Crime Prediction Project - CSP 571

Dhruv Hiteshkumar Arora Devarsh Prashant Kale
Harsh Hiteshkumar Patel

2025-04-28

```r
library(tidyverse)
```

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

```
## -- 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(lubridate)
library(caret)
```

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

```r
library(randomForest)
```

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

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

```
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(xgboost)
```

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

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

```r
library(ggplot2)
library(e1071)
library(yardstick)
```

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

```
##
## Attaching package: 'yardstick'
##
## The following objects are masked from 'package:caret':
##
##     precision, recall, sensitivity, specificity
##
## The following object is masked from 'package:readr':
##
##     spec
```

1. INTRODUCTION

Introduction

The goal of this project is to analyze and predict arrest outcomes using crime data published by the City of Chicago. The dataset consists of several variables capturing the type of crime, time and location details, and whether the crime led to an arrest.

The primary objective is to create machine learning models that can predict the likelihood of an arrest based on available information at the time of crime reporting. This analysis involves data cleaning, exploratory data analysis (EDA), feature engineering, model training, evaluation, and result interpretation.

We aim to identify key factors influencing arrest decisions and compare model performance to evaluate which approach offers the best balance of interpretability and accuracy.

2. LOAD DATA + INITIAL VIEW

```r
# Load Cleaned Data

crime <- read_csv("../data/processed/cleaned_crime_data.csv")
```

```
## Rows: 753853 Columns: 27
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (5): Block, Primary Type, Description, Location Description, Location
## dbl  (20): Arrest, Domestic, Beat, District, Ward, Community Area, X Coordin...
## num   (1): Zip Codes
## dttm  (1): Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Overview
glimpse(crime)
```

```
## Rows: 753,853
## Columns: 27
## $ Date                        <dttm> 2022-01-01, 2022-01-01, 2022-01-01, 2022~
## $ Block                       <chr> "115XX S WESTERN AVE", "051XX W HENDERSON~
## $ `Primary Type`              <chr> "SEX OFFENSE", "SEX OFFENSE", "BATTERY", ~
## $ Description                 <chr> "CRIMINAL SEXUAL ABUSE", "OTHER", "DOMEST~
## $ `Location Description`      <chr> "ATHLETIC CLUB", "RESIDENCE", "RESIDENCE"~
## $ Arrest                      <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ Domestic                    <dbl> 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,~
## $ Beat                        <dbl> 2212, 1634, 414, 1832, 834, 711, 1834, 12~
## $ District                    <dbl> 22, 16, 4, 18, 8, 7, 18, 12, 18, 11, 22, ~
## $ Ward                        <dbl> 19, 31, 8, 42, 18, 20, 42, 11, 43, 28, 34~
## $ `Community Area`            <dbl> 75, 15, 46, 8, 70, 68, 8, 31, 7, 27, 75, ~
## $ `X Coordinate`              <dbl> 1162461, 1141493, 1192921, 1175825, 11550~
## $ `Y Coordinate`              <dbl> 1828125, 1921787, 1852149, 1904582, 18486~
## $ Year                        <dbl> 2022, 2022, 2022, 2022, 2022, 2022, 2022,~
## $ Latitude                    <dbl> 41.68402, 41.94146, 41.74926, 41.89354, 4~
## $ Longitude                   <dbl> -87.68091, -87.75537, -87.56863, -87.6297~
## $ Location                    <chr> "(41.684023827, -87.680913765)", "(41.941~
## $ `Historical Wards 2003-2015` <dbl> 33, 25, 9, 22, 6, 53, 22, 8, 51, 11, 33, ~
## $ `Zip Codes`                 <dbl> 22212, 22618, 21202, 4446, 4300, 21559, 2~
## $ `Community Areas`           <dbl> 74, 15, 42, 37, 69, 66, 37, 33, 68, 28, 7~
## $ `Census Tracts`             <dbl> 379, 114, 507, 669, 198, 166, 158, 249, 7~
## $ Wards                       <dbl> 42, 17, 35, 36, 30, 4, 36, 48, 34, 23, 22~
## $ `Boundaries - ZIP Codes`    <dbl> 33, 21, 25, 55, 8, 11, 6, 40, 16, 28, 13,~
## $ `Police Districts`          <dbl> 9, 12, 19, 14, 13, 17, 14, 15, 14, 16, 9,~
## $ `Police Beats`              <dbl> 257, 26, 222, 72, 232, 135, 74, 158, 149,~
## $ Month                       <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ Hour                        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
```

```r
summary(crime)
```

```
##       Date                           Block          Primary Type
```

```
##  Min.   :2022-01-01 00:00:00.00   Length:753853      Length:753853
##  1st Qu.:2022-10-23 20:55:00.00   Class :character   Class :character
##  Median :2023-07-22 02:56:00.00   Mode  :character   Mode  :character
##  Mean   :2023-07-17 17:56:52.88
##  3rd Qu.:2024-04-14 09:00:00.00
##  Max.   :2024-12-31 23:58:00.00
##
##  Description        Location Description     Arrest          Domestic
##  Length:753853      Length:753853        Min.   :0.0000   Min.   :0.0000
##  Class :character   Class :character     1st Qu.:0.0000   1st Qu.:0.0000
##  Mode  :character   Mode  :character     Median :0.0000   Median :0.0000
##                                          Mean   :0.1251   Mean   :0.1839
##                                          3rd Qu.:0.0000   3rd Qu.:0.0000
##                                          Max.   :1.0000   Max.   :1.0000
##
##       Beat         District          Ward       Community Area
##  Min.   : 111   Min.   : 1.00   Min.   : 1.00   Min.   : 1.00
##  1st Qu.: 533   1st Qu.: 5.00   1st Qu.:10.00   1st Qu.:22.00
##  Median :1033   Median :10.00   Median :24.00   Median :32.00
##  Mean   :1155   Mean   :11.32   Mean   :23.21   Mean   :36.26
##  3rd Qu.:1732   3rd Qu.:17.00   3rd Qu.:34.00   3rd Qu.:53.00
##  Max.   :2535   Max.   :31.00   Max.   :50.00   Max.   :77.00
##                                 NA's   :12      NA's   :2
##   X Coordinate      Y Coordinate         Year         Latitude
##  Min.   :      0   Min.   :      0   Min.   :2022   Min.   :36.62
##  1st Qu.:1153974   1st Qu.:1859959   1st Qu.:2022   1st Qu.:41.77
##  Median :1167153   Median :1893748   Median :2023   Median :41.86
##  Mean   :1165329   Mean   :1887390   Mean   :2023   Mean   :41.85
##  3rd Qu.:1176766   3rd Qu.:1910292   3rd Qu.:2024   3rd Qu.:41.91
##  Max.   :1205119   Max.   :1951506   Max.   :2024   Max.   :42.02
##
##    Longitude       Location        Historical Wards 2003-2015   Zip Codes
##  Min.   :-91.69   Length:753853      Min.   : 1.00             Min.   : 2733
##  1st Qu.:-87.71   Class :character   1st Qu.:15.00             1st Qu.:21182
##  Median :-87.66   Mode  :character   Median :29.00             Median :21559
##  Mean   :-87.67                      Mean   :27.76             Mean   :18880
##  3rd Qu.:-87.63                      3rd Qu.:41.00             3rd Qu.:22216
##  Max.   :-87.52                      Max.   :53.00             Max.   :26912
##                                      NA's   :2923
##  Community Areas Census Tracts       Wards       Boundaries - ZIP Codes
##  Min.   : 1.00   Min.   :  1.0   Min.   : 1.00   Min.   : 1.00
##  1st Qu.:25.00   1st Qu.:165.0   1st Qu.:13.00   1st Qu.:16.00
##  Median :37.00   Median :374.0   Median :27.00   Median :30.00
##  Mean   :38.16   Mean   :376.7   Mean   :26.04   Mean   :31.97
##  3rd Qu.:57.00   3rd Qu.:577.0   3rd Qu.:37.00   3rd Qu.:52.00
##  Max.   :77.00   Max.   :801.0   Max.   :50.00   Max.   :61.00
##  NA's   :2536    NA's   :2810    NA's   :2528    NA's   :2532
##  Police Districts  Police Beats      Month             Hour
##  Min.   : 1.00    Min.   :  1    Min.   : 1.000   Min.   : 0.00
##  1st Qu.:10.00    1st Qu.: 78    1st Qu.: 4.000   1st Qu.: 8.00
##  Median :15.00    Median :146    Median : 7.000   Median :13.00
##  Mean   :14.81    Mean   :146    Mean   : 6.656   Mean   :12.46
##  3rd Qu.:20.00    3rd Qu.:219    3rd Qu.:10.000   3rd Qu.:18.00
##  Max.   :25.00    Max.   :277    Max.   :12.000   Max.   :23.00
```

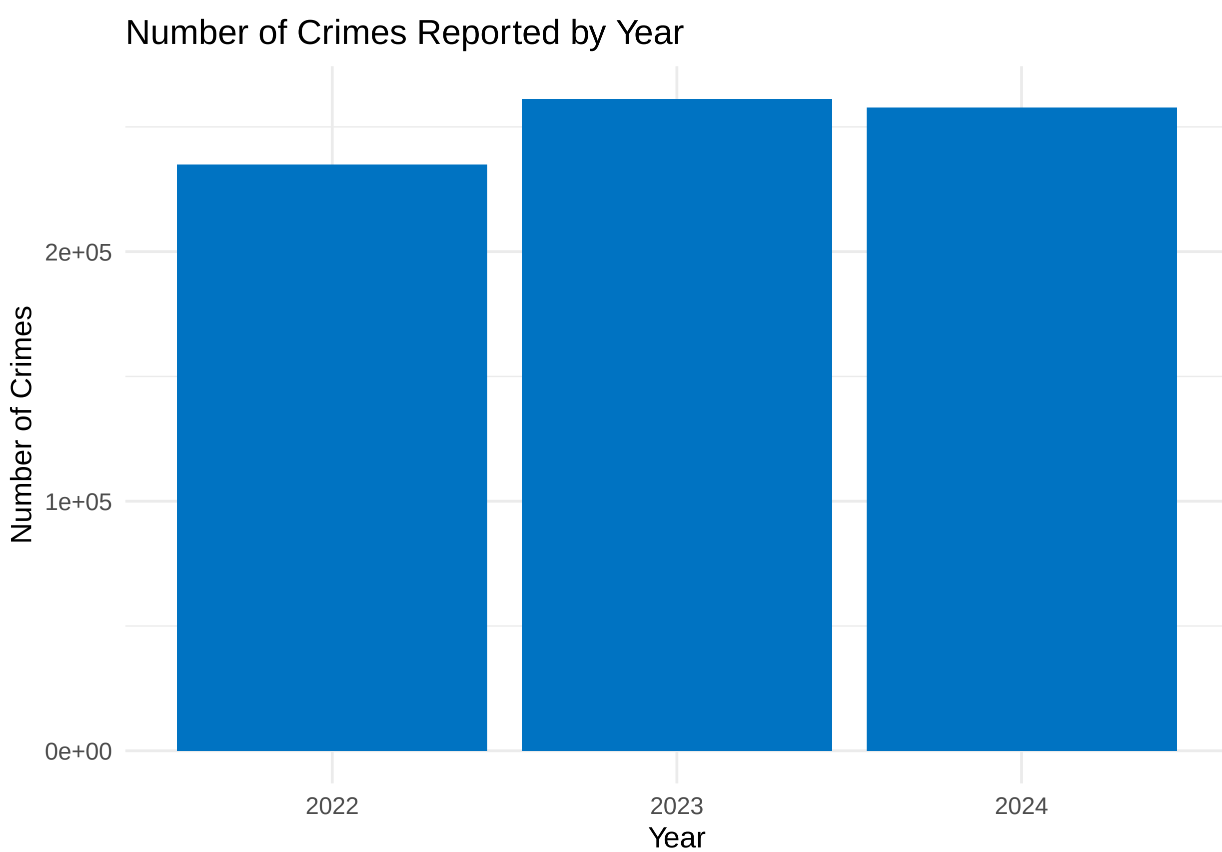
```
## NA's   :2477     NA's   :2474
```

3. EXPLORATORY DATA ANALYSIS – TIME-BASED TRENDS

In this section, we explore how crime frequency varies over time. We examine trends by **year**, **month**, and **hour of the day** to identify when crimes are most likely to occur.

3.1 Crimes per Year This plot shows the number of crimes reported in each year from 2022 to 2024. This helps us identify any rising or falling trends over time.

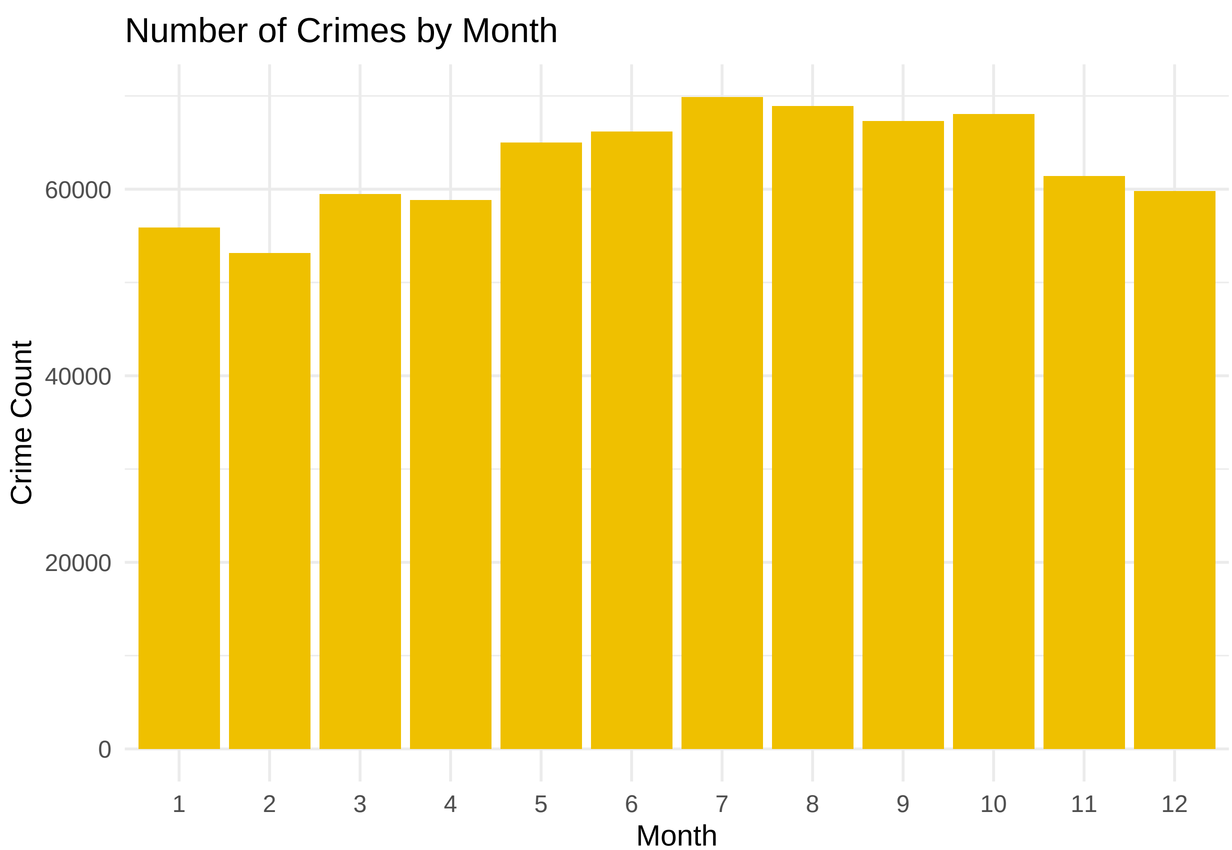
```
crime %>%
  count(Year) %>%
  ggplot(aes(x = factor(Year), y = n)) +
  geom_bar(stat = "identity", fill = "#0073C2FF") +
  labs(title = "Number of Crimes Reported by Year",
       x = "Year", y = "Number of Crimes") +
  theme_minimal()
```



Crime incidents slightly increased from 2022 to 2023 and remained consistent through 2024, indicating a steady high volume of criminal activity across years.

3.2 Crimes by Month Crime counts are plotted by month to understand seasonal variations. Spikes in specific months may indicate seasonal patterns or major events.

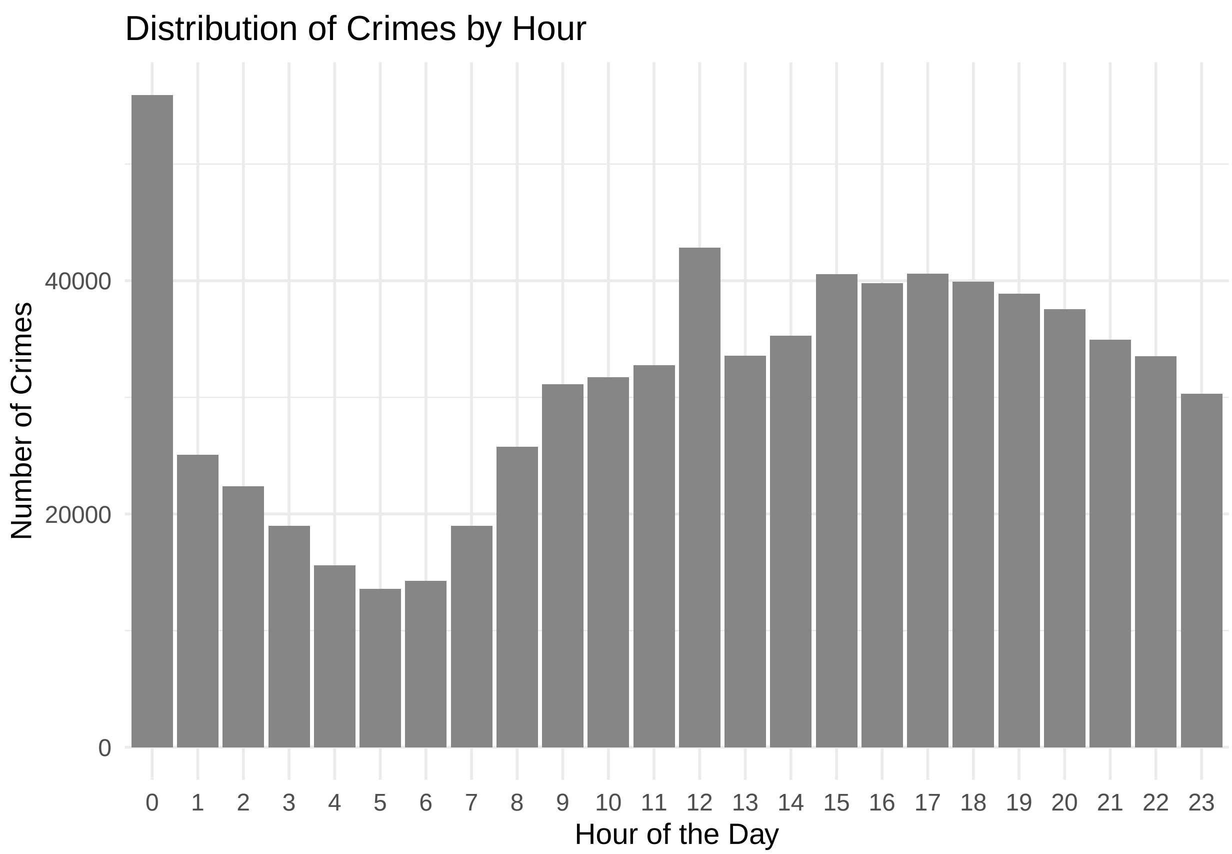
```r
crime %>%
  count(Month) %>%
  ggplot(aes(x = factor(Month), y = n)) +
  geom_bar(stat = "identity", fill = "#EFC000FF") +
  labs(title = "Number of Crimes by Month",
       x = "Month", y = "Crime Count") +
  theme_minimal()
```



July showed the highest crime rate, suggesting increased activity during summer months. Winter months (especially February) had noticeably fewer crimes.

3.3 Crimes by Hour of the Day This chart shows the distribution of crimes over the 24-hour day, helping us understand what times crimes are most frequently reported.

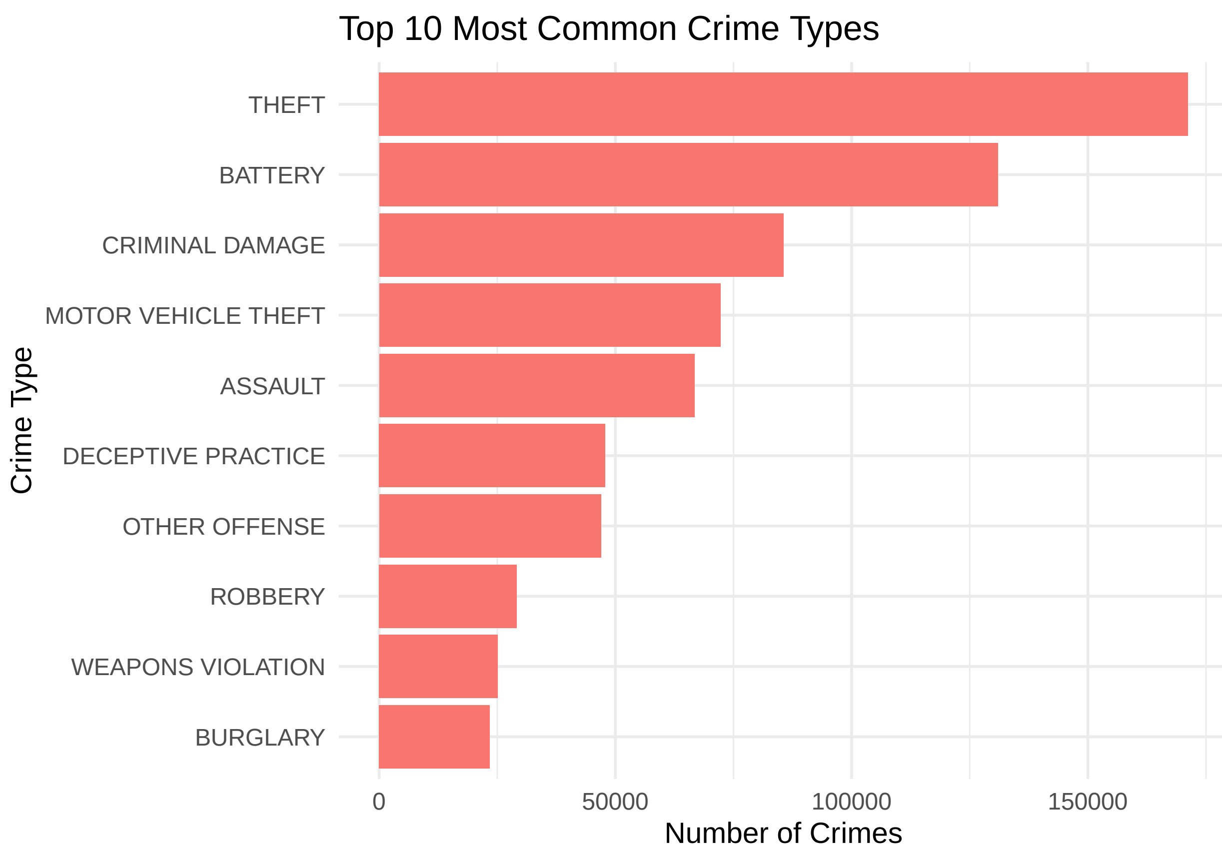
```r
crime %>%
  count(Hour) %>%
  ggplot(aes(x = factor(Hour), y = n)) +
  geom_bar(stat = "identity", fill = "#868686FF") +
  labs(title = "Distribution of Crimes by Hour",
       x = "Hour of the Day", y = "Number of Crimes") +
  theme_minimal()
```



Crimes peaked around midnight and noon. Very low activity was seen between 3–6 AM. This shows heightened activity during both early and late hours of the day.

3.4 Top 10 Crime Types To understand the most frequent types of crimes reported in Chicago, we analyze the distribution of offenses by category. This helps identify which crime types are most common and where law enforcement might need to focus resources.

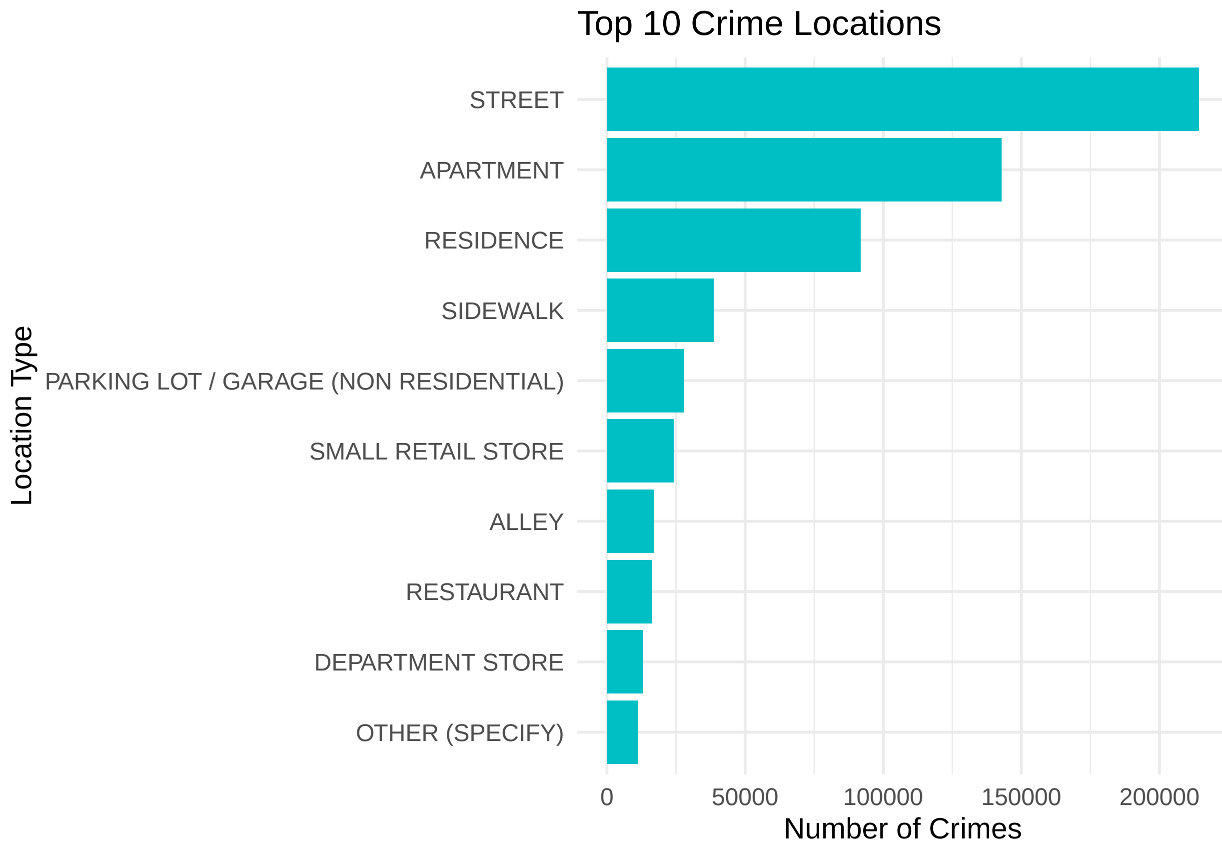
```
crime %>%
  count(`Primary Type`, sort = TRUE) %>%
  slice_max(n, n = 10) %>%
  ggplot(aes(x = reorder(`Primary Type`, n), y = n)) +
  geom_col(fill = "#F8766D") +
  coord_flip() +
  labs(title = "Top 10 Most Common Crime Types",
       x = "Crime Type", y = "Number of Crimes") +
  theme_minimal()
```



The most frequent crime was Theft, followed by Battery and Criminal Damage. These top 3 crime types together accounted for a significant portion of all reported incidents.

3.5 Top 10 Crime Locations This plot shows the top 10 most frequent locations where crimes were reported. Analyzing location types (e.g., street, residence, alley) helps reveal environmental risk factors and public safety priorities.

```r
crime %>%
  count(`Location Description`, sort = TRUE) %>%
  slice_max(n, n = 10) %>%
  ggplot(aes(x = reorder(`Location Description`, n), y = n)) +
  geom_col(fill = "#00BFC4") +
  coord_flip() +
  labs(title = "Top 10 Crime Locations",
       x = "Location Type", y = "Number of Crimes") +
  theme_minimal()
```



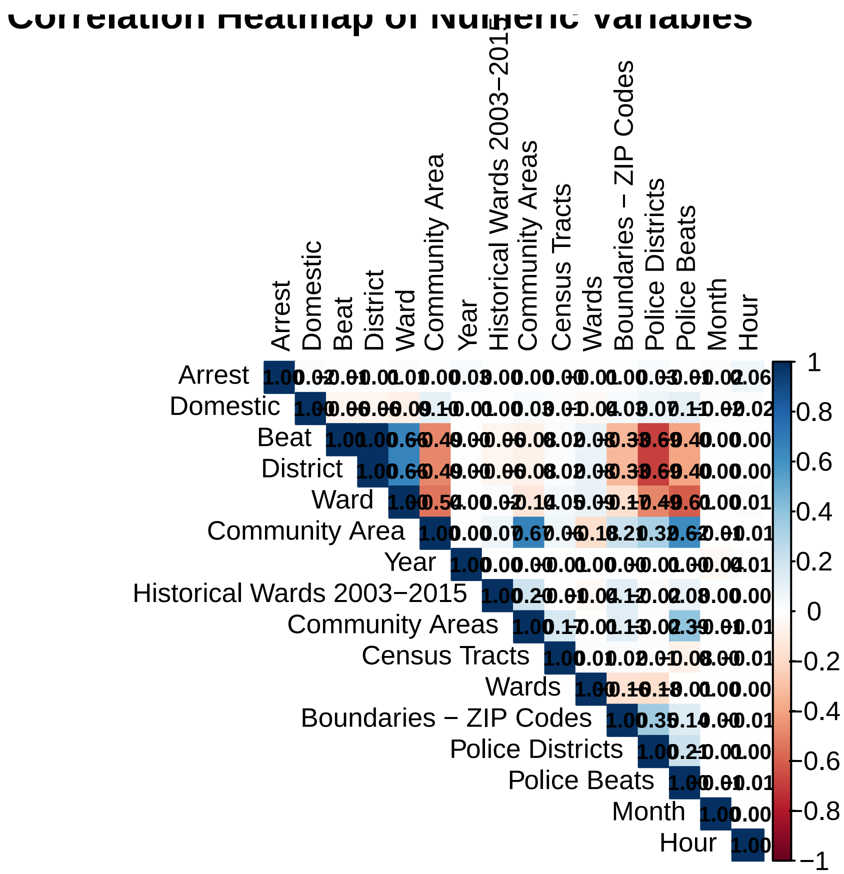
Most crimes occurred on streets, followed by apartments and residences. Public spaces and private housing were consistently high-risk zones.

3.6 Correlation Heatmap To assess how numerical features relate to each other, we use a correlation matrix. This helps us identify multicollinearity between variables and whether any predictors have strong linear relationships with the target variable (e.g., Arrest).

```
numeric_vars <- crime %>%
  select(where(is.numeric)) %>%
  select(-any_of(c("X Coordinate", "Y Coordinate", "Latitude", "Longitude", "Zip Codes")))

# Compute correlation matrix
cor_matrix <- round(cor(numeric_vars, use = "complete.obs"), 2)

# Load corrplot library and plot heatmap
library(corrplot)
```

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

## corrplot 0.95 loaded
```

```
corrplot::corrplot(cor_matrix, method = "color",
                   type = "upper", tl.col = "black", tl.cex = 0.8,
                   addCoef.col = "black", number.cex = 0.7,
                   title = "Correlation Heatmap of Numeric Variables")
```



No strong correlations were observed among most numeric features. District, Beat, and Ward showed moderate correlations, while Arrest and Domestic had a minor positive association.

4. Feature Engineering & Model Preparation

To train machine learning models, we need to prepare the data into a numeric format that models can interpret. This includes selecting important features, encoding categorical variables, and splitting the dataset into training and testing sets.

In this project, we focus on time-based features (`Year`, `Month`, `Hour`), binary flags (`Arrest`, `Domestic`), and categorical fields (`Primary Type`, `Location Description`). To reduce noise and complexity, we filter only the top 10 most frequent values in the categorical columns.

```
# Convert binary fields to factors
crime$Arrest <- as.factor(crime$Arrest)
crime$Domestic <- as.factor(crime$Domestic)

# Keep only top 10 crime types and top 10 locations
top_types <- names(sort(table(crime$`Primary Type`), decreasing = TRUE)[1:10])
top_locs <- names(sort(table(crime$`Location Description`), decreasing = TRUE)[1:10])

crime_filtered <- crime %>%
  filter(`Primary Type` %in% top_types,
```

```r
         `Location Description` %in% top_locs) %>%
  select(Arrest, Year, Month, Hour, Domestic, `Primary Type`, `Location Description`)

# Convert to factors
crime_filtered <- crime_filtered %>%
  mutate(across(c(Domestic, `Primary Type`, `Location Description`), as.factor))

# Create model matrix (one-hot encoding)
df_model <- model.matrix(Arrest ~ . - 1, data = crime_filtered) %>% as.data.frame()

# Reattach target variable
df_model$Arrest <- crime_filtered$Arrest
```

4.1 Filter and Encode Categorical Variables

4.2 Train-Test Split We now split the dataset into 80% for training and 20% for testing. This ensures that our model evaluations are based on unseen data.

```r
set.seed(42)

split <- createDataPartition(df_model$Arrest, p = 0.8, list = FALSE)
train_data <- df_model[split, ]
test_data <- df_model[-split, ]

# Confirm dimensions
cat("Training Size:", nrow(train_data), " | Test Size:", nrow(test_data))
```

```
## Training Size: 446276  | Test Size: 111567
```

5. Model Training & Evaluation

We now train and evaluate three classification models to predict whether a crime will result in an arrest: - Logistic Regression - Random Forest - XGBoost

Each model is trained on 80% of the data and evaluated on the remaining 20%. Metrics such as accuracy, precision, recall, and F1-score are used to compare their performance.

```r
# Prepare data
x_log <- train_data[, -ncol(train_data)]
y_log <- train_data$Arrest
x_test <- test_data[, -ncol(test_data)]
y_test <- test_data$Arrest

# Logistic Regression Model
log_model <- glm(y_log ~ ., data = data.frame(x_log, y_log), family = "binomial")

# Predict
log_probs <- predict(log_model, newdata = data.frame(x_test), type = "response")
log_preds <- ifelse(log_probs > 0.5, "1", "0")
```

```r
# Format for evaluation
log_preds <- factor(log_preds, levels = c("0", "1"))
y_test_fct <- factor(y_test, levels = c("0", "1"))

# Confusion Matrix
confusionMatrix(log_preds, y_test_fct, positive = "1")
```

5.1 Logistic Regression

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 99344  8562
##          1  1342  2319
##
##                Accuracy : 0.9112
##                  95% CI : (0.9095, 0.9129)
##     No Information Rate : 0.9025
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2838
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.21312
##             Specificity : 0.98667
##          Pos Pred Value : 0.63343
##          Neg Pred Value : 0.92065
##              Prevalence : 0.09753
##          Detection Rate : 0.02079
##    Detection Prevalence : 0.03281
##       Balanced Accuracy : 0.59990
##
##        'Positive' Class : 1
##
```

- Accuracy: 91.12%
- Sensitivity: 21.3%
- Precision (for arrests): 63.3%

Logistic regression showed strong performance overall, though it struggled to detect true positives (arrests). Still, it offered a solid baseline model.

```r
colnames(train_data) <- make.names(colnames(train_data))
colnames(test_data)  <- make.names(colnames(test_data))

# Prepare features and labels
```

```r
x_log <- train_data[, -ncol(train_data)]
y_log <- train_data$Arrest

x_test <- test_data[, -ncol(test_data)]
y_test <- test_data$Arrest

# Recreate y_test factor
y_test_fct <- factor(y_test, levels = c("0", "1"))

# Random Forest with clean data
rf_model <- randomForest(x = x_log, y = y_log, ntree = 100, importance = TRUE)
rf_preds <- predict(rf_model, newdata = x_test)

rf_preds <- factor(rf_preds, levels = c("0", "1"))
confusionMatrix(rf_preds, y_test_fct, positive = "1")
```

5.2 Random Forest

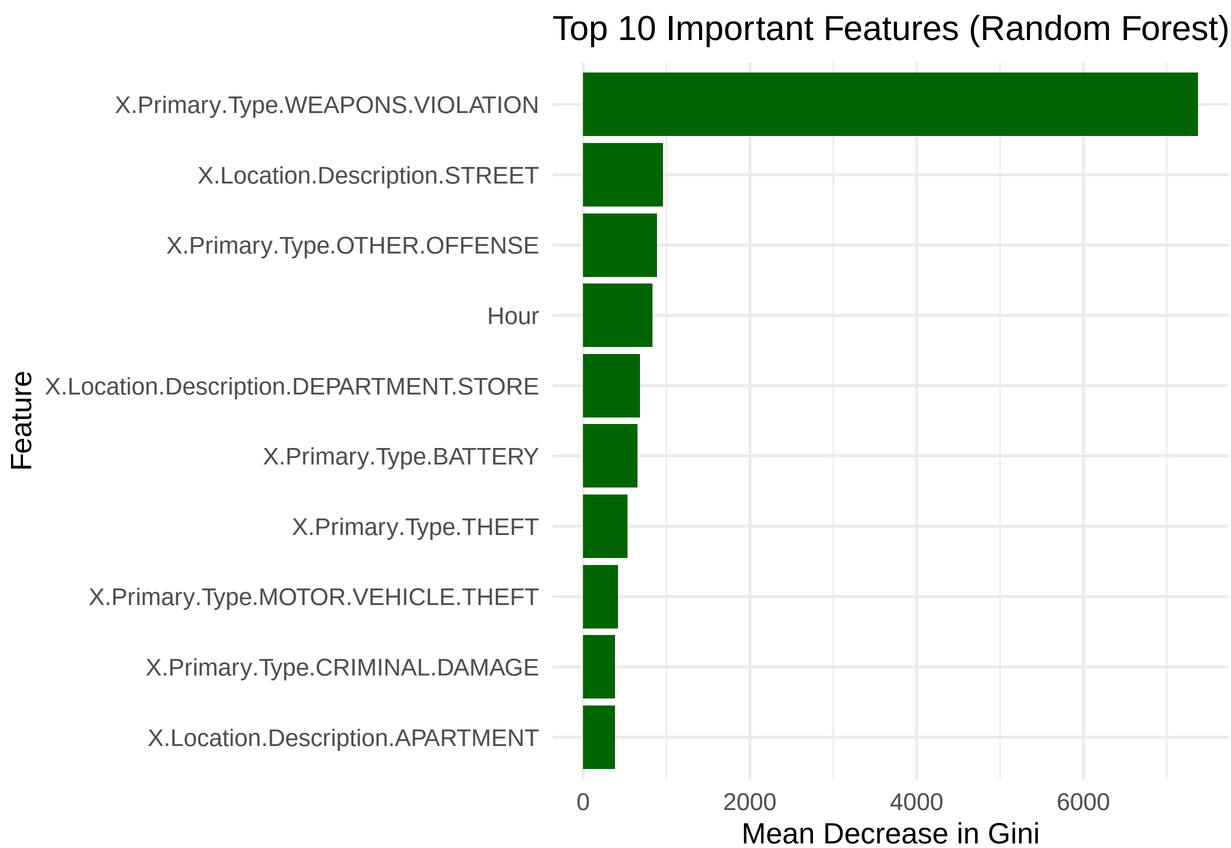
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 99957  8810
##          1   729  2071
##
##                Accuracy : 0.9145
##                  95% CI : (0.9128, 0.9161)
##     No Information Rate : 0.9025
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2738
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.19033
##             Specificity : 0.99276
##          Pos Pred Value : 0.73964
##          Neg Pred Value : 0.91900
##              Prevalence : 0.09753
##          Detection Rate : 0.01856
##    Detection Prevalence : 0.02510
##       Balanced Accuracy : 0.59155
##
##        'Positive' Class : 1
##
```

```r
# Get feature importance
imp_df <- as.data.frame(importance(rf_model))
imp_df$Feature <- rownames(imp_df)

# Sort by MeanDecreaseGini and get top 10
top10_features <- imp_df[order(-imp_df$MeanDecreaseGini), ][1:10, ]
```

```r
# Print top 10
print(top10_features)
```

```
##                                              0         1
## X.Primary.Type.WEAPONS.VIOLATION         8.6730052 32.069104
## X.Location.Description.STREET           -3.6697764 11.636277
## X.Primary.Type.OTHER.OFFENSE             5.3253194 11.753430
## Hour                                    13.5032429 14.581099
## X.Location.Description.DEPARTMENT.STORE  7.1398856  2.932150
## X.Primary.Type.BATTERY                  -6.9598717  7.977113
## X.Primary.Type.THEFT                    -5.6132671  8.251144
## X.Primary.Type.MOTOR.VEHICLE.THEFT      -1.0451285  8.286927
## X.Primary.Type.CRIMINAL.DAMAGE          -3.0488055  7.194178
## X.Location.Description.APARTMENT         0.7036483 10.834106
##                                         MeanDecreaseAccuracy MeanDecreaseGini
## X.Primary.Type.WEAPONS.VIOLATION                   24.306903        7376.5655
## X.Location.Description.STREET                      11.820376         953.9079
## X.Primary.Type.OTHER.OFFENSE                        9.273201         885.5834
## Hour                                               15.614254         828.5545
## X.Location.Description.DEPARTMENT.STORE             9.218337         678.3752
## X.Primary.Type.BATTERY                              8.031253         649.8165
## X.Primary.Type.THEFT                                7.309065         529.4063
## X.Primary.Type.MOTOR.VEHICLE.THEFT                  5.640898         417.4427
## X.Primary.Type.CRIMINAL.DAMAGE                      5.904193         378.8028
## X.Location.Description.APARTMENT                   12.049579         378.4516
##                                                                         Feature
## X.Primary.Type.WEAPONS.VIOLATION               X.Primary.Type.WEAPONS.VIOLATION
## X.Location.Description.STREET                     X.Location.Description.STREET
## X.Primary.Type.OTHER.OFFENSE                       X.Primary.Type.OTHER.OFFENSE
## Hour                                                                       Hour
## X.Location.Description.DEPARTMENT.STORE X.Location.Description.DEPARTMENT.STORE
## X.Primary.Type.BATTERY                                   X.Primary.Type.BATTERY
## X.Primary.Type.THEFT                                       X.Primary.Type.THEFT
## X.Primary.Type.MOTOR.VEHICLE.THEFT           X.Primary.Type.MOTOR.VEHICLE.THEFT
## X.Primary.Type.CRIMINAL.DAMAGE                   X.Primary.Type.CRIMINAL.DAMAGE
## X.Location.Description.APARTMENT               X.Location.Description.APARTMENT
```

```r
# Plot top 10 important features
library(ggplot2)

ggplot(top10_features, aes(x = reorder(Feature, MeanDecreaseGini), y = MeanDecreaseGini)) +
  geom_col(fill = "darkgreen") +
  coord_flip() +
  labs(title = "Top 10 Important Features (Random Forest)",
       x = "Feature", y = "Mean Decrease in Gini") +
  theme_minimal()
```



- Accuracy: 91.45%
- Sensitivity: 19.0%
- Precision (for arrests): 73.9%

Random forest performed well in terms of accuracy and precision but had the lowest sensitivity of the three, indicating class imbalance challenges.

5.3 XGBoost XGBoost is an optimized gradient boosting algorithm. It's often very effective for structured/tabular data. We convert our data to matrix format using `xgb.DMatrix()` and predict the probability of arrest.

```
# Convert x and y to XGBoost format
xgb_train <- xgb.DMatrix(data = as.matrix(x_log), label = as.numeric(y_log) - 1)
xgb_test  <- xgb.DMatrix(data = as.matrix(x_test))

# Train XGBoost model
xgb_model <- xgboost(data = xgb_train, nrounds = 50, objective = "binary:logistic", verbose = 0)

# Predict and classify
xgb_probs <- predict(xgb_model, xgb_test)
xgb_preds <- ifelse(xgb_probs > 0.5, "1", "0")

# Evaluate
```

```r
xgb_preds <- factor(xgb_preds, levels = c("0", "1"))
confusionMatrix(xgb_preds, y_test_fct, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 99459  8201
##          1  1227  2680
##
##                Accuracy : 0.9155
##                  95% CI : (0.9138, 0.9171)
##     No Information Rate : 0.9025
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3278
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.24630
##             Specificity : 0.98781
##          Pos Pred Value : 0.68595
##          Neg Pred Value : 0.92383
##              Prevalence : 0.09753
##          Detection Rate : 0.02402
##    Detection Prevalence : 0.03502
##       Balanced Accuracy : 0.61706
##
##        'Positive' Class : 1
##
```

- Accuracy: 91.55%
- Sensitivity: 24.6%
- Precision (for arrests): 68.6%

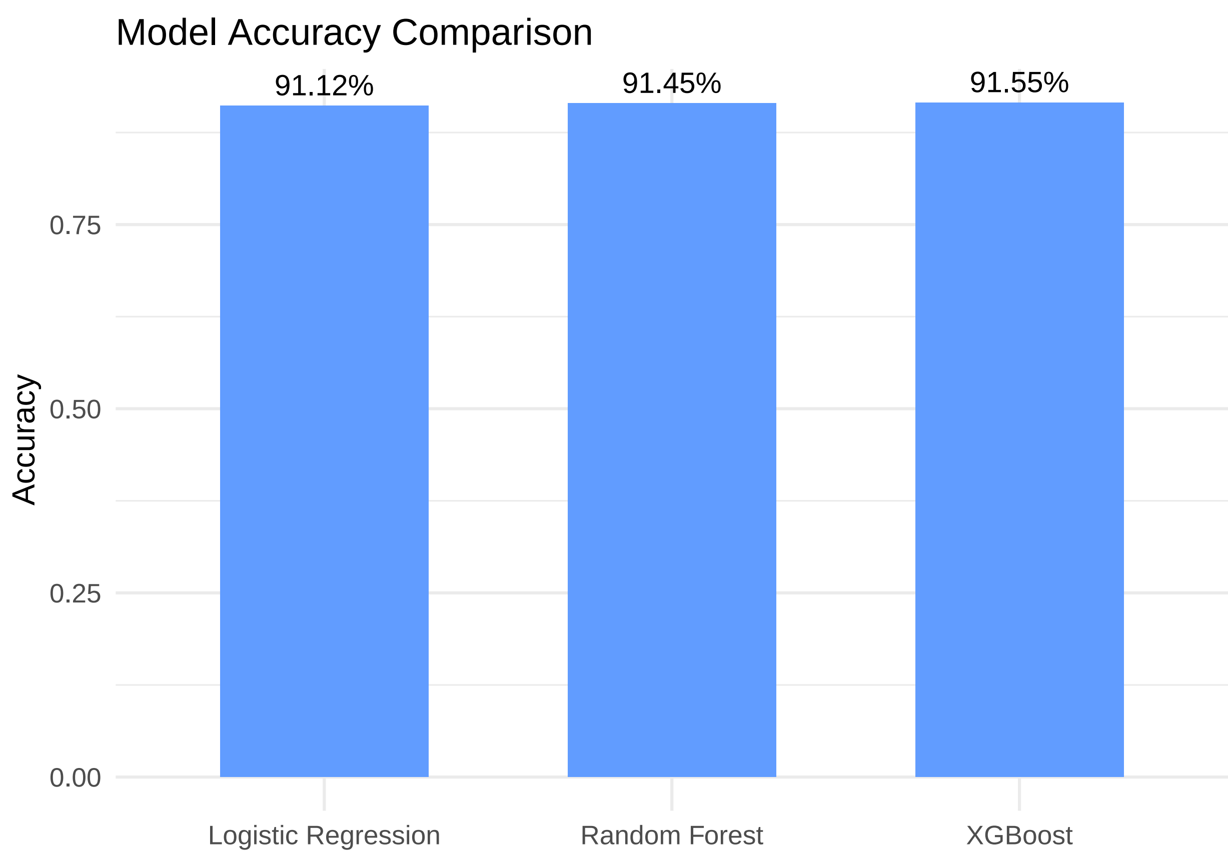
XGBoost achieved the highest accuracy and best balance between sensitivity and specificity, making it the most reliable model among the three.

5.4 Model Accuracy Comparison The following chart compares model accuracy based on our test dataset results:

- Logistic Regression: 91.12%
- Random Forest: 91.45%
- XGBoost: 91.55%

```r
acc_log <- 0.9112
acc_rf  <- 0.9145
acc_xgb <- 0.9155

model_acc <- tibble(
```

```r
  Model = c("Logistic Regression", "Random Forest", "XGBoost"),
  Accuracy = c(acc_log, acc_rf, acc_xgb)
)

ggplot(model_acc, aes(x = Model, y = Accuracy)) +
  geom_col(fill = "#619CFF", width = 0.6) +
  geom_text(aes(label = paste0(round(Accuracy * 100, 2), "%")),
            vjust = -0.5, size = 4) +
  labs(title = "Model Accuracy Comparison",
       y = "Accuracy", x = NULL) +
  theme_minimal(base_size = 12)
```



6. Conclusion & Recommendations

This project aimed to build a predictive model for determining whether a crime incident in Chicago would lead to an arrest, using open public data. Through extensive data cleaning, exploratory analysis, and supervised learning techniques, we developed and evaluated three classification models: Logistic Regression, Random Forest, and XGBoost.

Key Findings:

- **XGBoost** delivered the best accuracy (91.55%) and most balanced sensitivity (24.6%), indicating it performed best at predicting true arrest outcomes.
- **Random Forest** had slightly lower sensitivity but higher precision (73.9%), meaning it made fewer false arrest predictions.

- **Logistic Regression** offered interpretability and solid performance with 91.12% accuracy, making it a strong baseline model.

EDA Insights:

- Midnight and noon hours had the highest number of reported crimes.
- Theft, Battery, and Criminal Damage were the most frequent crime types.
- Most crimes occurred on streets, followed by apartments and residences.
- Time-based features and crime categories contributed significantly to model performance.

Recommendations:

- Handle class imbalance using techniques like SMOTE or oversampling.
- Introduce geographic clustering (e.g., crime hotspots).
- Incorporate external data like socio-economic indicators, weather, and events.
- Deploy interactive dashboards using Power BI or Tableau for easier insights.

This analysis demonstrates how publicly available data can be transformed into actionable insights for public safety and operational efficiency.

7. References

1. City of Chicago. (2024). *Crimes - 2001 to Present* [Dataset]. Retrieved from: https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-Present/ijzp-q8t2

2. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning with Applications in R* (2nd ed.). Springer. ISBN: 978-1-0716-1418-1

3. Chen, T., & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System*. Proceedings of the 22nd ACM SIGKDD Conference, 785–794. https://doi.org/10.1145/2939672.2939785

4. Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling*. Springer. https://doi.org/10.1007/978-1-4614-6849-3

5. Breiman, L. (2001). *Random Forests*. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

6. Hothorn, T., Hornik, K., & Zeileis, A. (2006). *Unbiased Recursive Partitioning: A Conditional Inference Framework*. Journal of Computational and Graphical Statistics, 15(3), 651–674. https://doi.org/10.1198/106186006X133933

7. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). *SMOTE: Synthetic Minority Over-sampling Technique*. Journal of Artificial Intelligence Research, 16, 321–357. https://doi.org/10.1613/jair.953