Predicting Violent Crimes in San Francisco

Rushil Nakkana

Spring 2023

Abstract

According to the California Department of Corrections and Rehabilitation, violent crimes in California can be defined as one of the following 5 categories: Assault, Burglary, Robbery, Domestic Violence, and Weapon Laws. Police dispatchers receive 240 million calls per year, amounting to 7.6 calls ever second. Since a majority of these calls have nothing to do with crime or violence, I thought it would be beneficial to be able to predict violent crimes to provide law enforcement with the means to make safer, data-supported decisions. I attempted to predict violent crimes using a variety of models that were trained and tested on the same data set. My data was collected from DataSF, which is a site containing hundreds of data sets from the city of San Francisco. By navigating to the 'Public Safety' tab, I was able to find a crime data set that seemed to contain the most predictive variables for violent crimes.

Background Info

San Francisco, California has long struggled with crime and safety concerns. In recent years, the city has seen an increase in violent crime, particularly in neighborhoods with high poverty rates. In order to address this issue, it is essential that law enforcement agencies use data to make more informed decisions about where to allocate resources and how to prevent crime. Without a data-driven approach, efforts to reduce crime can be misguided and ineffective. By utilizing analytics and other tools to identify patterns and trends, police departments can better understand the root causes of crime and develop strategies to address them. This can lead to safer communities and more effective law enforcement practices.

Anne Milgram, a criminal justice reformer committed to using data and analytics to fight crime, discussed the need to use data to make safer decisions in a TED Talk. She recounts spending a day in the Camden, New Jersey police department, which was known as the most dangerous city in America at the time. While there, she observed senior police officials brainstorming ways to reduce crime, but noticed that their approach was limited to using yellow sticky notes to track unsolved crimes. This lack of data-driven policing made Milgram realize that the department was failing to effectively combat crime. They noticed that there was no comprehensive data about criminals in the justice system, and that the department did not use analytics or tools to make better decisions.

This experience prompted me to think about how I can use the techniques and practices I learned in this class to make information from data to allow law enforcement to make safer decisions.

Data Collection / Cleaning

As mentioned earlier, my data was collected from DataSF. It is a data set from 2016 consisting of every police incident that occurred in San Francisco that year. Prior to cleaning, the data set had 150500 rows of data across 13 variables.

My data cleaning process consisted of six steps:

1) Removed rows with missing values (there was only one NA in the Police District column)
2) Created a new category for **Domestic Violence** because these incidents were contained in the 'Secondary Codes' category
3) Subsetted entire data set to only contain incidents where the category was one of the top 15 occurring categories
4) Factorized categorical variables (PdDistrict, DayOfWeek) and made Date column a date type
5) Removed unnecessary columns (Resolution, PdId, Location)
6) Created a target variable (violent) as a factor to show violent vs nonviolent incidents

Importing my Data

```r
crime = read.csv("crime.csv")
crime_original = read.csv("crime.csv")
# sort(table(crime$Category))

# Dealing with empty cells
crime = replace(crime, crime=='', NA)
crime = na.omit(crime)

# Adding/Creating domestic violence as a category because it is the 15th most occurring crime but
# falls under secondary codes
# Domestive Violence is also considered a violent crime, so best to make it its own category
# table(crime$Descript[crime$Category=="SECONDARY CODES"])
crime$Category[crime$Descript=="DOMESTIC VIOLENCE"] = "DOMESTIC VIOLENCE"

# Subsetting dataset to include only top 15 occurring crime categories
crime = subset(crime, Category=="LARCENY/THEFT" | Category=="NON-CRIMINAL" | Category=="ASSAULT" |
                 Category=="VANDALISM" | Category=="VEHICLE THEFT" | Category=="WARRANTS" |
                 Category=="BURGLARY" | Category=="SUSPICIOUS OCC" | Category=="MISSING PERSON" |
                 Category=="DRUG/NARCOTIC" | Category=="ROBBERY" | Category=="FRAUD" |
                 Category=="DOMESTIC VIOLENCE" | Category=="TRESPASS" | Category=="WEAPON LAWS")

categories = c("LARCENY/THEFT", 'NON-CRIMINAL', 'ASSAULT', 'VANDALISM', 'VEHCILE THEFT',
               'WARRANTS', 'BURGLRARY', 'SUSPICIOUS OCC', 'MISSING PERSON', 'DRUG/NARCOTIC',
               'ROBBERY', 'FRAUD', 'DOMESTIC VIOLENCE', 'TRESPASS', 'WEAPON LAWS')

# Cleaning other variables
# crime$Date = substring(crime$Date,1,10)
crime$PdDistrict = as.factor(crime$PdDistrict)
crime$DayOfWeek = as.factor(crime$DayOfWeek)
crime = crime[ , !names(crime) %in%
    c("Resolution","PdId", "Location")]
crime$Date = as.Date(crime$Date, "%m/%d/%Y")
crime$Hour = substring(crime$Time,1,2)
crime$Hour = as.numeric(crime$Hour)

# Adding factor variable called "violent" and setting violent crimes to 1
crime$violent = 0
crime$nonviolent = 0
crime$violent[crime$Category=="ASSAULT"] = 1
crime$violent[crime$Category=="BURGLARY"] = 1
crime$violent[crime$Category=="ROBBERY"] = 1
```

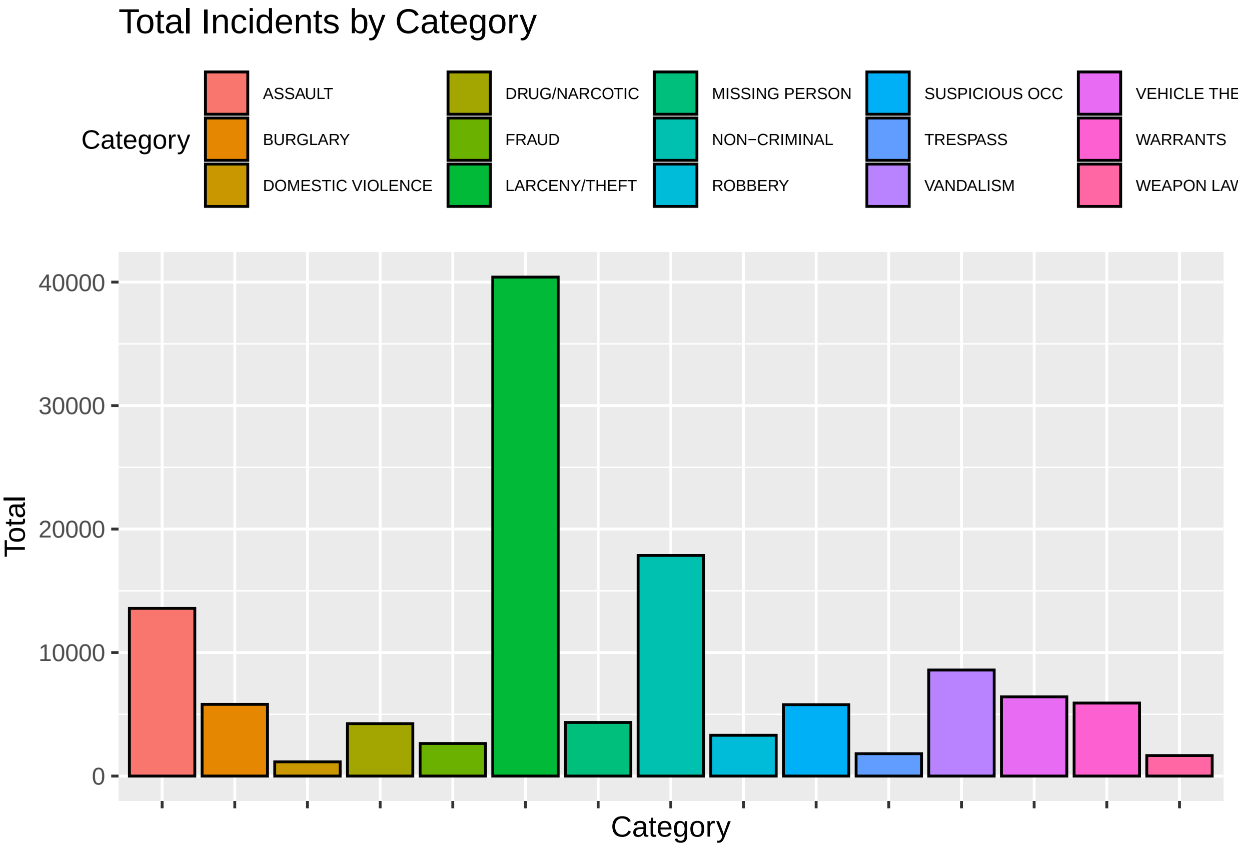
```
crime$violent[crime$Category=="DOMESTIC VIOLENCE"] = 1
crime$violent[crime$Category=="WEAPON LAWS"] = 1
crime$nonviolent[crime$violent==0] = 1
crime$violent = as.factor(crime$violent)
crime$nonviolent = as.factor(crime$nonviolent)
crime$Hour = as.integer(crime$Hour)
violent_crimes = c("ASSAULT", "BURGLARY", "ROBBERY", "DOMESTIC VIOLENCE", "WEAPON LAWS")
```

Exploratory Data Analysis

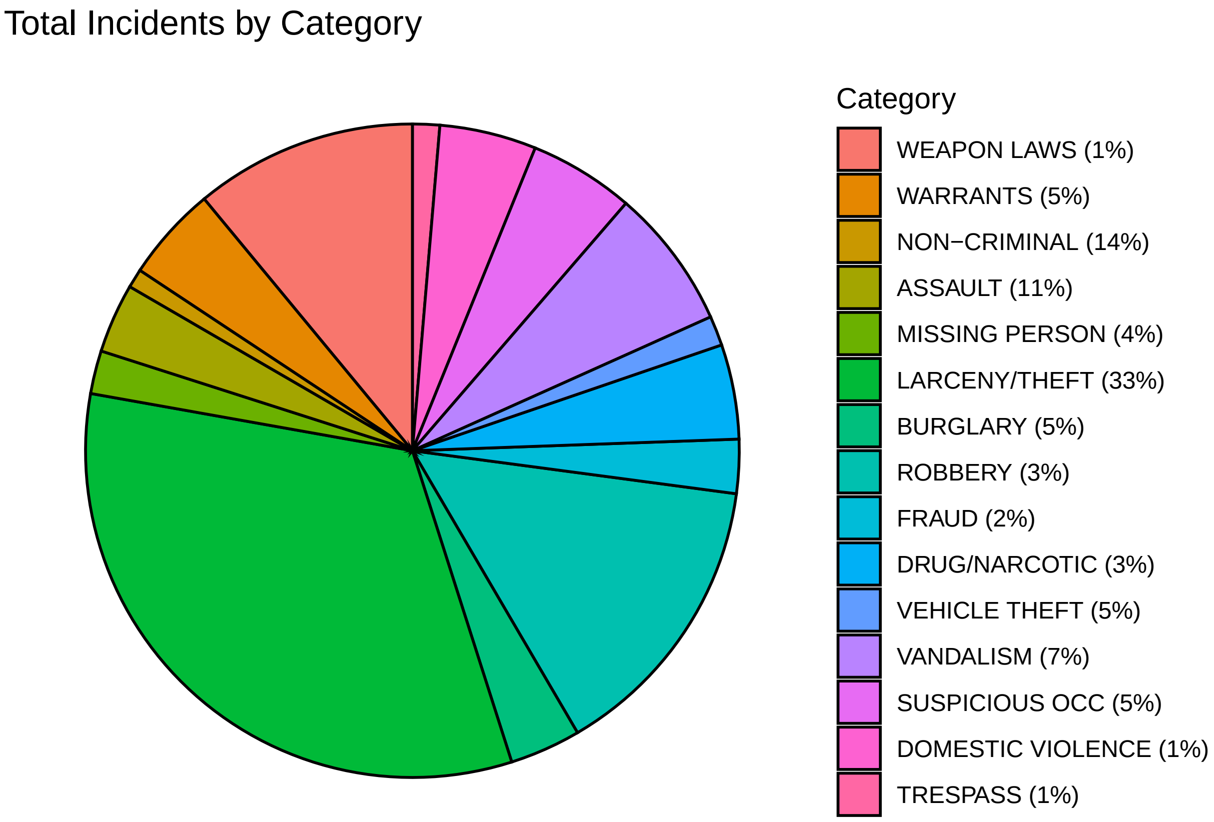
Before exploring the data set, let's understand our variables. The following table displays the columns left in the data set following the cleaning steps I previously outlined.

I then decided to investigate the categorical variables: Category, DayOfWeek, PdDistrict.

Category:

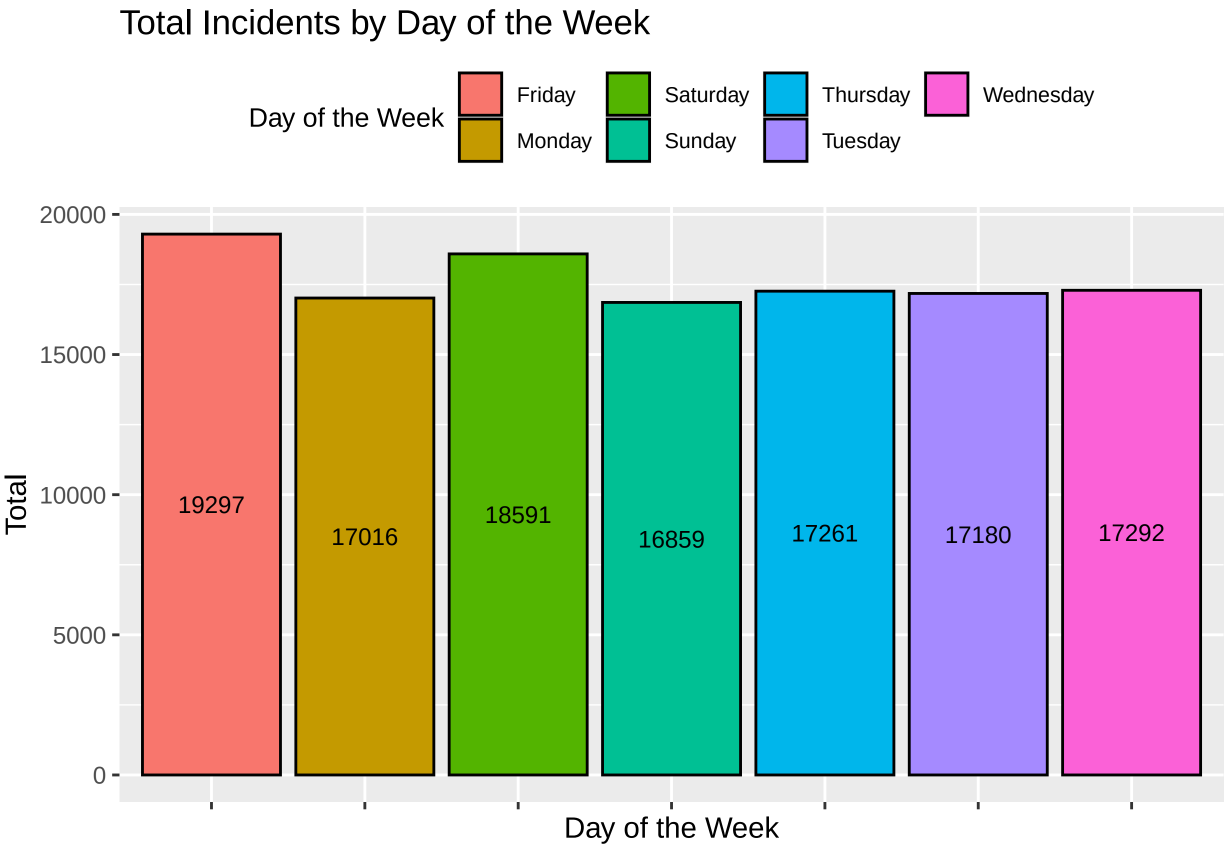


Based on the bar chart above, it is clear that there is a discrepancy across the categories of crime in terms of the number of incidents across the year. To understand the relative proportion of crimes in each category, I decided to plot the same data in a pie chart.



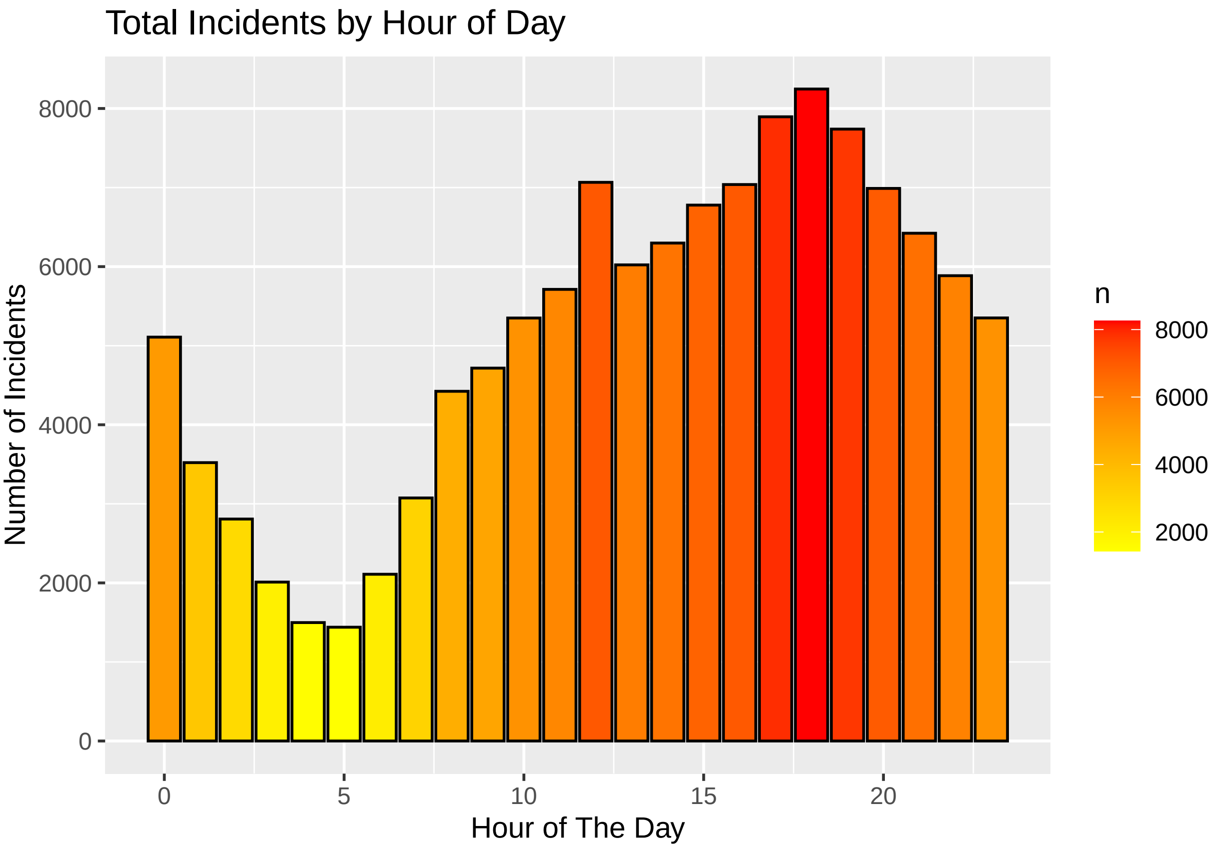
This pie chart shows the distribution of incidents by category, with each slice representing a category and its size proportional to the number of incidents. The chart reveals that Larceny/Theft is the most frequent category (33%), followed by Non-Criminal (14%), and Assault (11%). The least frequent categories were Weapon Laws, Domestic Violence, and Trespass (1%). The percentages of incidents for the remaining categories are displayed in the legend on the right side of the chart.

Day of the Week:



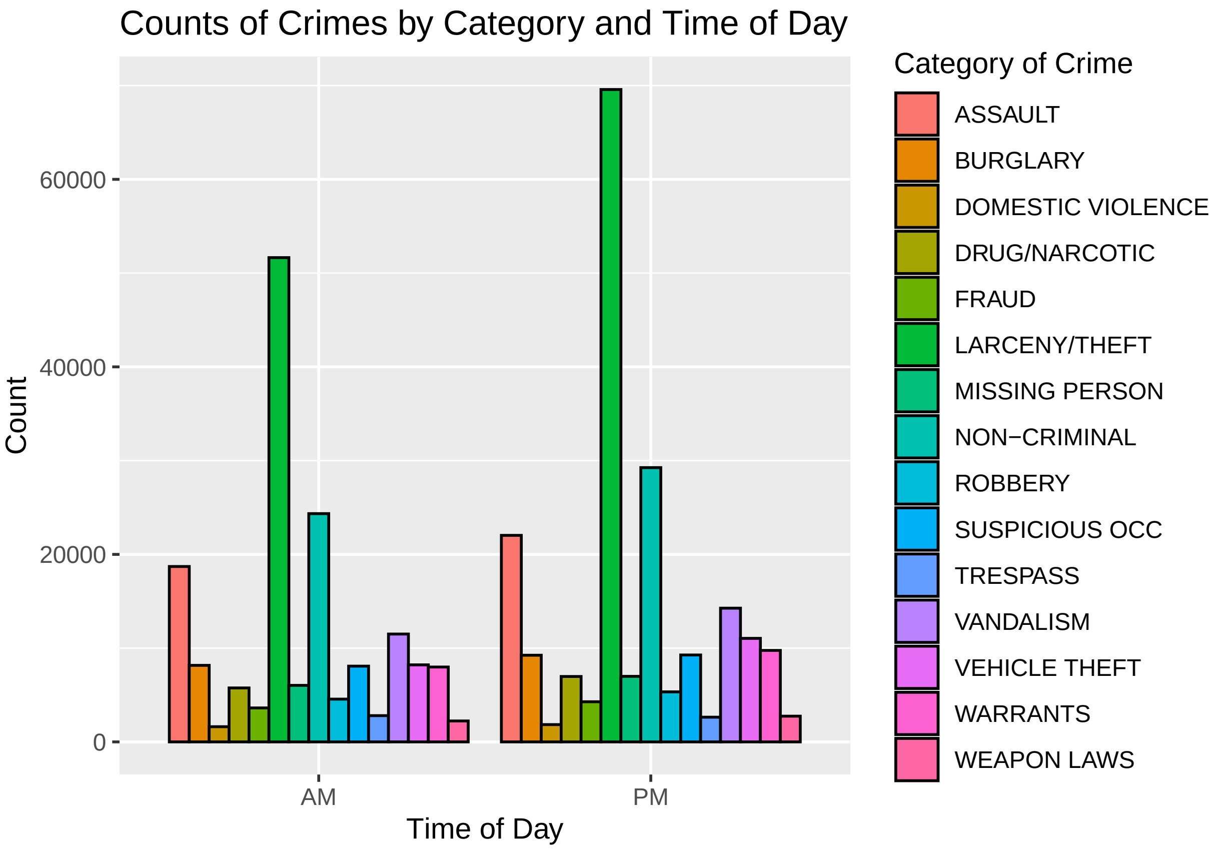
The bar chart above shows the total incidents by day of the week. The total number of incidents per day is relatively uniform, with the highest number of incidents having occurred on Friday and Saturday, while the lowest number occurred on Monday.

Time:



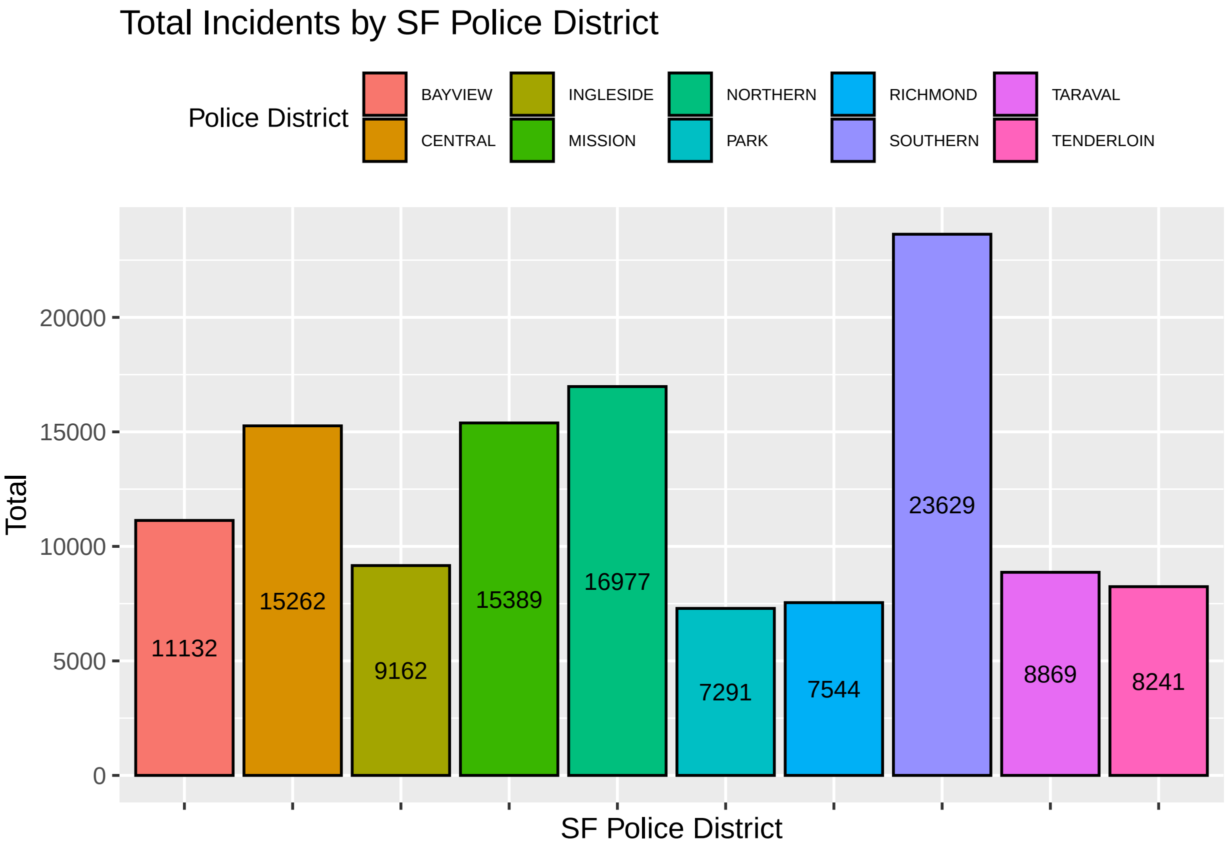
The bar chart above displays the hourly incidents of crimes, with bars colored by a gradient scale based on the number of incidents per hour. The plot shows a clear pattern of increased crimes during late afternoon and evening hours, or the PM, with the highest number of incidents occurring between 6 PM and midnight. On the other hand, the lowest number of crimes occurred in the early morning hours, or the AM, with the lowest number of incidents occurring between 2 AM and 7 am.

We can further investigate the role time plays on crime by looking at the number of incidents by category separately for AM times vs PM times.



This plot shows us that across most categories of crimes, incidents are more likely to occur in the PM hours. This is good to know, because it is an indication that the effect of the occurrence of a crime in the AM vs PM is similar across all categories.

Police District:

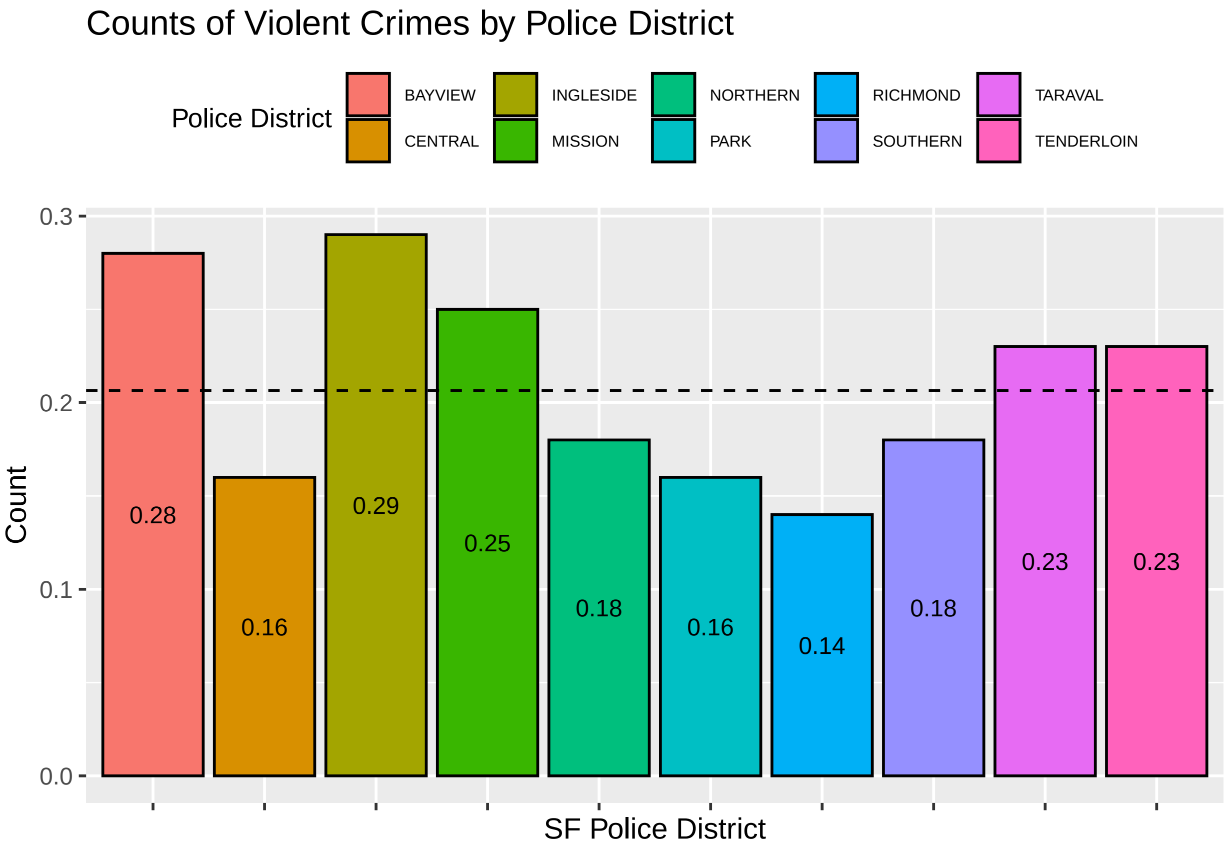


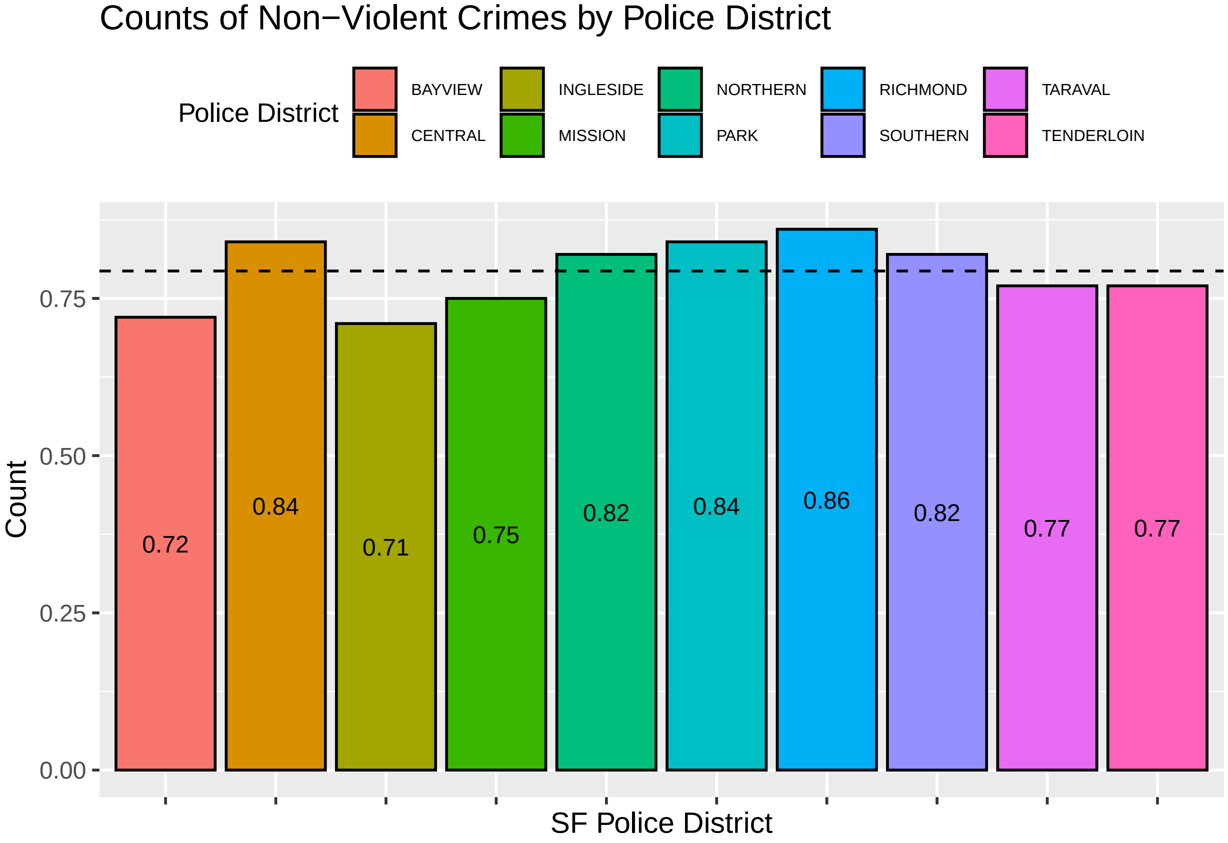
The bar chart above shows that the number of incidents per police district is inconsistent, so it still has the possibility of being an influential feature.

Target Variable Segmenting:

The final EDA step I performed before getting into the bulk of my analysis was segmenting the data based on the target outcome. I would like to revisit the PdDistrict variable separately for violent crimes (violent = 1) and those that were non-violent (violent = 0). Below is the proportion of these classes in the overall data set.

```
##
##         0         1
## 0.7935966 0.2064034
```


Counts of Violent Crimes by Police District
Police District
BAYVIEW
CENTRAL
INGLESIDE
MISSION
NORTHERN
PARK
RICHMOND
SOUTHERN
TARAVAL
TENDERLOIN
Count
0.3
0.2
0.1
0.0
0.28
0.16
0.29
0.25
0.18
0.16
0.14
0.18
0.23
0.23
SF Police District


Counts of Non−Violent Crimes by Police District
Police District
BAYVIEW
CENTRAL
INGLESIDE
MISSION
NORTHERN
PARK
RICHMOND
SOUTHERN
TARAVAL
TENDERLOIN
Count
0.75
0.50
0.25
0.00
0.72
0.84
0.71
0.75
0.82
0.84
0.86
0.82
0.77
0.77
SF Police District

Introduction: Binary Classification

Classification refers to predicting a qualitative or categorical response variable, based on a collection of predictor variables. **Binary classification** refers to a problem of this nature in which the response variable can be one of two outcomes.

In our scenario, we want to predict the class each incident of crime will belong to (violent or non-violent) based on a collection of potential predictors (day of week, date, time, police district, address, location). This will in turn provide law enforcement with the means to make data-supported decisions that will lead to increased safety for both themselves, as well as the public. To compare techniques and try to create a model with the highest predictive capabilities, I will perform binary classification using the following modelling techniques:

1. Decision Tree

2. Naive Bayes

3. Random Forest

The same set of predictors were used to construct each of these models, as shown below. Additionally, I decided to split my data set such that 80% was put into the training set which was used to induce each model. The remaining 20% was used for validation purposes as part of the testing set.

Inspecting the Data

```r
# Split into testing and training
indxTrain = createDataPartition(y = crime$violent,p = 0.8,list = FALSE)
training = crime[indxTrain,]
testing = crime[-indxTrain,]
predictors = c('DayOfWeek','Hour','Date','X','Y','PdDistrict')

# Making sure distributions of outcome variable are similar following split
prop.table(table(training$violent)) * 100
```

```
##
##       0       1
## 79.3597 20.6403
```

```r
prop.table(table(testing$violent)) * 100
```

```
##
##        0        1
## 79.35949 20.64051
```

To ensure this division represents random assignment, I decided to observe the proportions of the target variable for both sets, also shown above Since the proportions are nearly identical, we can proceed with our analysis.

Analysis: Statistical Modelling

Decision Tree

I first decided to begin with a decision tree algorithm to predict violent crimes. Decision trees are a popular machine learning algorithm used for both classification and regression tasks. One of the main advantages of using a decision tree is that they are easy to interpret and visualize, which makes them useful in explaining the logic behind a model's predictions. Decision trees can handle both categorical and numerical data, and they can also handle missing values. In addition, decision trees can be used for both binary and multiclass classification problems. Since my project is working with a combination of both categorical and numerical data to solve a binary classification problem, I thought this would be a good place to start.

```
library(caret)
set.seed(500)
x = training[, predictors]
x$violent = training$violent
dt_model = rpart(violent~., data = x, method="class")
dt_predictions = predict(dt_model, newdata = testing, type="class")
dt_confmat = confusionMatrix(dt_predictions, testing$violent)
dt_confmat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 19601  5098
##          1     0     0
##
##                Accuracy : 0.7936
##                  95% CI : (0.7885, 0.7986)
##     No Information Rate : 0.7936
##     P-Value [Acc > NIR] : 0.5037
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.7936
##          Neg Pred Value :    NaN
##              Prevalence : 0.7936
##          Detection Rate : 0.7936
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

As shown in the confusion matrix above, there is something obviously wrong with the results of the predictions made by the decision tree. It seems that my decision tree predicted "non-violent" for every incident in the test set. Before addressing why this is the case, let's address some drawbacks to decision trees. Decision trees are easily manipulated by unbalanced data sets. Thinking back to the distribution of the target variable in the overall data set, my decision tree encountered this exact problem. Since roughly 79.36% of our data consisted of non-violent incidents, it is of no surprise that a model that predicts the simple majority for every case will have an accuracy of the same value.

Rather than address the issue of working with an unbalanced data set through processes such as down sampling, I decided to proceed to the second method, Naive Bayes.

Naive Bayes

Naive Bayes is a probabilistic algorithm that works by calculating the probability of a sample belonging to a particular class based on the values of its features. One of the main advantages of Naive Bayes is that it is computationally efficient and able to work with unbalanced data sets, which is why I wanted to try this method next. Naive Bayes can also handle both categorical and continuous data, and it is not manipulated by noise and irrelevant features. However, a key assumption made by the Naive Bayes algorithm is that each feature makes an equal and independent contribution to predicting the target variable. Keeping that fact in mind, I proceeded with running the algorithm.

```r
# PdDistrict was producing errors in the model so I chose to not include it here for simplicity
predictors = c('DayOfWeek','Hour','Date','X','Y')

# Cleaning variables to allow naive bayes model to work
training$Date = as.numeric(training$Date)
testing$Date = as.numeric(testing$Date)
x = training[, predictors]
y = training$violent
set.seed(500)
nb_model = train(x,y,'nb',trControl=trainControl(method='cv',number=10))

# Making predictions and checking testing accuracy
nb_predictions = predict(nb_model, newdata = testing )
nb_confmat = confusionMatrix(nb_predictions, testing$violent)
nb_confmat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 19518  5031
##          1    83    67
##
##                Accuracy : 0.7929
##                  95% CI : (0.7878, 0.798)
##     No Information Rate : 0.7936
##     P-Value [Acc > NIR] : 0.6029
##
##                   Kappa : 0.0139
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99577
##             Specificity : 0.01314
##          Pos Pred Value : 0.79506
##          Neg Pred Value : 0.44667
##              Prevalence : 0.79359
##          Detection Rate : 0.79023
##    Detection Prevalence : 0.99393
##       Balanced Accuracy : 0.50445
##
##        ’Positive’ Class : 0
##
```

```
# Numbers from output may not match numbers in my text
```

By observing the confusion matrix above, we can see a slight improvement in the predictions from the Naive Bayes model compared to those of the Decision Tree model. While the overall accuracy, 79.37%, was negligibly higher than before, this model was able to make some ‘violent’ classifications, and correctly identify 80 violent crimes. Since there are still a lot of false negatives (5046 cases in which a violent incident occurred and the model missed it), it is clear that the model is not working as well as we would like despite the slight increase in accuracy. This poor results of this model can be attributed to the assumption mentioned before. It is difficult to believe that there is no correlation between the features included in our model, so this assumption could be a potential hindrance to our predictions.

In search of an algorithm to address the downfalls experienced with the previous two methods, I proceeded to my final method, Random Forest.

Random Forest

A disadvantage to the Decision Tree algorithm that I did not discuss is the issue of high variance. With each unique subset of values included in the training set, the algorithm has the potential to pick up on different patterns, and thus, create different trees. This leads to an infinite amount of trees that can lead to different predictions. To eliminate this problem of high variance, as well as address the other road blocks encountered with the previous modelling techniques, I decided to end with a Random Forest model.

Random Forest is a machine learning algorithm that uses an ensemble of decision trees to make predictions. One of the main advantages of Random Forest is its ability to handle high-dimensional and complex data. It can handle a large number of input variables, including both numerical and categorical variables, and can identify the most important variables for prediction. In this technique, several decision trees are fitted using a randomly sampled set of values and features, each producing a prediction. Random Forest is particularly useful for binary classification tasks, where it can accurately predict the probability of a binary outcome by averaging the predictions of each of these individual decision trees. This can make it more robust to outliers and noise in the data, and can help to reduce the variance of the prediction compared to a single decision tree.

```r
library(randomForest)
set.seed(500)
rf_model = randomForest(formula = violent~DayOfWeek+Hour+Date+X+Y+PdDistrict, data=training)
rf_predictions = predict(rf_model, newdata = testing)
rf_confmat = confusionMatrix(rf_predictions, testing$violent)
rf_confmat
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction     0     1
##          0 18725  3732
##          1   876  1366
##
##                Accuracy : 0.8134
##                  95% CI : (0.8085, 0.8183)
##     No Information Rate : 0.7936
##     P-Value [Acc > NIR] : 3.356e-15
##
##                   Kappa : 0.2816
##
##  Mcnemar’s Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9553
##             Specificity : 0.2679
##          Pos Pred Value : 0.8338
##          Neg Pred Value : 0.6093
##              Prevalence : 0.7936
##          Detection Rate : 0.7581
##    Detection Prevalence : 0.9092
##       Balanced Accuracy : 0.6116
##
##        ’Positive’ Class : 0
##
```

```
# Numbers from output may not match numbers in my text
```

The confusion matrix above shows that the predictions from this model have been the best so far. Classification accuracy increased to 81.23%, and the number of correctly predicted violent incidents increased to 1314, while significantly decreasing the number of missed violent incidents.

So far, the **random forest** model seems to have performed the best. Since classification accuracy, alone, isn’t a suggestive metric in this case due to the imbalance of our data set’s target variable, I decided to use an additional technique for model evaluation to get a better understanding of each model’s performance: precision/recall analysis.

Analysis: Model Evaluation

Precision

Precision is a performance metric used to evaluate the accuracy of a binary classification model. It is the ratio between the true positive predictions and the total number of positive predictions made by the model. Precision can be described by the following equation:

- Precision = True Positives / (True Positives + False Positives)

In other words, precision measures the proportion of positive predictions that are actually true positive predictions. It provides a measure of how well a model avoids making false positive predictions, which can be important in many applications. True Positives are the number of incidents correctly predicted as positive by the model, while False Positives are the number of incidents incorrectly predicted as positive by the model. These measures are outlined in the lens of our problem:

- True Positives: model correctly predicted a ‘violent’ incident

- False Positives: model incorrectly predicted a 'violent' incident
- True Negatives: model correctly predicted a 'non-violent' incident
- False Negatives: model missed a 'violent' incident

In our problem, a false positive could lead to unnecessary preparation or allocation of resources by law enforcement, which can be unsafe to the public. A false negative, however, could be more detrimental in that law enforcement can be responding to an incident they don't believe to be violent, when in reality it is.

Using the confusion matrices of each model, I calculated their precisions:

Decision Tree: 0 / (0 + 0) = 0
Naive Bayes: 72 / (72 + 92) = 0.4390
Random Forest: 1385 / (1385 + 962) = 0.5902

Knowing what went wrong with the decision tree model, we can ignore it for the remaining portion of my analysis. The **random forest** model had the best precision of 0.5902, meaning it had the lowest false positive rate. This means the model incorrectly predicted violent incidents less often than the naive bayes model, which had a lower precision of 0.4390. Since I previously mentioned the cost of false negative predictions can be more detrimental, let's now examine recall.

Recall

Recall is another performance metric used to evaluate the accuracy of a binary classification model. It measures the proportion of actual positive samples that are correctly identified by the model. Recall can be described by the following equation:

Recall = True Positives / (True Positives + False Negatives)

In other words, recall measures the model's ability to correctly identify positive samples, including those that are difficult to detect. It provides a measure of how well a model avoids making false negative predictions, which can also be important in many applications. True Positives are the number of samples correctly predicted as positive by the model, and False Negatives are the number of samples incorrectly predicted as negative by the model.

I previously mentioned that in the scope of our problem, a false negative prediction can be detrimental in practice. If law enforcement were to assume an incident was non-violent as predicted by the model, but instead encounter a violent incident, the will have lacked the necessary insight which is expected from the model. To avoid this circumstance, a model with higher recall can be seen as more reliable.

Using the confusion matrices of each model, here are their respective recalls:

Decision Tree: 0 / (0 + 5098) = 0
Naive Bayes: 72 / (72 + 5026) = 0.0141
Random Forest: 1385 / (1385 + 3713) = 0.2717

The **random forest** model, once again, had the best recall of 0.2670, meaning it had the lowest false negative rate. This means the model had the less amount of incorrect predictions that missed violent incidents. Since the main objective of these predictive models is to capture as many violent incidents as possible, the random forest model performed the best yet again. The low recall of 0.0102 by the naive bayes model indicates a high false negative rate, which means implementing this model in practice can be highly dangerous in practice.

Analysis: Balanced Datasets

Down Sampling

Down sampling is a technique used to balance the distribution of a target variable in a dataset by reducing the number of instances in the majority class such that the number of instances in each class is equal. In the case of this problem, down sampling would involve randomly selecting a subset of the non-violent incidents so that the number of violent and non-violent incidents are approximately equal.

```
training_downsample = downsample(training, cat_col = "violent")
table(training$violent) / nrow(training)
```

```
##
##        0        1
## 0.793597 0.206403
```

```
table(training_downsample$violent) / nrow(training_downsample)
```

```
##
##   0   1
## 0.5 0.5
```

```
testing_downsample = downsample(testing, cat_col = "violent")
table(testing$violent) / nrow(testing)
```

```
##
##         0         1
## 0.7935949 0.2064051
```

```
table(testing_downsample$violent) / nrow(testing_downsample)
```

```
##
##   0   1
## 0.5 0.5
```

Down sampling will be beneficial in this case because the class distribution is highly imbalanced as seen above. After performing this technique on the training and testing set, we can see that the proportions of the target variable are now 0.5:0.5. By down sampling, I can balance the class distribution and ensure that the models are not biased towards predicting the majority class, as seen in the previous analysis. This can lead to better model performance, especially in terms of recall, as the model will be trained on a more balanced data set and will be better equipped to handle rare events such as violent incidents.

Decision Tree

Down sampling can benefit decision trees in by reducing the effect of class imbalance, which can negatively impact the performance of decision tree models as experienced in my previous analysis. In that scenario, we had a significant class imbalance, with a majority of non-violent incidents and a minority of violent incidents. The decision tree model was biased towards predicting non-violent incidents, as they are the dominant class.

By down sampling the majority class (non-violent incidents) to have an equal number of samples as the minority class (violent incidents), we can create a balanced data set that allows the decision tree model to make more accurate predictions. This can help the model learn to identify patterns and features that are specific to the minority class, leading to better performance.

```
set.seed(500)
x = training_downsample[, predictors]
x$violent = training_downsample$violent
dt_model = rpart(violent~., data = x, method="class")
dt_predictions = predict(dt_model, newdata = testing_downsample, type="class")
dt_confmat = confusionMatrix(dt_predictions, testing_downsample$violent)
dt_confmat
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    0    1
##          0 3035 2287
##          1 2063 2811
##
##                Accuracy : 0.5734
##                  95% CI : (0.5637, 0.583)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1467
##
##  Mcnemar's Test P-Value : 0.0007219
##
##             Sensitivity : 0.5953
##             Specificity : 0.5514
##          Pos Pred Value : 0.5703
##          Neg Pred Value : 0.5767
##              Prevalence : 0.5000
##          Detection Rate : 0.2977
##    Detection Prevalence : 0.5220
##       Balanced Accuracy : 0.5734
##
##        'Positive' Class : 0
##
```

Decision Tree Metrics:
* Accuracy: 58.05%
* Precision: 61.00%
* Recall: 44.66&

Naive Bayes

Down sampling can also benefit Naive Bayes in this scenario. Since Naive Bayes assumes that the features are conditionally independent in predicting the target variable, it can be prone to overfitting when the classes are imbalanced. By down sampling the majority class, we can balance the class distribution and minimize this risk in hopes of helping the model learn underlying patterns in the data more accurately, leading to better predictions. Additionally, down sampling can reduce the bias of the model towards the majority class, which can result in a more accurate model for both the majority and minority classes.

```
set.seed(500)
predictors = c('DayOfWeek','Hour','Date','X','Y')

# Cleaning variables to allow naive bayes model to work
training$Date = as.numeric(training$Date)
testing$Date = as.numeric(testing$Date)
x = training_downsample[, predictors]
y = training_downsample$violent
set.seed(500)
nb_model = train(x,y,'nb',trControl=trainControl(method='cv',number=10))

# Making predictions and checking testing accuracy
nb_predictions = predict(nb_model, newdata = testing_downsample)
nb_confmat = confusionMatrix(nb_predictions, testing_downsample$violent)
nb_confmat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3376 2544
##          1 1722 2554
##
##                Accuracy : 0.5816
##                  95% CI : (0.572, 0.5912)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1632
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6622
##             Specificity : 0.5010
##          Pos Pred Value : 0.5703
##          Neg Pred Value : 0.5973
##              Prevalence : 0.5000
##          Detection Rate : 0.3311
##    Detection Prevalence : 0.5806
##       Balanced Accuracy : 0.5816
##
##        'Positive' Class : 0
##
```

```
# Numbers from output may not match numbers in my text
```

Naive Bayes Metrics:
* Accuracy: 58.46%
* Precision: 59.93%
* Recall: 51.08%

Random Forest

Random Forest is yet another classification algorithm that can benefit from down sampling. Since Random Forest is an ensemble method that combines multiple decision trees, it can be prone to overfitting if the dataset is imbalanced. When the classes are balanced, the individual decision trees in the forest will have a more equal representation of each class and will be less likely to overemphasize the majority class. This can lead to a more accurate overall prediction for the model.

```r
library(randomForest)
set.seed(500)
rf_model = randomForest(formula = violent~DayOfWeek+Hour+Date+X+Y+PdDistrict, data=training_downsample)
rf_predictions = predict(rf_model, newdata = testing_downsample)
rf_confmat = confusionMatrix(rf_predictions, testing_downsample$violent)
rf_confmat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3294 1915
##          1 1804 3183
##
##                Accuracy : 0.6352
##                  95% CI : (0.6258, 0.6446)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.2705
##
##  Mcnemar’s Test P-Value : 0.07127
##
##             Sensitivity : 0.6461
##             Specificity : 0.6244
##          Pos Pred Value : 0.6324
##          Neg Pred Value : 0.6383
##              Prevalence : 0.5000
##          Detection Rate : 0.3231
##    Detection Prevalence : 0.5109
##       Balanced Accuracy : 0.6352
##
##        ’Positive’ Class : 0
##
```

```r
# Numbers from output may not match numbers in my text
```

Random Forest Metrics:
* Accuracy: 62.95%
* Precision: 63.22%
* Recall: 61.93%

Conclusion

Classification Accuracy

- Decision Tree: 58.05%
- Naive Bayes: 58.46%
- Random Forest: 62.95%

While the classification accuracies for all models decreased as a result of balancing the data sets, the **random forest** model still outperformed the other techniques with an accuracy of 62.95%.

Precision

- Decision Tree: 61.00%
- Naive Bayes: 59.93%
- Random Forest: 63.33%

As a result of balancing the data sets, it is immediately evident that the performances of each model improved by the significant increase in each precision metric. Still, the **random forest** model had the highest precision of 63.33%.

Recall

- Decision Tree: 44.66%
- Naive Bayes: 51.08%
- Random Forest: 61.93%

Finally, recall improved as well for all models. There was more disparity in the recall values than precision, but the **random forest** model still had the best performance with a recall of 61.93%.

Concluding Remarks

By analyzing the results of the predictions made by each model through assessment of classification accuracy, precision, and recall, it is evident that the **random forest** model turns out to have the best predictive capabilities. Precision and recall are two important metrics used to assess the accuracy of the models when we encounter a situation when the target variable is so unbalanced. The precision metric measures how well a model avoids making false positive predictions, which can lead to unnecessary allocation of resources by law enforcement. On the other hand, recall measures how well a model avoids making false negative predictions, which can be more detrimental in practice. In this analysis, the random forest model outperformed the decision tree and naive bayes models in both precision and recall, making it the most reliable model for predicting violent incidents in San Francisco. These findings highlight the importance of using data-driven approaches in making safer decisions in law enforcement.

References

Blog, Rsquared Academy. “A Comprehensive Introduction to Handling Date & Time in R: R-Bloggers.” R, 17 Apr. 2020, https://www.r-bloggers.com/2020/04/a-comprehensive-introduction-to-handling-date-time-in-r/.

“DataSF: City and County of San Francisco.” San Francisco Data, https://data.sfgov.org/browse?category=Public%2BSafety.

“Downsampling of Rows in a Data Frame.” R, https://search.r-project.org/CRAN/refmans/groupdata2/html/downsample.html.

Kharwal, Aman. “Crime Analysis with Data Science: Aman Kharwal.” Thecleverprogrammer, 22 June 2021, https://thecleverprogrammer.com/2020/05/26/san-francisco-crime-analysis-with-data-science/#google_vignette.

Milgram, Anne. “Why Smart Statistics Are the Key to Fighting Crime.” Anne Milgram: Why Smart Statistics Are the Key to Fighting Crime | TED Talk, https://www.ted.com/talks/anne_milgram_why_smart_statistics_are_the_key_to_fighting_crime/transcript?referrer=playlist-making_sense_of_too_much_data&autoplay=true.

Prabhakaran, Selva. “Feature Selection - Ten Effective Techniques with Examples: ML+.” Machine Learning Plus, 28 Apr. 2022, https://www.machinelearningplus.com/machine-learning/feature-selection/.

“Violent Offenses Defined.” Division of Adult Parole Operations (DAPO), 3 Feb. 2020, https://www.cdcr.ca.gov/parole/violent-offenses-defined/.

“Why Smart Statistics Are the Key to Fighting Crime.” TED, TED@BCG San Francisco, https://www.ted.com/talks/anne_milgram_why_smart_statistics_are_the_key_to_fighting_crime/transcript?referrer=playlist-making_sense_of_too_much_data&autoplay=true.