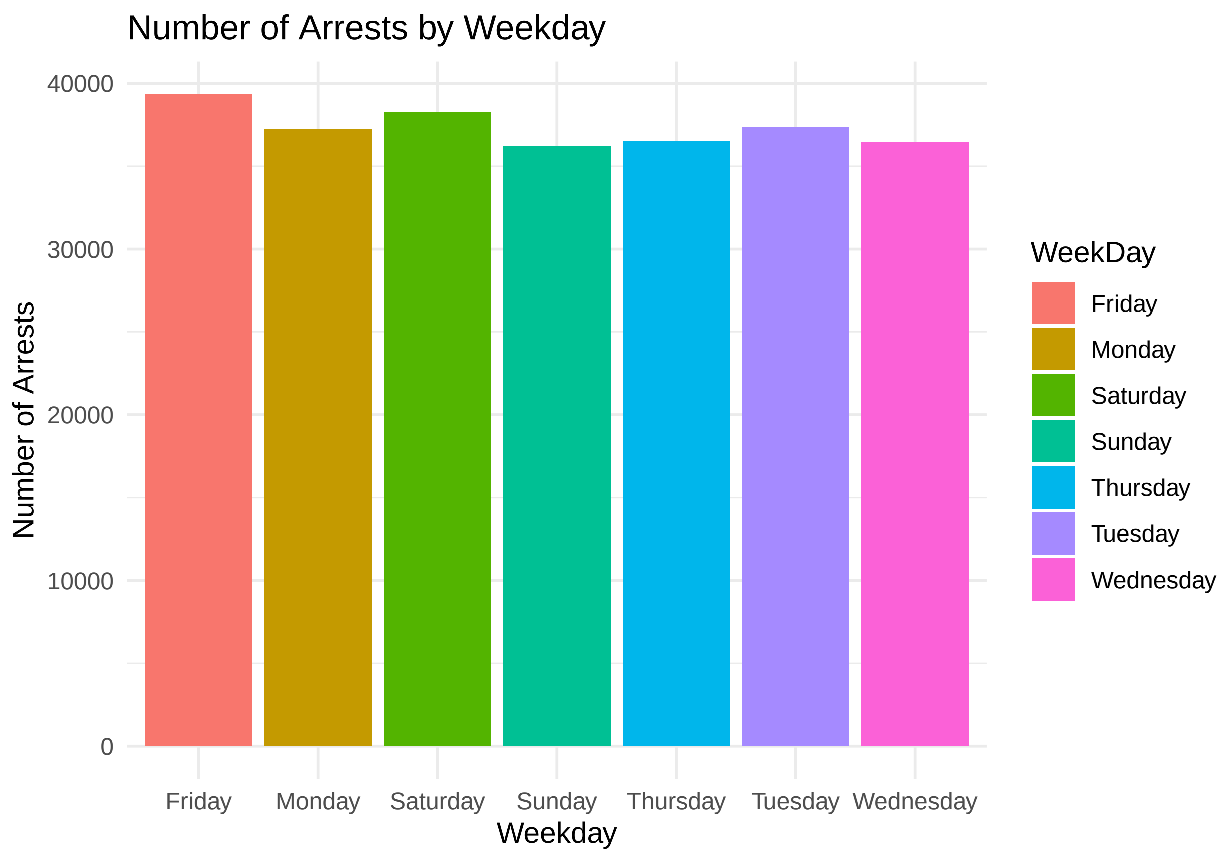
```
data$Primary.Type <- as.factor(data$Primary.Type)
data$time <- as.numeric(data$time)
data$District <- as.factor(data$District)
data$weekday <- as.factor(data$weekday)

data <- data %>% select(ID, Primary.Type, Arrest, District, time, DayofYear, weekday) %>% drop_na()
```

```
# Investigate of variables
df_vis <- df

df_vis$newDate <- mdy_hms(df_vis$Date)
df_vis$WeekDay <- weekdays(df_vis$newDate)
df_vis$DayOfMonth <- day(df_vis$newDate)
df_vis$DayOfYear <- yday(df_vis$newDate)
df_vis$Month <- lubridate::month(df_vis$newDate, label = TRUE, abbr = FALSE)
df_vis$TimeOfDay <- cut(
  hour(df_vis$newDate),
  breaks= c(-Inf, 5, 12, 17, 20, Inf),
  labels = c("Night", "Early Morning", "Morning", "Afternoon", "Evening"),
  include.lowest = TRUE
)

# Detailed Time
# This format of `Time` variable contains information of hour and minute only
df_vis$newDate <- as.POSIXct(df_vis$newDate, format = "%H:%M:%S")
df_vis$Hour <- hour(df_vis$newDate)
df_vis$Minute <- minute(df_vis$newDate)/60*10
df_vis$Time <- df_vis$Hour + df_vis$Minute/60
```

Weekdays

```
# ------------------------------------------------------------------------------
# Components of Dates
# ------------------------------------------------------------------------------
# Weekdays
```

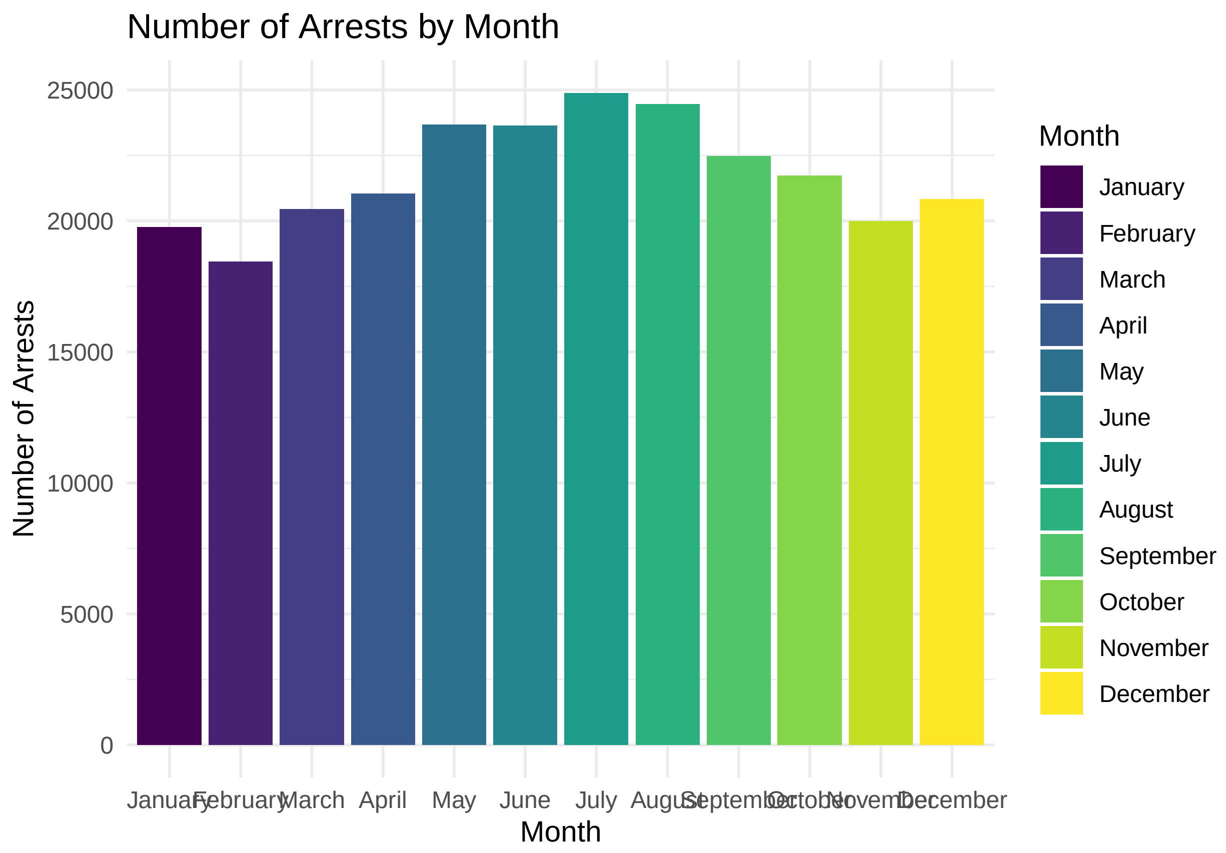
```r
arrests_by_weekday <- df_vis %>%
  group_by(WeekDay) %>%
  summarise(num_arrests = n())

ggplot(arrests_by_weekday, aes(x = WeekDay, y = num_arrests, fill = WeekDay)) +
  geom_bar(stat = "identity") +
  labs(title = "Number of Arrests by Weekday",
       x = "Weekday",
       y = "Number of Arrests") +
  theme_minimal()
```



The Figure provides a clear visualisation of the distribution of arrests across weekdays. Comparing the heights of the bars, we see that the arrest number generally stays steady on weekdays, while the number on Friday and Saturday may be a little higher than on other days - which corresponds to our intuitive understanding of organized crime and provides insights into potential areas for further investigation.

Month & Day-of-year

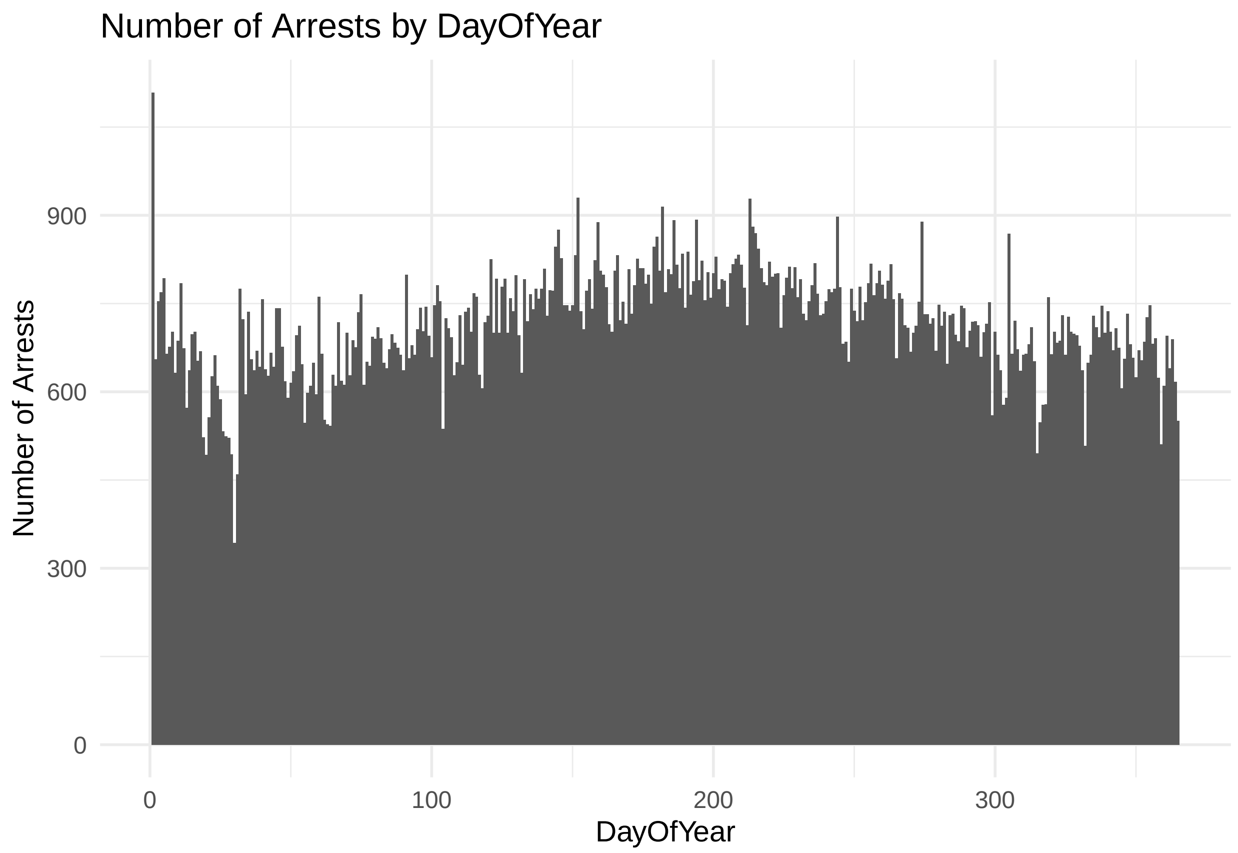
```r
# Month
arrests_by_Month <- df_vis %>%
  group_by(Month) %>%
  summarise(num_arrests = n())

ggplot(df_vis, aes(x = Month, fill = Month)) +
  geom_bar() +
  labs(title = "Number of Arrests by Month",
       x = "Month",
       y = "Number of Arrests") +
  theme_minimal()
```



Observing the distribution of arrests across different months, the pattern represents a trigonometric and periodic trend. See above, we conjecture this could be influenced by potential factors of arrests by month, which may consist of seasonal variations, holidays, or specific events. Due to this visualized information, we shall attempt the trigonometric transformation for the corresponding features in the regression model.

```
# DayOfYear
arrests_by_DayOfYear <- df_vis %>%
  group_by(DayOfYear) %>%
  summarise(num_arrests = n())

ggplot(df_vis, aes(x = DayOfYear, fill = DayOfYear)) +
  geom_bar() +
  labs(title = "Number of Arrests by DayOfYear",
       x = "DayOfYear",
       y = "Number of Arrests") +
  theme_minimal()
```

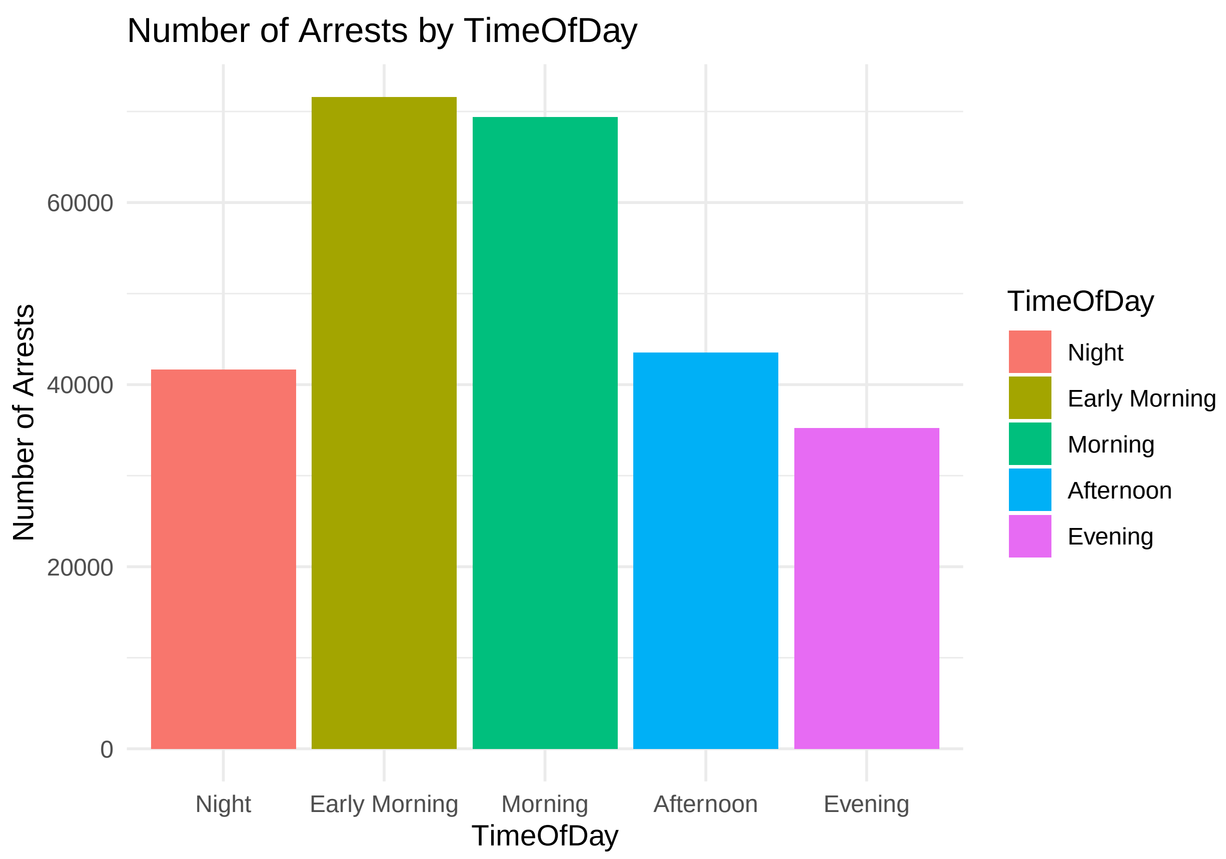
```
## Warning: The following aesthetics were dropped during statistical transformation: fill
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a ‘group‘ aesthetic or to convert a numerical
##   variable into a factor?
```



On the other hand, considering the bar chart by Day-of-year, we find that in the period of the given year 2019, the number of arrests conveys a periodic pattern as the bar chart by Month. It seems that the arrest actions concentrate on the middle of the year, while at the end and beginning of the year, the arrest is relatively decreased.

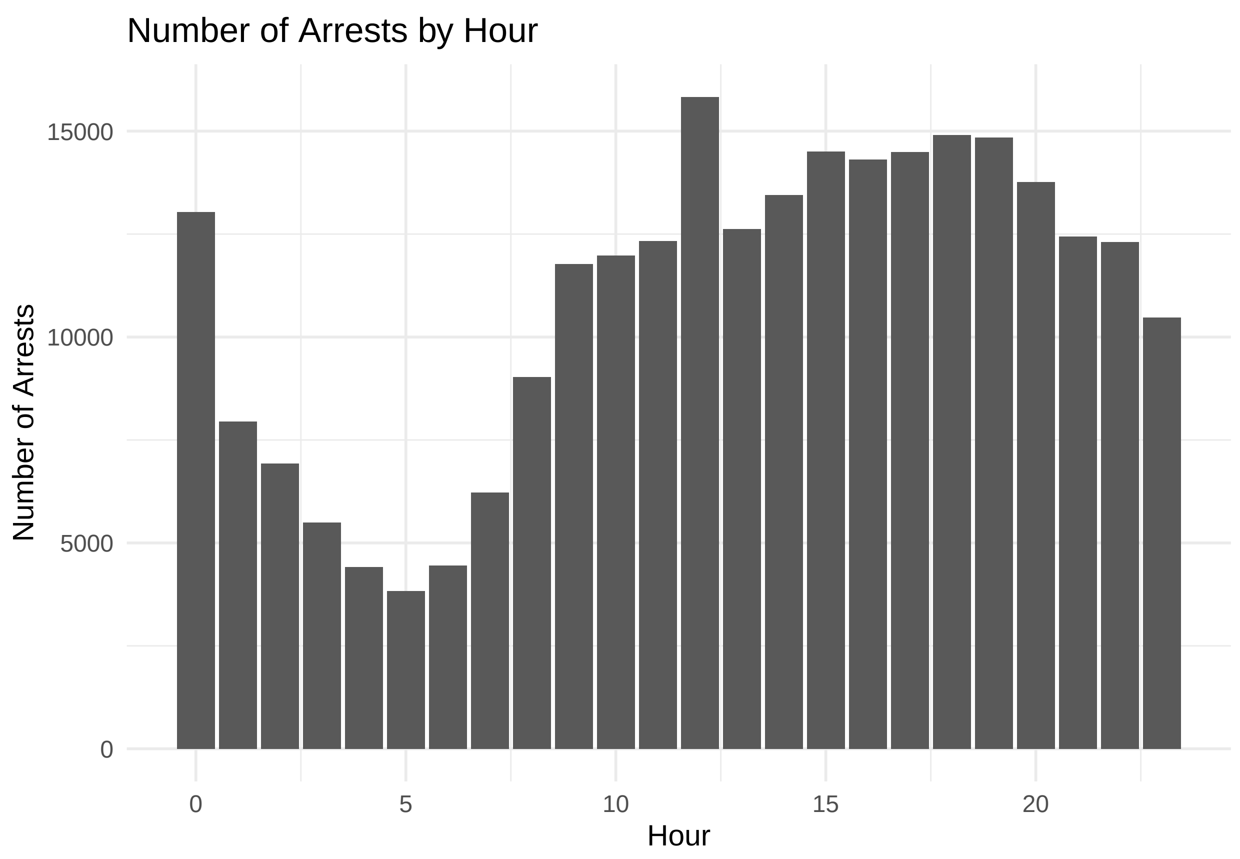
Hour & Time-of-day

```r
# TimeOfDay
arrests_by_TimeOfDay <- df_vis %>%
  group_by(TimeOfDay) %>%
  summarise(num_arrests = n())

ggplot(df_vis, aes(x = TimeOfDay, fill = TimeOfDay)) +
  geom_bar() +
  labs(title = "Number of Arrests by TimeOfDay",
       x = "TimeOfDay",
       y = "Number of Arrests") +
  theme_minimal()
```



```r
# hours
arrests_by_hour <- df_vis %>%
  group_by(Hour) %>%
  summarise(num_arrests = n())

ggplot(df_vis, aes(x = Hour, fill = Hour)) +
  geom_bar() +
  labs(title = "Number of Arrests by Hour",
       x = "Hour",
       y = "Number of Arrests") +
  theme_minimal()
```

```
## Warning: The following aesthetics were dropped during statistical transformation: fill
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a ‘group‘ aesthetic or to convert a numerical
##   variable into a factor?
```



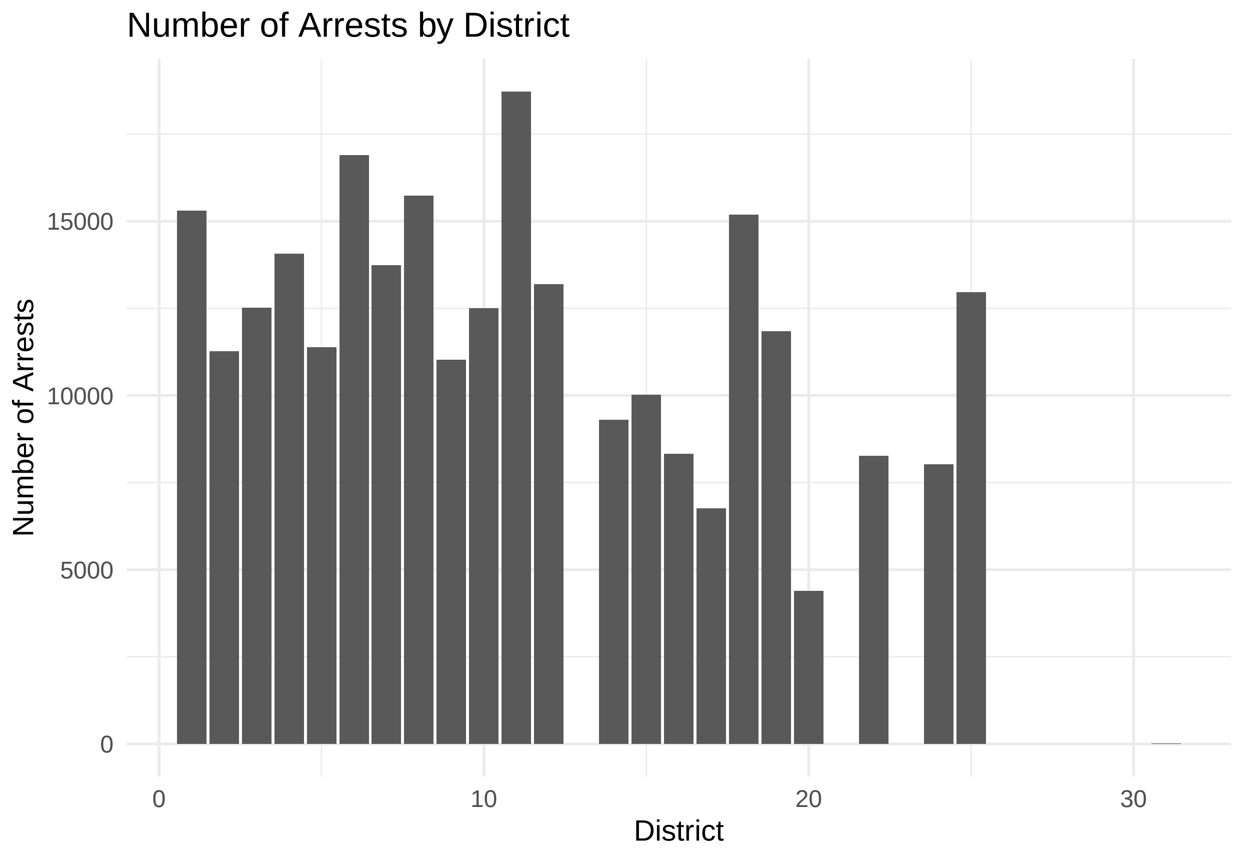
The variable `TimeOfDay` likely categorizes the hours of the day into distinct time ranges (e.g., morning, afternoon, evening, night) and provides a visual representation of how arrests are distributed throughout the day. We plot the above two figures. In fact, the variables `Hour` and `Time-of-day` essentially convey the same information. We examine the distribution of arrests across different times of the day and identify that in the morning time ranges, the arrests are the most common, whereas during the night, we have fewer arrests - and this corresponds to the information reflected by the bar chart of `Hour`. The consistency between `Time-of-day` and `Hour` underscores the reliability of the data and the coherence of the information regarding the timing of arrests. Accounting for the variable `Hour` may encompass more detailed information than the rough time ranges; we intend to use it as the "daily" description of time-related variables.

Space-related Variables

The two variables `District` and `X(Y).coordinate` both represent spatial knowledge, and we may use different features to assess the problems. Like the Time-related variables, they describe the situation at different levels of precision.

```r
# District
arrests_by_dist <- df_vis %>%
  group_by(District) %>%
  summarise(num_arrests = n())

ggplot(df_vis, aes(x = District, fill = District)) +
  geom_bar() +
  labs(title = "Number of Arrests by District",
       x = "District",
       y = "Number of Arrests") +
  theme_minimal()
```

```
## Warning: The following aesthetics were dropped during statistical transformation: fill
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a ‘group‘ aesthetic or to convert a numerical
##   variable into a factor?
```



Here we mainly consider the district-wise distribution to simplify our EDA. See above, we find some of the districts with significantly higher arrest numbers, while few of the districts even did not have arrest cases in the year 2019, which stands for variability between the districts of Chicago. To some extent, it might inform the insights of the law enforcement activities, like if there will be needed resources in such specific districts.

Type-related Variables

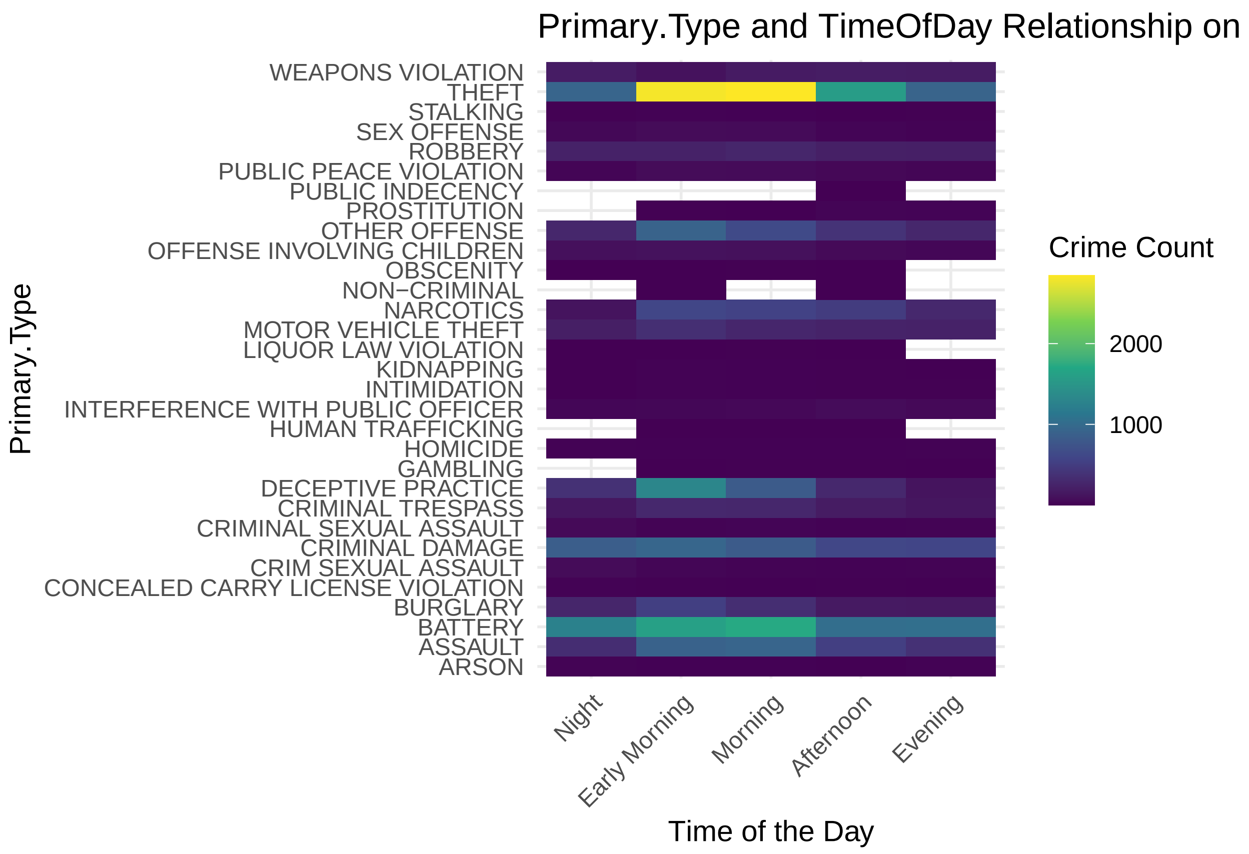
We plot a heatmap to understand the distribution of crime counts across different combinations of `Primary.Type` and `TimeOfDay` on the selected weekday. Here we take Monday as an example to illustrate the main features.

```
# ---------------------------------------------------------------------------
# Relations between `Primary.Type` and `TimeOfDay` in a specific `Weekday`
# ---------------------------------------------------------------------------
# Choose a specific weekday (e.g., "Monday")
```

```r
selected_weekday <- "Monday"
# Filter the data for the selected weekday
filtered_data <- df_vis %>% filter(WeekDay == selected_weekday)

# Summary table with the count of crimes for each combination of `Primary.Type` and `TimeOfDay`
crime_time_counts <- filtered_data %>%
  group_by(Primary.Type, TimeOfDay) %>%
  summarise(Count = n())
```

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

```r
ggplot(crime_time_counts, aes(x = TimeOfDay, y = Primary.Type, fill = Count)) +
  geom_tile() +
  scale_fill_viridis_c() +
  labs(title = paste("Primary.Type and TimeOfDay Relationship on", selected_weekday),
       x = "Time of the Day",
       y = "Primary.Type",
       fill = "Crime Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "right")
```



As shown in the above figure, we find that the type of Theft happened most commonly among all the other types. It also shows that the peak of theft is in the (early) morning. In addition, the types of battery and assault also often took place in 2019, and the time peak is in the mornings as well. The time ranges of evening and night hold the least common rate of crime, which conflicts with our intuitive understanding. The reasons behind this phenomenon need to be deeply analyzed and researched.

Binary Classification

Preparation

Feature Selection

Our first task is to predict whether arrest or not given time, location, and the crime type.

```r
data %<>% filter(District != "31")
sum(data$District=="31")
```

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

Define Cross-validation

```r
# Define a 5-fold cross validation
ctrl <- trainControl(method = "cv", number = 5, summaryFunction = twoClassSummary, classProbs = TRUE)
```

Cross-validation is a model evaluation technique that assesses the model performance. It involves partitioning the dataset into multiple subsets, training the model on some of these subsets, and evaluating its performance on the remaining. This process is repeated multiple times, and the performance metrics are averaged over the iterations.

Here we set up a 5-fold cross-validation, which means that the dataset was divided into five folds, and our model was trained over five times. In each iteration, the model was trained on four folds and tested on the remaining one. This process was repeated five times, ensuring that each fold was used as the test set exactly once.

The choice of 5-fold cross-validation helps provide a more robust assessment of the model performance. By averaging the evaluation metrics (such as ROC) over multiple folds, we obtain a more reliable estimate of how well the model is expected to perform on unseen data. Additionally, it helps detect if the model is overfitting or underfitting by assessing its consistency across different subsets of the dataset.

Imbalance Data Experiments with a Baseline Model

In this section, we further down-sample the dataset so that it only has approximately 5% data points with positive label. We will try different re-sampling techniques includes under-sampling, over-sampling, and ROSE-sampling (Random Over-Sampling Examples), which involves generating synthetic samples within the minority class, aiming to balance the class distribution and enhance the model's ability to learn from the minority class.

```r
set.seed(123)
# Separate positive and negative classes
true_class <- data %>% filter(Arrest == "true")
false_class <- data %>% filter(Arrest == "false")
```

```r
# Set the desired ratio of positive to negative samples
desired_positive_ratio <- 0.05

# Calculate the number of positive samples needed for downsampling
num_true_samples <- 0.05*nrow(false_class) / (1 - desired_positive_ratio)

# Randomly sample positive samples
true_class_downsampled <- true_class %>% sample_n(num_true_samples, replace = FALSE, seed = 42)

# Combine positive and downsampled negative samples
downsampled_dataset <- bind_rows(false_class, true_class_downsampled)

# Shuffle the dataset to mix positive and negative samples
set.seed(42)
downsampled_dataset <- downsampled_dataset[sample(nrow(downsampled_dataset)), ]

sum(downsampled_dataset[,"Arrest"]=="true")/dim(df)[1]
```

```
## [1] 0.04130081
```

```r
set.seed(123)
#use 80% of dataset as training set and 20% as test set
variables_to_convert <- c("Arrest", "Primary.Type", "weekday")

imbalanced_train <- downsampled_dataset %>% dplyr::sample_frac(0.80)
imbalanced_train[, variables_to_convert] <- lapply(imbalanced_train[, variables_to_convert], as.factor)
imbalanced_test  <- dplyr::anti_join(downsampled_dataset, imbalanced_train, by = 'ID')
imbalanced_test[, variables_to_convert] <- lapply(imbalanced_test[, variables_to_convert], as.factor)
imbalanced_train %<>% select(-"ID")
imbalanced_test %<>% select(-"ID")
```

```r
base_m <- train(Arrest ~ District + DayofYear + time +
    Primary.Type, data = imbalanced_train, method = "glm", family = "binomial", trControl = ctrl, metri
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
pred <- predict(base_m, imbalanced_test[,c("District", "DayofYear", "time", "Primary.Type")], type = "p
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
binary_predictions <- ifelse(pred[,2] >= 0.5, "true", "false")
# create confusion matrix
confusionMatrix(as.factor(binary_predictions), imbalanced_test$Arrest,
                mode = "everything",
                positive="true")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction false  true
##      false 41018  1460
##      true     16   686
##
##                Accuracy : 0.9658
##                  95% CI : (0.9641, 0.9675)
##     No Information Rate : 0.9503
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4687
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.31966
##             Specificity : 0.99961
##          Pos Pred Value : 0.97721
##          Neg Pred Value : 0.96563
##               Precision : 0.97721
##                  Recall : 0.31966
##                      F1 : 0.48174
##              Prevalence : 0.04970
##          Detection Rate : 0.01589
##    Detection Prevalence : 0.01626
##       Balanced Accuracy : 0.65964
##
##        'Positive' Class : true
##
```

```r
set.seed(123)
#over sampling
train_balanced_over <- ovun.sample(Arrest ~ District + DayofYear + time +
    Primary.Type, data = imbalanced_train, method = "over")$data
train_balanced_over$Arrest <- as.factor(train_balanced_over$Arrest)
train_balanced_under <- ovun.sample(Arrest ~ District + DayofYear + time +
    Primary.Type, data = imbalanced_train, method = "under")$data
train_balanced_under$Arrest <- as.factor(train_balanced_under$Arrest)
train.rose <- ROSE(Arrest ~ District + DayofYear + time + Primary.Type, data = imbalanced_train)$data
train.rose$Arrest <- as.factor(train.rose$Arrest)
table(train.rose$Arrest)
```

```
##
## false  true
## 86481 86239
```

```r
table(train_balanced_over$Arrest)
```

```
##
##  false   true
## 164071 163812
```

```r
table(train_balanced_under$Arrest)
```

```
##
## false  true
##  8648  8649
```

Upon examination of the resampled dataset, it becomes apparent that under-sampling has resulted in the removal of a substantial number of data points with positive labels, potentially leading to information loss. Conversely, the repeated sampling of data points with negative labels may introduce redundancy. In contrast, ROSE-sampling has successfully generated a dataset with a comparable number of data points to the original dataset, addressing concerns related to drastic imbalances in class representation.

```r
base_m_under <- train(Arrest ~District + DayofYear + time + Primary.Type, data = train_balanced_under, 
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
base_m_over <- train(Arrest ~ District + DayofYear + time + Primary.Type, data = train_balanced_over, m
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
base_m_rose <- train(Arrest ~ District + DayofYear + time + Primary.Type, data = train.rose, method = "
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
```

```
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
pred_under <- predict(base_m_under, imbalanced_test[,c("District", "DayofYear", "time", "Primary.Type")]
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
pred_over <- predict(base_m_over, imbalanced_test[,c("District", "DayofYear", "time", "Primary.Type")],
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
pred_rose <- predict(base_m_rose, imbalanced_test[,c("District", "DayofYear", "time", "Primary.Type")],
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
# Create a ROC curve object for each prediction
roc_curve <- pROC::roc(imbalanced_test$Arrest, pred[,2])
```

```
## Setting levels: control = false, case = true

## Setting direction: controls < cases
```

```
#the roc function name is used in several packages, `pROC::roc` is used to ensure the roc function is f
roc_curve_under <- pROC::roc(imbalanced_test$Arrest, pred_under[,2])
```

```
## Setting levels: control = false, case = true
## Setting direction: controls < cases
```

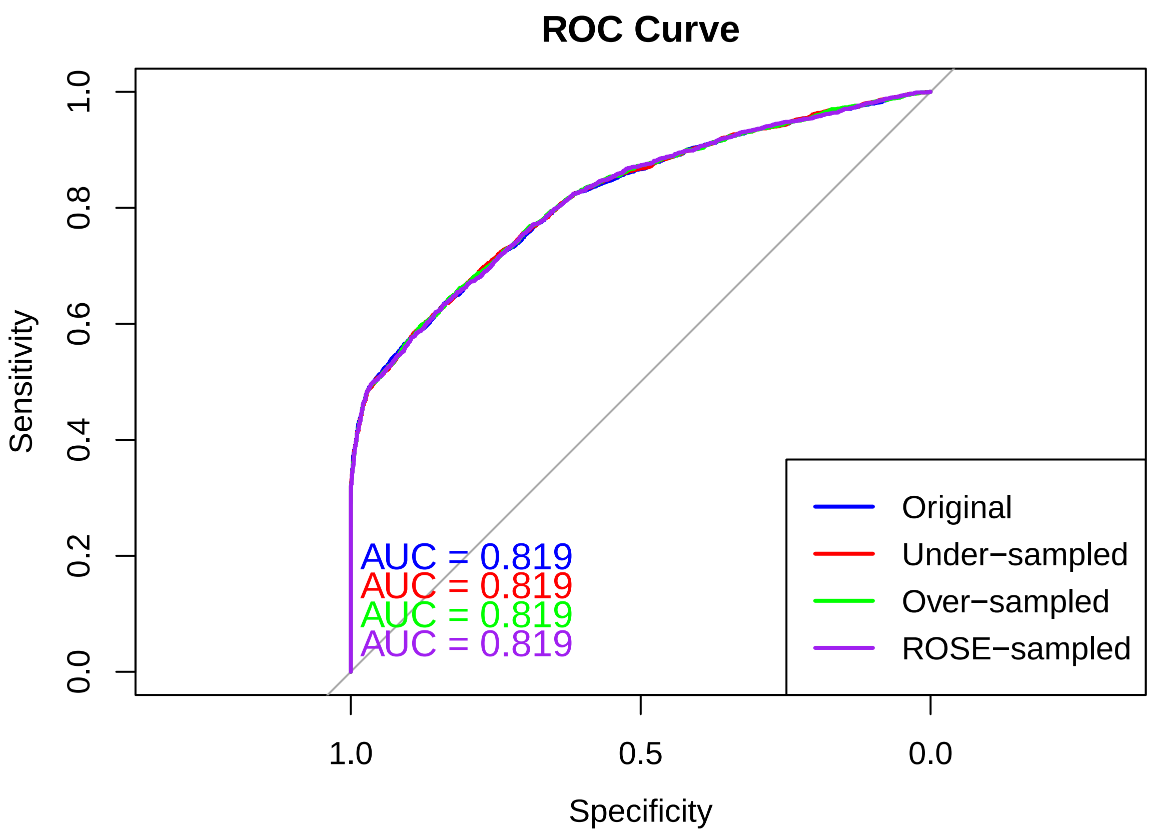
```r
roc_curve_over <- pROC::roc(imbalanced_test$Arrest, pred_over[,2])
```

```
## Setting levels: control = false, case = true
## Setting direction: controls < cases
```

```r
roc_curve_rose <- pROC::roc(imbalanced_test$Arrest, pred_rose[,2])
```

```
## Setting levels: control = false, case = true
## Setting direction: controls < cases
```

```r
# Plot the first ROC curve
plot(roc_curve, col = "blue", lwd = 2, main = "ROC Curve")

# Add AUC to the plot for the first curve
auc_value <- pROC::auc(roc_curve)
text(0.8, 0.2, paste("AUC =", round(auc_value, 3)), col = "blue", cex = 1.2)

# Add the other ROC curves to the plot
lines(roc_curve_under, col = "red", lwd = 2)
lines(roc_curve_over, col = "green", lwd = 2)
lines(roc_curve_rose, col = "purple", lwd = 2)

# Add AUC values for the other curves
text(0.8, 0.15, paste("AUC =", round(pROC::auc(roc_curve_under), 3)), col = "red", cex = 1.2)
text(0.8, 0.1, paste("AUC =", round(pROC::auc(roc_curve_over), 3)), col = "green", cex = 1.2)
text(0.8, 0.05, paste("AUC =", round(pROC::auc(roc_curve_rose), 3)), col = "purple", cex = 1.2)

legend("bottomright", legend = c("Original", "Under-sampled", "Over-sampled","ROSE-sampled"),
       col = c("blue", "red", "green","purple"), lty = 1, lwd = 2)
```



Here, wwe calculate the confusion matrices for all four logistic regression models, presenting the resulting metrics in columns for convenient comparison.

```
# Original Predictions
original_predictions <- ifelse(pred[,2] >= 0.5, "true", "false")
conf_matrix_original <- confusionMatrix(as.factor(original_predictions), imbalanced_test$Arrest,
                                        mode = "everything", positive = "true")
metrics_original <- conf_matrix_original$byClass

# Under Predictions
under_predictions <- ifelse(pred_under[,2] >= 0.5, "true", "false")
conf_matrix_under <- confusionMatrix(as.factor(under_predictions), imbalanced_test$Arrest,
                                     mode = "everything", positive = "true")
metrics_under <- conf_matrix_under$byClass

# Over Predictions
over_predictions <- ifelse(pred_over[,2] >= 0.5, "true", "false")
conf_matrix_over <- confusionMatrix(as.factor(over_predictions), imbalanced_test$Arrest,
                                    mode = "everything", positive = "true")
metrics_over <- conf_matrix_over$byClass

# Rose Predictions
rose_predictions <- ifelse(pred_rose[,2] >= 0.5, "true", "false")
conf_matrix_rose <- confusionMatrix(as.factor(rose_predictions), imbalanced_test$Arrest,
                                    mode = "everything", positive = "true")
metrics_rose <- conf_matrix_rose$byClass
```

```
# Combine Metrics into a Data Frame
metrics_df <- data.frame(
  Scenario = c("Original", "Under-sampled", "Over-sampled", "Rose-sampled"),
  Sensitivity = c(metrics_original["Sensitivity"], metrics_under["Sensitivity"], metrics_over["Sensitivi
  Specificity = c(metrics_original["Specificity"], metrics_under["Specificity"], metrics_over["Specific
  Precision = c(metrics_original["Pos Pred Value"], metrics_under["Pos Pred Value"], metrics_over["Pos P
  Positive_Predicted_Value = c(metrics_original["Pos Pred Value"], metrics_under["Pos Pred Value"], metr
  Negative_Predictive_Value = c(metrics_original["Neg Pred Value"], metrics_under["Neg Pred Value"], me
  Balanced_Accuracy = c(metrics_original["Balanced Accuracy"], metrics_under["Balanced Accuracy"], metri
  F1_Score = c(metrics_original["F1"], metrics_under["F1"], metrics_over["F1"], metrics_rose["F1"])
)

# Transpose the Data Frame to Swap Rows and Columns
transposed_metrics_df <- t(metrics_df)

# Display Transposed Metrics Data Frame
print(transposed_metrics_df)
```

```
##                           [,1]        [,2]            [,3]
## Scenario                  "Original"  "Under-sampled" "Over-sampled"
## Sensitivity               "0.3196645" "0.6211556"     "0.6216216"
## Specificity               "0.9996101" "0.8498075"     "0.8472243"
## Precision                 "0.9772080" "0.1778282"     "0.1754571"
## Positive_Predicted_Value  "0.9772080" "0.1778282"     "0.1754571"
## Negative_Predictive_Value "0.9656293" "0.9772167"     "0.9771763"
## Balanced_Accuracy         "0.6596373" "0.7354816"     "0.7344229"
## F1_Score                  "0.4817416" "0.2764987"     "0.2736691"
##                           [,4]
## Scenario                  "Rose-sampled"
## Sensitivity               "0.6202237"
## Specificity               "0.8529025"
## Precision                 "0.1806706"
## Positive_Predicted_Value  "0.1806706"
## Negative_Predictive_Value "0.9772429"
## Balanced_Accuracy         "0.7365631"
## F1_Score                  "0.2798276"
```

All three re-sampling techniques exhibit a notable increase in sensitivity, as anticipated, while the specificity experiences a corresponding decrease, reflecting a trade-off. In the context of highly imbalanced data, this trade-off is deemed satisfactory. The metrics demonstrate close proximity across all three re-sampling techniques.

F-beta Score

Given the current highly imbalanced nature of the data, the effectiveness of comparing models using the AUC curve diminishes. It's advisable to explore alternative evaluation metrics. The provided code below computes the F-β score, an generalization of the F-1 score, the F-1 score is included in the output of `confusionMatrix()` function. The parameter $\beta \geq 0$ is the weight assigned to recall, in the case, we should choose $\beta > 1$, so we prioritize the recall.

```
f_beta_score <- function(X, y, model, beta, threshold=0.5) {
  predictions <- predict(model, X, type = "prob")[,2]
  predicted_classes <- ifelse(predictions > threshold, "true", "false")
  predicted_classes <- as.factor(predicted_classes)
  pr <- confusionMatrix(data = predicted_classes, reference = y,
                        mode = "everything", positive="true")

  precision <- as.numeric(pr$byClass["Precision"])
  recall <- as.numeric(pr$byClass["Recall"])

  f_beta <- ((1 + beta^2) * precision * recall) / (beta^2 * precision + recall)
  return(f_beta)
}
```

```
f2 <- f_beta_score(X=imbalanced_test[,c("District", "DayofYear", "time", "Primary.Type")], y=imbalanced_
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
f2_under <- f_beta_score(X=imbalanced_test[,c("District", "DayofYear", "time", "Primary.Type")], y=imbal
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
f2_over <- f_beta_score(X=imbalanced_test[,c("District", "DayofYear", "time", "Primary.Type")], y=imbala
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
print(c(f2,f2_under,f2_over))
```

```
## [1] 0.3281572 0.5668073 0.5662417
```

Logistic Rregression

Training Set and Testing Set

In this case, we randomly select 80% of the rows from the original dataset, creating the testing set by excluding the rows that were included in the training set. This ensures that the testing set is independent of the training set.

```
#use 80% of dataset as training set and 20% as test set
variables_to_convert <- c("Arrest", "Primary.Type", "weekday")
train <- data %>% dplyr::sample_frac(0.80)
train[, variables_to_convert] <- lapply(train[, variables_to_convert], as.factor)
test  <- dplyr::anti_join(data, train, by = 'ID')
test[, variables_to_convert] <- lapply(test[, variables_to_convert], as.factor)
train %<>% select(-"ID")
test %<>% select(-"ID")
```

To predict the probability of arrest, we initially intend to build up a baseline model as a comparison to help evaluate other methods. After that, we hope to involve some other variables indicating the spatial and time information, or some other interactions to improve the baseline model and achieve better classification performance.

Logistic Regression Model

Baseline Model For the baseline model, we only select the variables `District`, `DayOfYear`, `Hour`, and `Primary.Type`:

$$\text{Baseline Model: Arrests} \sim \text{District} + \text{DayOfYear} + \text{Hour} + \text{Primary.Type}$$

where the first one indicates the spatial information, the second and third ones represent the time information yearly and daily, and the last indicator describes the crime types. We would like to use this model as our baseline.

```
base_m <- train(Arrest ~ District + DayofYear + time + Primary.Type,
                data = train, method = "glm", family = "binomial", trControl = ctrl, metric = "ROC")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

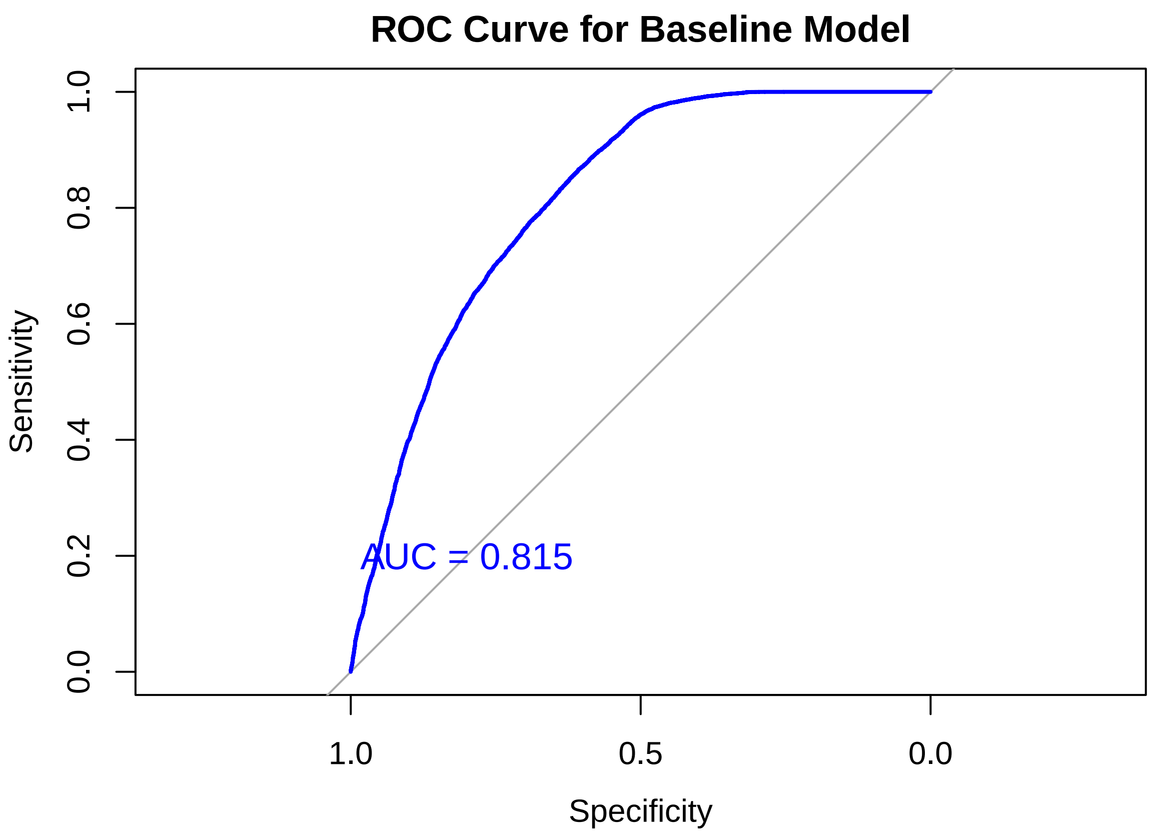
```
pred <- predict(base_m, test[,c("District", "DayofYear", "time", "Primary.Type")], type = "prob")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```r
binary_predictions <- ifelse(pred[,2] >= 0.5, "true", "false")
# create confusion matrix
confusionMatrix(as.factor(binary_predictions), test$Arrest,
                mode = "everything",
                positive="true")
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction false  true
##      false 40079  5853
##      true   1070  5252
##
##                Accuracy : 0.8675
##                  95% CI : (0.8646, 0.8704)
##     No Information Rate : 0.7875
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5303
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4729
##             Specificity : 0.9740
##          Pos Pred Value : 0.8307
##          Neg Pred Value : 0.8726
##               Precision : 0.8307
##                  Recall : 0.4729
##                      F1 : 0.6027
##              Prevalence : 0.2125
##          Detection Rate : 0.1005
##    Detection Prevalence : 0.1210
##       Balanced Accuracy : 0.7235
##
##        'Positive' Class : true
##
```

```r
roc_curve <- pROC::roc(test$Arrest, pred[,2],levels = c("true", "false"),direction = ">")
plot(roc_curve, main = "ROC Curve for Baseline Model", col = "blue", lwd = 2)

auc_curve <- pROC::auc(roc_curve)
text(0.8, 0.2, paste("AUC =", round(auc_curve, 3)), col = "blue", cex = 1.2)
```



The trained baseline model is applied to the test dataset to generate predictions, from which we can assess its performance by various metrics, see above results.

The baseline model demonstrates relatively good overall performance, with an accuracy of 86.6%. The ROC curve and AUC further illustrate the ability of the model to discriminate between positive and negative instances. Sensitivity, specificity, and other performance metrics provide a comprehensive understanding of the model's strengths and limitations.

The model exhibits high specificity (97.46%), indicating a strong ability to correctly identify non-arrest instances. On the other hand, sensitivity (47.36%) suggests the model's capability to identify arrest cases, though there is room for improvement in capturing true positives. Precision (83.78%) reflects the reliability of the positive predictions, while the negative predictive value (86.99%) indicates the accuracy of the negative predictions.

Feature Transform and Model Selection According to our EDA earlier, we aim to improve the model performance on the basis of the existing baseline.

The improved regression model is developed by introducing quadratic terms for the variables `Hour` and `DayOfYear`. Apart from that, the improved model is trained and evaluated using the same features as the baseline model, with the addition of squared terms. That is to say:

$$\text{Improved Model: Arrests} \sim \text{District} + \text{DayOfYear}^2 + \text{Hour}^2 + \text{Primary.Type}$$

```
train$time_t <- train$time^2
test$time_t <- test$time^2
train$DayofYear_t <- train$DayofYear^2
```

```r
test$DayofYear_t <- test$DayofYear^2

logistic_im_2 <- train(Arrest ~ District + DayofYear_t + weekday + time_t + Primary.Type,
                       data = train, method = "glm", family = "binomial", trControl = ctrl, metric = "R
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

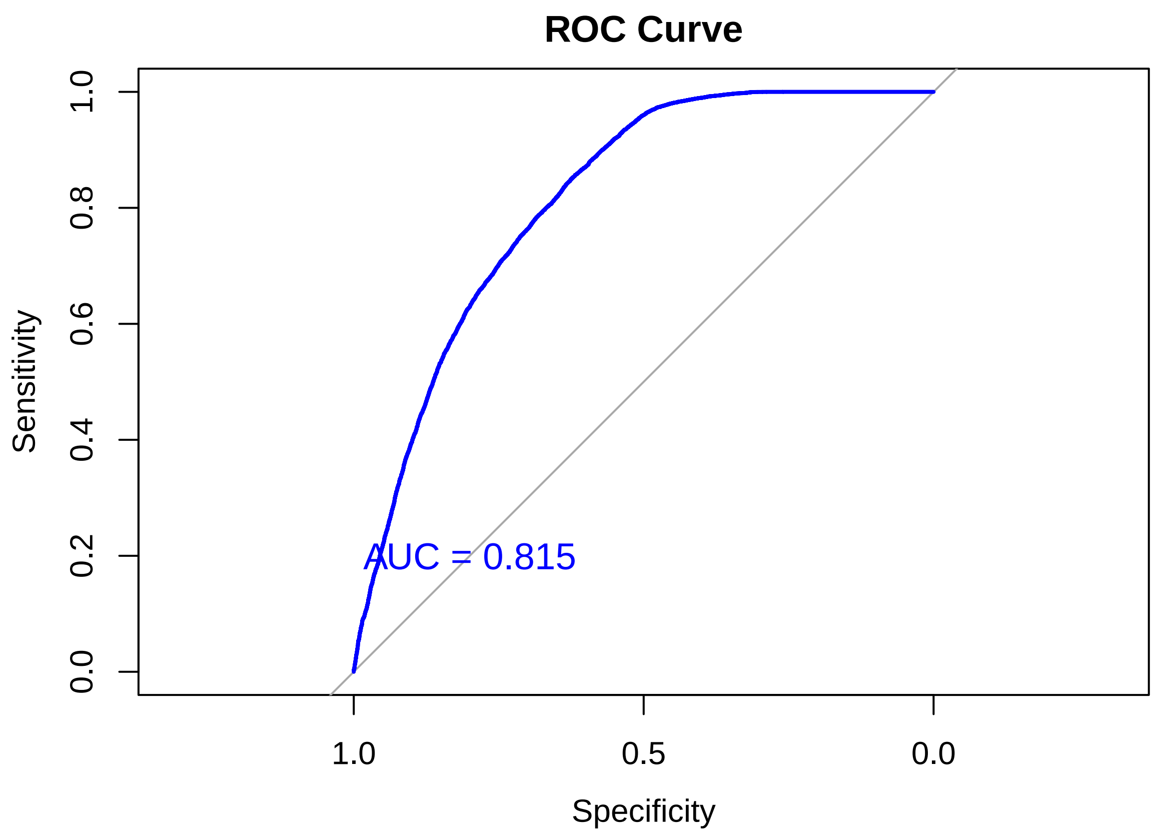
```r
# Cannot use `Time` - result in multicollineararity

pred_im_2 <- predict(logistic_im_2, test[,c("District", "DayofYear_t", "weekday", "time_t", "Primary.Ty
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```r
binary_predictions_im_2 <- ifelse(pred_im_2[,2] >= 0.5, "true", "false")
# create confusion matrix
confusionMatrix(as.factor(binary_predictions_im_2), test$Arrest,
                mode = "everything",
                positive="true")
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction false  true
##      false 40090  5873
##      true   1059  5232
##
##                Accuracy : 0.8673
##                  95% CI : (0.8644, 0.8702)
##     No Information Rate : 0.7875
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5291
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4711
##             Specificity : 0.9743
##          Pos Pred Value : 0.8317
##          Neg Pred Value : 0.8722
##               Precision : 0.8317
##                  Recall : 0.4711
##                      F1 : 0.6015
##              Prevalence : 0.2125
##          Detection Rate : 0.1001
##    Detection Prevalence : 0.1204
##       Balanced Accuracy : 0.7227
##
##        'Positive' Class : true
##
```

```r
roc_2 <- pROC::roc(test$Arrest, pred_im_2[,2], levels = c("true", "false"),direction = ">")
plot(roc_2, main = "ROC Curve", col = "blue", lwd = 2)

auc_2 <- pROC::auc(roc_2)
text(0.8, 0.2, paste("AUC =", round(auc_2, 3)), col = "blue", cex = 1.2)
```



Using the same evaluating metrics, we obtain the results see above.

Both models demonstrate high accuracy, sensitivity, and specificity. The improved model introduces quadratic terms for `Hour` and `DayofYear`, aiming to capture non-linear relationships in the data. It slightly adjusts the performance metrics, with marginal changes in accuracy, sensitivity, and specificity compared to the baseline model. The AUC values for both models are also very close, indicating similar discrimination capabilities.

While both models exhibit strong predictive performance, the improved model captures potential non-linear relationships with quadratic terms, providing more flexibility in capturing complex patterns.

Further exploration and feature engineering may lead to more substantial improvements in model performance. Due to the negligible improvements, we are considering changing our prediction method to approaches like SVM (support vector machine) to obtain better performance.

Support Vector Machine

Next, we use SVM to predict the outcome of a crime (arrest or no arrest) using the variables `Primary.Type`, `Disctrict`, `time` and `weekday`. As the relationship between outcome and variables is unclear, we use SVM with radial basis kernel. For SVM, the variables need to be one hot encoded.

```r
# One hot encode data
data_svm <- data %>% select(-c(ID,DayofYear))
data_svm <- as.data.table(data_svm)
data_svm <- one_hot(data_svm)
```

Support Vector Machines (SVMs) face challenges with large datasets due to their cubic time complexity during training, making them computationally expensive as the dataset size increases. Additionally, SVMs often require storing the entire training dataset in memory, which can be impractical for large datasets, leading to reliance on disk storage and slowing down the training process. Furthermore, SVMs exhibit scalability issues, as the optimization problem they solve is quadratic in the number of samples, diminishing their efficiency for large-scale datasets. So when fitting our models, we will use a random sample from our data set, using 3 types of sampling methods.

1. Proportional sampling: this sampling method generates a sample where the proportion of each class emulates the original data set.

We take 6% of the data to train our model on, keeping the class proportions as in the original data set and use it to fit the SVM.

```r
# Take proportional training sample
set.seed(123) # For reproducibility
index <- createDataPartition(data_svm$Arrest, p = .06, list = TRUE)
train <- data_svm[index$Resample,]
train %>% dplyr::count(Arrest)
```

```
##    Arrest     n
## 1:  false 12307
## 2:   true  3370
```

```r
# CV seeds
seeds <- vector(mode = "list", length = 6)
for(i in 1:6) {
  seeds[[i]] <- sample.int(n = 1000, 3) }

# CV function with 5-fold cross-validation
control <- trainControl(method = 'cv', number = 5, allowParallel = TRUE, seeds = seeds)
```

```r
# SVM with proportional sample

set.seed(123)

cl <- makePSOCKcluster(detectCores() - 1)

registerDoParallel(cl)

svm.model <- train(Arrest ~ ., data = train, trControl = control, method = 'svmRadial', family = binomi
```

```
## Warning in .local(x, ...): Variable(s) ‘’ constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) ‘’ constant. Cannot scale data.
```

```r
stopCluster(cl)

# Test data
index <- createDataPartition(data_svm$Arrest, p = .03, list = TRUE)

test <- data_svm[index$Resample,]

# Confusion matrix
```

```
test$Arrest <- as.factor(test$Arrest)

conf_matrix1 <- confusionMatrix(data = predict(svm.model, newdata = test), reference = test$Arrest, pos

conf_matrix1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction false true
##      false  6023  953
##      true    131  732
##
##                Accuracy : 0.8617
##                  95% CI : (0.8539, 0.8693)
##     No Information Rate : 0.785
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5021
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.43442
##             Specificity : 0.97871
##          Pos Pred Value : 0.84820
##          Neg Pred Value : 0.86339
##               Precision : 0.84820
##                  Recall : 0.43442
##                      F1 : 0.57457
##              Prevalence : 0.21495
##          Detection Rate : 0.09338
##    Detection Prevalence : 0.11009
##       Balanced Accuracy : 0.70657
##
##        'Positive' Class : true
##
```

2. Downsampling : downsampling is a mechanism that reduces the count of training samples falling under the majority class, by randomly removing instances from the majority class until the class distribution is balanced. This method is straightforward but may discard potentially valuable information. We make sure that the model is trained using a random sample which is the same size as the sample used to fit the model before.

```
# Down sample the dataset
set.seed(123) # For reproducibility

data_svm$Arrest = as.factor(data_svm$Arrest)

down_sample <- downSample(x = data_svm, y = data_svm$Arrest)

down_sample %>% count(Arrest)
```

```
##   Arrest     n
## 1  false 56166
## 2   true 56166
```

```
# Take random tarining data from down sampled data
index <- createDataPartition(down_sample$Arrest, p = nrow(train)/nrow(down_sample), list = TRUE)

train2 <- down_sample[index$Resample,] %>% select(-Class)

train2 %>% count(Arrest)
```

```
##   Arrest    n
## 1  false 7839
## 2   true 7839
```

```
set.seed(123) # For reproducibility

# Fit SVM using down sampling
cl <- makePSOCKcluster(detectCores() - 1)

registerDoParallel(cl)

svm.model2 <- train(Arrest ~ ., data = train2, trControl = control, method = 'svmRadial', family = binor
```

```
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.
```

```
stopCluster(cl)

# Test data
index <- createDataPartition(data_svm$Arrest, p = .03, list = TRUE)

test <- data_svm[index$Resample,]

# Confusion matrix
test$Arrest <- as.factor(test$Arrest)

conf_matrix2 <- confusionMatrix(data = predict(svm.model2, newdata = test), reference = test$Arrest, pos

conf_matrix2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction false true
##      false  5199  631
##      true    955 1054
##
##                Accuracy : 0.7977
##                  95% CI : (0.7886, 0.8065)
##     No Information Rate : 0.785
##     P-Value [Acc > NIR] : 0.003212
```

```
##
##                   Kappa : 0.4396
##
##  Mcnemar's Test P-Value : 5.039e-16
##
##             Sensitivity : 0.6255
##             Specificity : 0.8448
##          Pos Pred Value : 0.5246
##          Neg Pred Value : 0.8918
##               Precision : 0.5246
##                  Recall : 0.6255
##                      F1 : 0.5707
##              Prevalence : 0.2150
##          Detection Rate : 0.1345
##    Detection Prevalence : 0.2563
##       Balanced Accuracy : 0.7352
##
##        'Positive' Class : true
##
```

3. Upsampling: this is a technique used in machine learning to address imbalanced datasets. Upsampling involves increasing the number of instances in the minority class to balance it with the number of instances in the majority class by randomly duplicating instances from the minority class until the class distribution is balanced. This method may lead to overfitting since it replicates existing instances. Again, we make sure that the model is trained using a random sample which is the same size as the samples used to fit the models before.

```r
set.seed(123) # For reproducibility

# Generate up sampled dataset
up_sample <- upSample(x = data_svm, y = data_svm$Arrest)

up_sample %>% count(Arrest)
```

```
##   Arrest      n
## 1  false 205105
## 2   true 205105
```

```r
# Up sampled training data
index <- createDataPartition(up_sample$Arrest, p = nrow(train)/nrow(up_sample), list = TRUE)

train3 <- up_sample[index$Resample,] %>% select(-Class)

train3 %>% count(Arrest)
```

```
##   Arrest    n
## 1  false 7839
## 2   true 7839
```

```r
set.seed(123) # For reproducibility

# Fit SVM with upsampling
```

```
cl <- makePSOCKcluster(detectCores() - 1)

registerDoParallel(cl)

svm.model3 <- train(Arrest ~ ., data = train3, trControl = control, method = 'svmRadial', family = binor
```

```
## Warning in .local(x, ...): Variable(s) ‘’ constant. Cannot scale data.

## Warning in .local(x, ...): Variable(s) ‘’ constant. Cannot scale data.
```

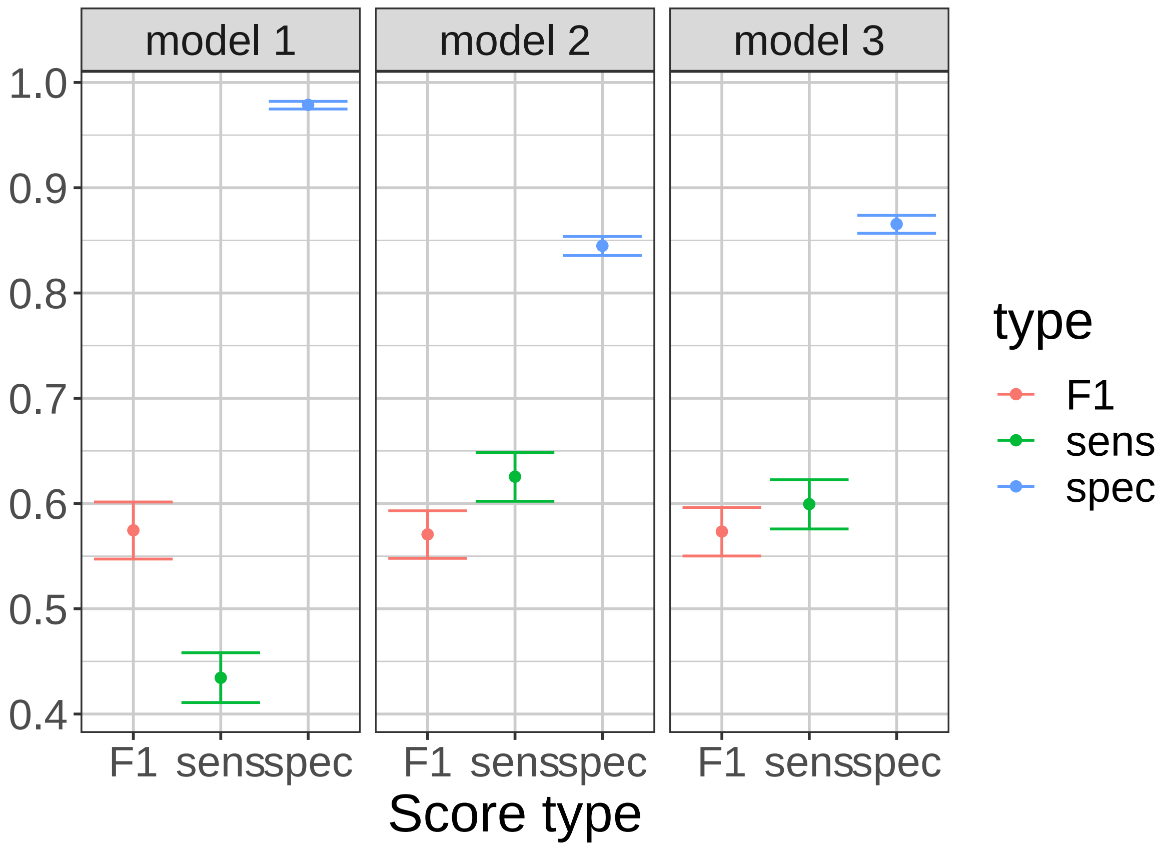
```
stopCluster(cl)

# Test data

index <- createDataPartition(data_svm$Arrest, p = .03, list = TRUE)

test <- data_svm[index$Resample,]

test$Arrest <- as.factor(test$Arrest)

# Confucion matrix
conf_matrix3 <- confusionMatrix(data = predict(svm.model3, newdata = test), reference = test$Arrest, pos

conf_matrix3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction false true
##      false  5326  675
##      true    828 1010
##
##                Accuracy : 0.8083
##                  95% CI : (0.7994, 0.8169)
##     No Information Rate : 0.785
##     P-Value [Acc > NIR] : 2.137e-07
##
##                   Kappa : 0.45
##
##  Mcnemar's Test P-Value : 8.829e-05
##
##             Sensitivity : 0.5994
##             Specificity : 0.8655
##          Pos Pred Value : 0.5495
##          Neg Pred Value : 0.8875
##               Precision : 0.5495
##                  Recall : 0.5994
##                      F1 : 0.5734
##              Prevalence : 0.2150
##          Detection Rate : 0.1288
##    Detection Prevalence : 0.2345
##       Balanced Accuracy : 0.7324
```

```
##
##        'Positive' Class : true
##
```

We want to present the results from all 3 models below. We plot the sensitivities, specificities and for a single, combined accuracy measure, the F1 scores, and their corresponding 95% confidence intervals.

```
D = conf_matrix1$table[1,1]
C = conf_matrix1$table[1,2]
B = conf_matrix1$table[2,1]
A = conf_matrix1$table[2,2]

accuracy_table <- data.frame()

accuracy_table <- rbind(accuracy_table,cbind("model 1", "sens",BinomCI(A, A+C)))
accuracy_table <- rbind(accuracy_table,cbind("model 1", "spec",BinomCI(D, B+D)))
accuracy_table <- rbind(accuracy_table,cbind("model 1", "F1",BinomCI(A,A+B/2+C/2)))

D = conf_matrix2$table[1,1]
C = conf_matrix2$table[1,2]
B = conf_matrix2$table[2,1]
A = conf_matrix2$table[2,2]

accuracy_table <- rbind(accuracy_table,cbind("model 2", "sens",BinomCI(A, A+C)))
accuracy_table <- rbind(accuracy_table,cbind("model 2", "spec",BinomCI(D, B+D)))
accuracy_table <- rbind(accuracy_table,cbind("model 2", "F1",BinomCI(A,A+B/2+C/2)))

D = conf_matrix3$table[1,1]
C = conf_matrix3$table[1,2]
B = conf_matrix3$table[2,1]
A = conf_matrix3$table[2,2]

accuracy_table <- rbind(accuracy_table,cbind("model 3", "sens",BinomCI(A, A+C)))
accuracy_table <- rbind(accuracy_table,cbind("model 3", "spec",BinomCI(D, B+D)))
accuracy_table <- rbind(accuracy_table,cbind("model 3", "F1",BinomCI(A,A+B/2+C/2)))



colnames(accuracy_table) <- c("model", "type", "est", "lwr", "upr")

accuracy_table$est <- as.numeric(accuracy_table$est)
accuracy_table$lwr <- as.numeric(accuracy_table$lwr)
accuracy_table$upr <- as.numeric(accuracy_table$upr)

plot <- accuracy_table %>% group_by(type) %>%
  ggplot(aes(x = type,y = est, color = type)) +
  geom_point() +
  geom_errorbar(aes(ymin = lwr, ymax = upr)) +
  facet_grid(cols = vars(model)) +
  ylab("") +
  xlab("Score type") +
  scale_y_continuous(n.breaks = 10) +
  theme_bw() +theme(panel.grid.major=element_line
(colour="grey80"))+theme(panel.grid.minor=element_line(colour="grey80")) + theme(text = element_text(si
plot
```



In the assessment of three SVM models on the Chicago crime dataset, models 2 and 3 are meant to deal with the challenges of unbalanced training data using downsampling and upsampling. Model 2 achieves a sensitivity of 0.60 and a specificity of 0.8622, while model 3 achieves very similar metrics with a sensitivity of 0.6012 and a specificity of 0.8637.

Conversely, model 1 demonstrates outstanding specificity at 0.9805 but at the cost of poor sensitivity (0.4273). This is due to the fact that with model 1, we used a proportional sample where the negatives outweight the positives, so in this model negative outcomes dominate. The high specificity of this model suggests that it makes very few false positive predictions, but that is because it makes few positive predictions overall.

The choice between these models depends on the cost considerations associated with false positives and false negatives. If the priority is to minimize false positives, especially when the cost is high, model 1 is a strong candidate due to its exceptional specificity. Alternatively, if capturing arrests is crucial and the cost of false negatives is significant, models 2 or 3 would be the preferred choice with their superior sensitivity.

Examining the F1 score, a metric that balances precision and recall, all three models exhibit nearly identical performance, with a score around 0.57 for all of them. However, models 2 and 3 show slightly narrower intervals for the F1 scores compared to model 1, suggesting a higher degree of confidence in the F1 scores for models 2 and 3.

In conclusion, the implemented downsampling in model 2 and upsampling in model 3 have contributed to some improvements in addressing the challenges of unbalanced training data in the dataset. While these sampling techniques have led to more balanced sensitivity and specificity, the models' overall performance remains very similar to the performance of model 1, where we used a proportional sample.

=======

Spatial Regression

Processing variable 'Date'

```
data2 <- df %>%
  select(Date, Primary.Type, Longitude, Latitude)

data2 %<>%
  filter(!Primary.Type %in% c("NON-CRIMINAL", "OTHER NARCOTIC VIOLATION", "PUBLIC INDECENCY", "HUMAN TR

data2 %<>%
  mutate(Date = as.POSIXct(Date, format = "%m/%d/%Y %I:%M:%S %p"),
         month = format(as.Date(Date, format = "%m/%d/%Y"), "%m"))

data2 %<>%
  na.omit()
```

Predict Number of Cases withing Radius of 5 km

The second objective involves forecasting the occurrence of a specific type of crime within a 5 km radius, based on a given location point and time window. The variables `Longitude` and `Latitude` are used to denote the location, encompassing crucial distance-related information. To execute this task, it is necessary to subset the entire dataset based on the relevant `month` and `Prime.Type`. Subsequently, a new variable, `crimes_within_5km`, is created by tallying the number of data points within a 5 km radius for each row within the subset.

```
calculate_crimes_within_5km <- function(data, target_month, target_crime_type) {
   # Filter data based on specific inputs
  data %<>% filter(month == target_month) %>%
    filter(Primary.Type == target_crime_type)

  for (i in 1:nrow(data)) {
  distances <- numeric(nrow(data))  # Initialize a vector to store distances for each point

  for (j in 1:nrow(data)) {
    # Calculate distances between point i and all other points
    distances[j] <- distGeo(c(data$Longitude[i], data$Latitude[i]),c(data$Longitude[j], data$Latitude[j
  }

  # Store the distances in the data frame
  data$crimes_within_5km[i] <- sum(distances <= 5000)
  }
  return(data)
}
```

Split training and test sets

Here we take a specific time window (January-2019) and a certain crime type (say `Primary.Type` = ROBBERY) into consideration. On the basis of the two given features, we filter the dataset and calculate the new variable `crimes_within_5km` as a separate column.

```r
data_Jan_Rob <- calculate_crimes_within_5km(data2, target_month = "01", target_crime_type = "ROBBERY")
```

Similar as the regression in the last section, we randomly select indices for the training set while ensuring that approximately 80% of the data is included. The remaining indices form the test set.

```r
# Split the data into training and testing sets
set.seed(42)
split_index <- sample(seq_len(nrow(data_Jan_Rob)), size = 0.8 * nrow(data_Jan_Rob))
train_data <- data_Jan_Rob[split_index, ]
test_data <- data_Jan_Rob[-split_index, ]
```

Linear Regression

Fitting Model

The input variables for our model only include the spatial information - the `Longitude` and `Latitude`. We hope to predict the crime number given a specific month and time window, here they refer to ROBBERY and January-2019 respectively. Hence the model is of the form:

$$\text{Linear Regression Model: crimes\_within\_5km} \sim \text{Longitude} + \text{Latitude}$$

```r
# Fit a linear regression model
reg_Jan_Rob <- lm(crimes_within_5km ~ Longitude + Latitude, data = train_data)
```

We run the above model on the test set and obtain the predictions.

```r
# Make predictions on the test set
pred_Jan_Rob <- predict(reg_Jan_Rob, newdata = test_data)
```

Spatial Regression Evaluation

```r
# Evaluate the model
mse <- mean((test_data$crimes_within_5km - pred_Jan_Rob)^2)
mae <- mean(abs(test_data$crimes_within_5km - pred_Jan_Rob))
rmse <- sqrt(mean((test_data$crimes_within_5km - pred_Jan_Rob)^2))
rsquared <- 1 - (sum((test_data$crimes_within_5km - pred_Jan_Rob)^2) / sum((test_data$crimes_within_5km

print(paste("Mean Squared Error:", mse))
```

```
## [1] "Mean Squared Error: 1927.53918568966"
```

```r
print(paste("Mean Absolute Error:", mae))
```

```
## [1] "Mean Absolute Error: 34.708323526401"
```

```r
print(paste("Root Mean Squared Error:", rmse))
```

```
## [1] "Root Mean Squared Error: 43.9037491074471"
```

```r
print(paste("R-squared:", rsquared))
```

```
## [1] "R-squared: 0.0651729196744408"
```

An MSE of 2235.46 indicates that, on average, the squared difference between the predicted and actual number of crimes within 5 kilometers is approximately 2235.46, which is relatively high. With an MAE of 38.04, it suggests that the model's predictions deviate by approximately 38.04 units from the actual values. The RMSE of 47.28 represents the model's predictions deviate by approximately 47.28 units from the actual values. We are expecting a lower value for all three metrics.

The value of $R^2 = 0.066$ indicates that our model explains about 6.6% of the variability in the number of points within a 5-kilometer radius for robbery incidents in January. While R^2 provides an indication of goodness-of-fit, the relatively lower value suggests that the factors not included in the model contribute to the variability.

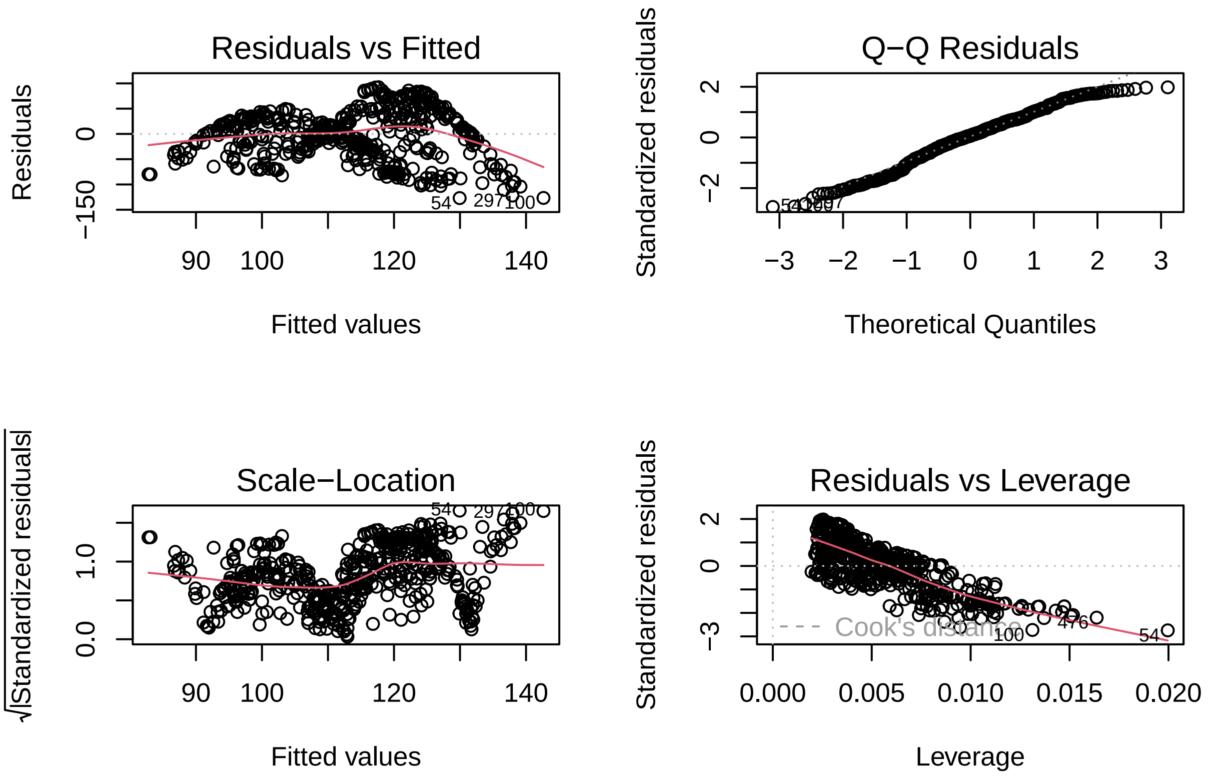
```r
# Plotting actual vs. predicted values
dev.new()
plot(test_data$crimes_within_5km, pred_Jan_Rob, main = "Actual vs. Predicted", xlab = "Actual", ylab =
abline(0, 1, col = "red")  # Adding a line of equality
```

We also generate a scatter plot, comparing the actual values to the predicted values from the linear regression model for visually assessing the predictive performance of our model and identifying potential discrepancies between observed and predicted outcomes. The position of the points is far away from the red line indicating that the model's predictions aligning with the actual values is very poor.

Diagnostics

Since the metrics and plot show a weak performance of the model, it is crucial to assess the underlying assumptions of the linear regression model. These assumptions include linearity, independence, homoscedasticity, and normality of residuals.

```r
# check assumptions
par(mfrow = c(2, 2))
plot(reg_Jan_Rob)
```



See above, the first plot shows the residuals vs fitted values, identifying patterns in residuals and suggesting that the model is approximately linear while representing heteroscedasticity. The Q-Q plot also confirms linearity, as the points fall approximately along a straight line. The Scale-Location checks for constant variance of residuals; it is expected that points should be scattered evenly with no discernible pattern, but the trend here refuses the assumption. The last one, Residuals vs. Leverage, identifies influential observations or outliers; our result figures out the existence of outliers, but it might not be the main reason for the poor performance of our model.

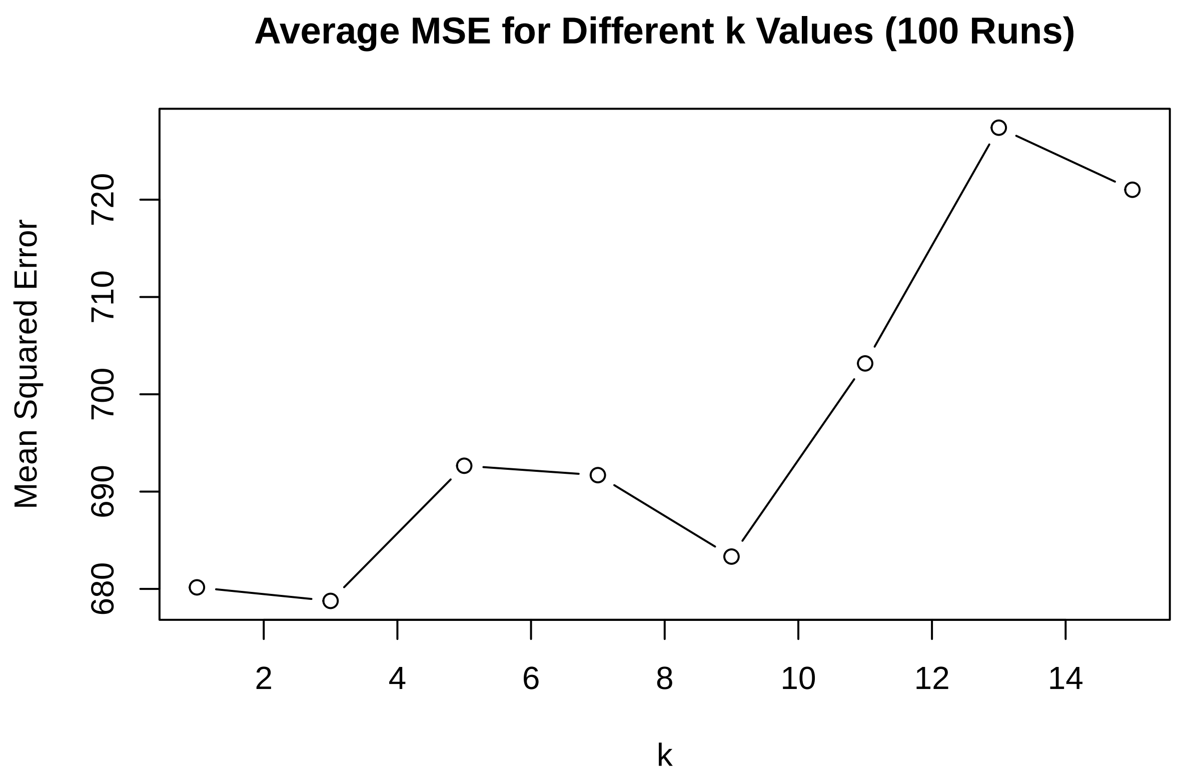
Therefore, we would pursue alternative modeling techniques later on to potentially improve the predictive performance of the crime numbers.

KNN clustering

```
set.seed(123)
# Combine longitude and latitude into matrices
train_coordinates <- cbind(train_data$Longitude, train_data$Latitude)
test_coordinates <- cbind(test_data$Longitude, test_data$Latitude)
```

```
set.seed(123)
# Set a range of k values
k_values <- c(1, 3, 5, 7, 9, 11, 13, 15)

# Initialize an empty matrix to store MSE values for each run
mse_matrix <- matrix(NA, nrow = 100, ncol = length(k_values))
```

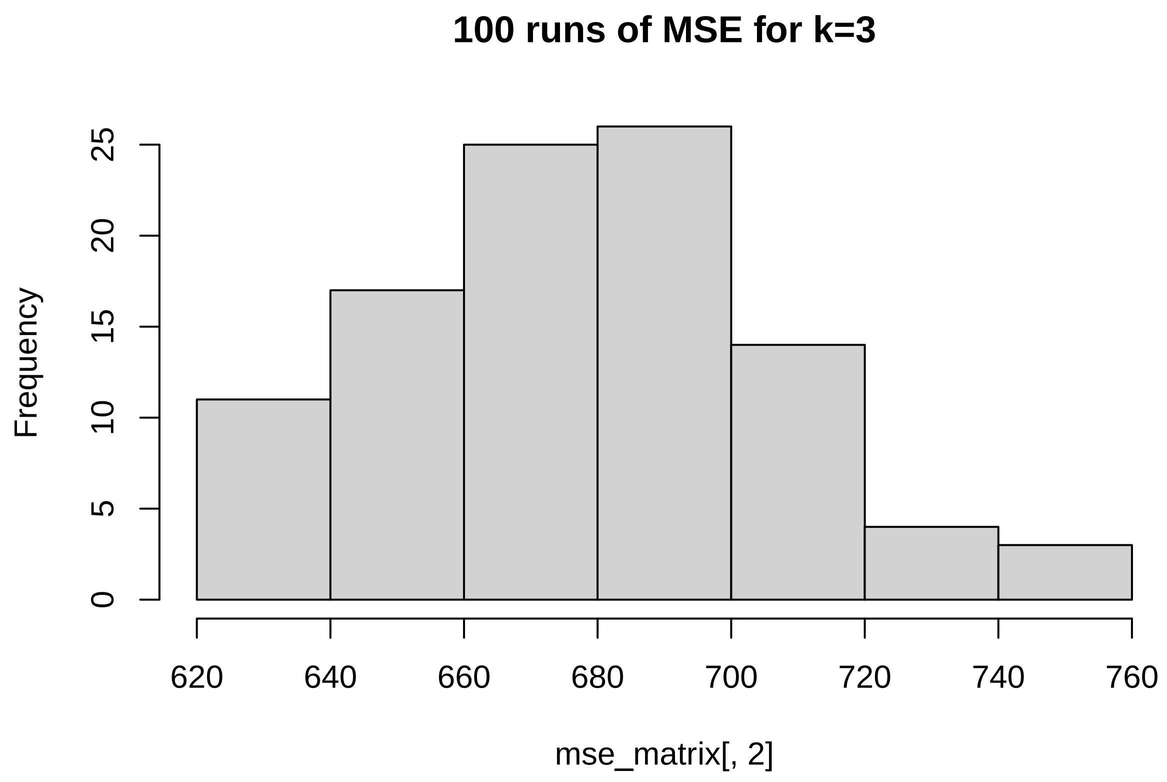
```
# Loop through different k values
for (k_index in seq_along(k_values)) {
  for (run in 1:100) {
    # Use KNN for regression to predict the number of crimes on the test set
    predicted_crimes <- class::knn(train = train_coordinates, test = test_coordinates, cl = train_data$

    # Evaluate the performance (you can use various regression metrics)
    mse_matrix[run, k_index] <- mean((as.numeric(predicted_crimes) - test_data$crimes_within_5km)^2)
  }
}

# Calculate the average MSE for each k value
average_mse <- colMeans(mse_matrix, na.rm = TRUE)

# Plot the average MSE values for different k
plot(k_values, average_mse, type = "b", main = "Average MSE for Different k Values (100 Runs)", xlab =
```



```
# Create a data frame for the results
results_df <- data.frame(k = k_values, average_mse = average_mse)

# Print the results table
print(results_df)
```

```
##    k average_mse
## 1  1    680.1591
## 2  3    678.7689
## 3  5    692.6561
## 4  7    691.6914
## 5  9    683.3217
## 6 11    703.1745
## 7 13    727.3976
## 8 15    721.0211
```

```r
hist(mse_matrix[,2], main="100 runs of MSE for k=3")
```


100 runs of MSE for k=3
Frequency
mse_matrix[, 2]