# Comparing Time-Series Models to Predict Missing Person Data from San-Francisco Police Database

Jessica Brown, He Yang

**Abstract:**

## 1. Introduction

The San Francisco government of California publishes their public data onto their government website. Among these are police incident reports. This paper aims to predict when specifically Missing Person incidents will occur in a matter of number of occurrences per month. It also may venture to see if a case is likely to be resolved based on available indicators.

Predicting crimes could be useful to a police department in terms of allocating resources which in term saves time and money for the police department itself the the tax paying citizens of a city.

The methods employed utilize the programming language R and various libraries openly available on the internet. These libraries can be created by anyone since R is an open-source free software. This paper uses many libraries such as lubridate, forecast, prophet, and Metrics among others to properly analyze the data.

R will be used to forecast time-series data with multiple statistical models. Of these models are Simple Exponential Smoothing, Autoregressive Integrated Moving Average (ARIMA), Facebook Prophet, K-Nearest Neighbors (KNN), and Long Term Short Memory (LTSM).

## 2. Data

As mentioned previously, all data is pulled from the San-Francisco government's official website (data.sfgov.org). We will be analyzing the dataset called *Police Department Incident Reports: 2018 to Present*. This dataset consists of over 760,000 data points and 35 columns. Missing Person data is only 16,457 data points. Because much of 2023's data is incomplete, we decided to narrow our focus to 2018 through 2022 making the number of data points 14,814.

This makes 14,814 data points to be analyzed from 2018 to 2022. The data was split into two separate parts for the purpose of forecasting. The data from 2018-2021 will be used to predict the number of incidents per month in the first half of 2023. The terminology for 2018-2021 is the training set whereas 2023 is the testing set. The more accurately the training set can predict the testing set, the more accurate the model.

Data pre-processing for this dataset was quite simple. We only wanted to analyze “initial reports" as supplemental reports are extensions of the same incident. Counting supplemental as well would result in double-counting. When only taking the initial reports, the data points fall from 14,814 to 9,494 data points. That is 9,494 data points throughout the entirety of 2018 till the end of 2023. After this, since we are only counting the number of incidents, the only thing to do is count the number of incident reports per month and by year. From this point, any null values in other columns are irrelevant as long as each incident has a datetime. Every datapoint in the San-Francisco dataset has a datetime, so this is of no worry. Most models were able to run R’s built-in time-series structure indicated as ts(). However, Facebook Prophet requires its own data structure so the data had to be restructured into two columns denoted ds (datetime) and y (values).

## 3. Methodology

### *3.1 Exponential Smoothing Algorithm*

The R library used for exponential smoothing was HoltWinters [7]. The standard exponential smoothing algorithm uses the previous actual value times alpha plus actual value of the period you are trying to calculate times 1 minus alpha to get the forecast for that given period. Here, alpha is a weight that prioritizes the previous value between 0 and 1. For example alpha of .5 would be equal to averaging the previous and current period value to get the forecast.

The formula that HoltWinters uses is [7]:

$$\hat{Y}[t+h] = a[t] + hb[t] + s[t-p+1+(h-1) \bmod p],$$

Here, a is alpha, b is beta, and s is gamma. Y is the forecast, t is the value at the observed time, p is the time series’ period length, and h is the selected number of periods to forecast.

We did one calculation with alpha which is the standard calculation known as Simple Exponential Smoothing. Using alpha and beta is known as Holt’s Method or Double Exponential Smoothing [6]. The addition of beta allows trends to be modeled.

HoltWinters automatically calculates alpha and beta through embedded formulas denoted as [7]:

$$a[t] = \alpha(Y[t] - s[t-p]) + (1-\alpha)(a[t-1] + b[t-1])$$

$$b[t] = \beta(a[t] - a[t-1]) + (1-\beta)b[t-1]$$

### *3.3 Autoregressive Integrated Moving Average*

Autoregressive Integrated Moving Average (ARIMA) was calculated using the Arima function in the R package forecast [8]. ARIMA utilizes the dependent relationship between observations and lagged observations to create a moving average.

The formula used by forecast's ARIMA [8]:

$$X_t = a_1 X_{t-1} + \cdots + a_p X_{t-p} + e_t + b_1 e_{t-1} + \cdots + b_q e_{t-q}$$

Importantly, Arima uses values p, d, and q where p is the AR order, d is the degree of differencing, and q is the moving average (MA) order. The function auto.arima in forecast will automatically provide you with the correct p, d, and q.

What do other values mean? I don't know.

### *3.4 Facebook Prophet*

Our prophet observation was granted with the use of the R package Prophet. Prophet uses trends, seasons, and holidays to make trendlines then choose the best predictors for forecasting [9]. As prophet is modeled after daily data, unlike with the other methods used, we had to feed prophet the daily crime reports instead of feeding it per month data. For comparison with other methods, we took the results for each day and summed them up into their respective months to get an overall month value.

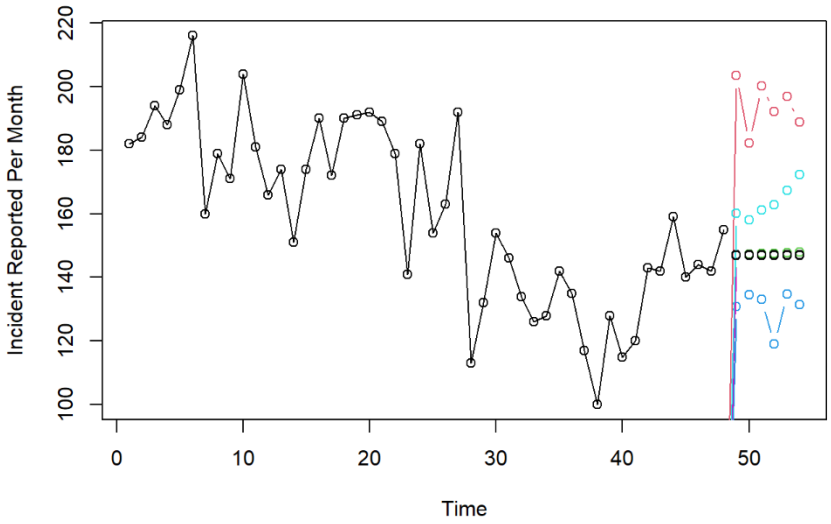
### *3.5 K Nearest Neighbors*

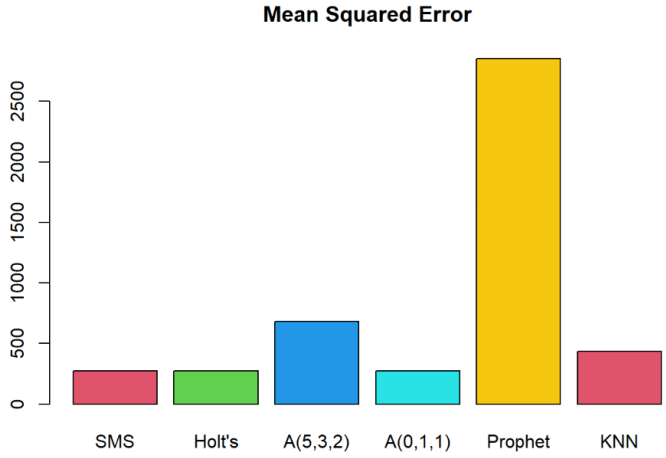
This model was achieved with the help of the R package tsfknn [10]. K Nearest Neighbors (KNN) is a model where data is grouped into clusters based on similarity to each other. It forecasts a value by comparing its properties with already existing values.

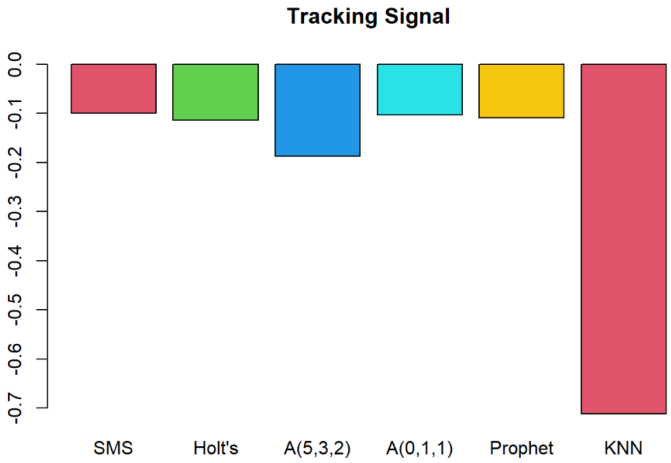
### *3.6 Long Term Short Memory*

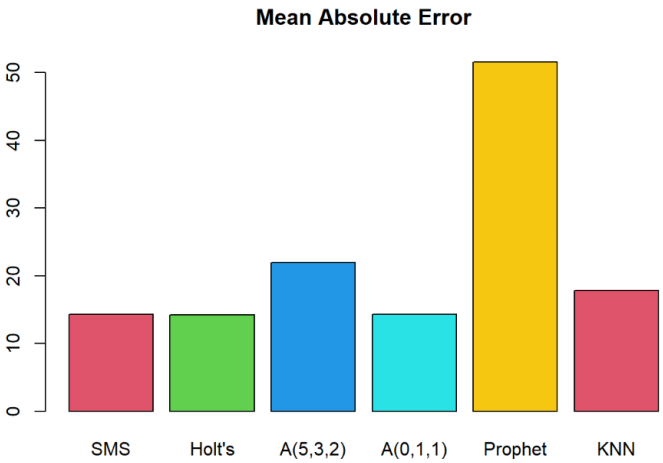
## 4. Results & Comparison

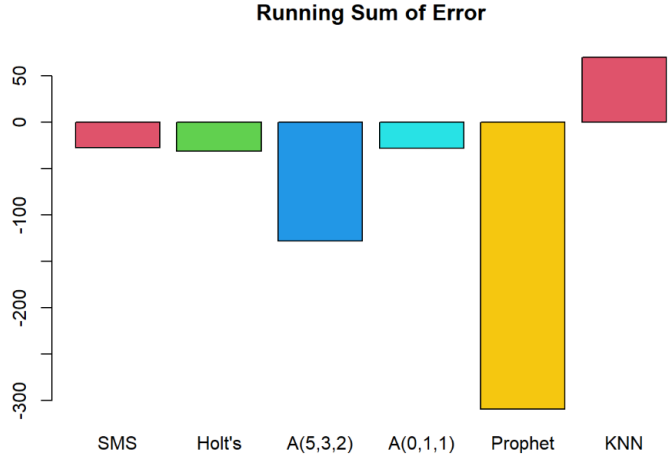
### *4.1 Baseline Comparison*


Incident Reported Per Month
100
120
140
160
180
200
220
0
10
20
30
40
50
Time


Mean Squared Error
0
500
1000
1500
2000
2500
SMS
Holt's
A(5,3,2)
A(0,1,1)
Prophet
KNN

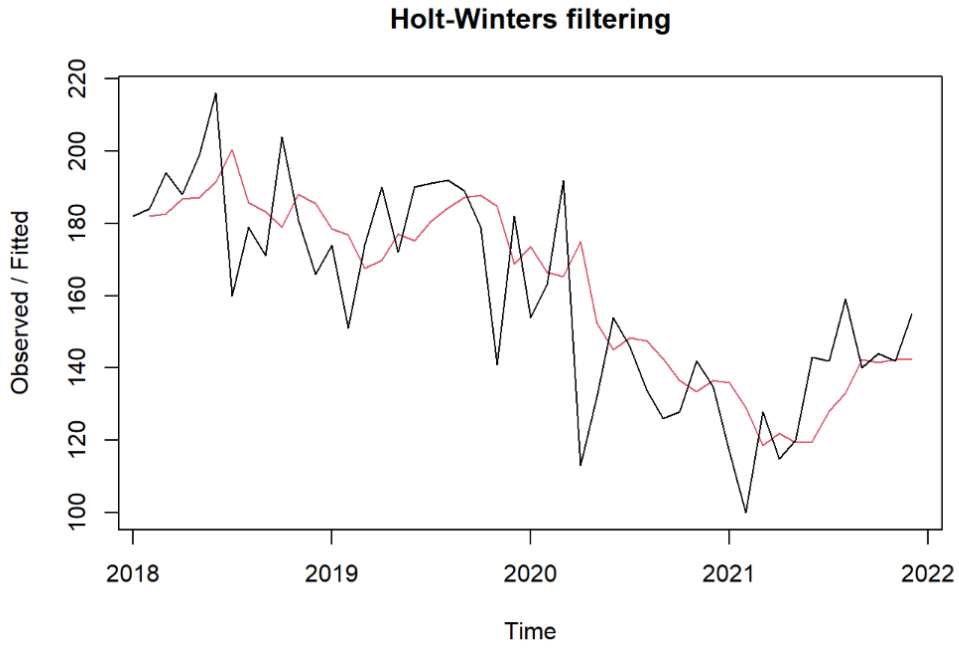

Tracking Signal
-0.7
-0.6
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
SMS
Holt's
A(5,3,2)
A(0,1,1)
Prophet
KNN

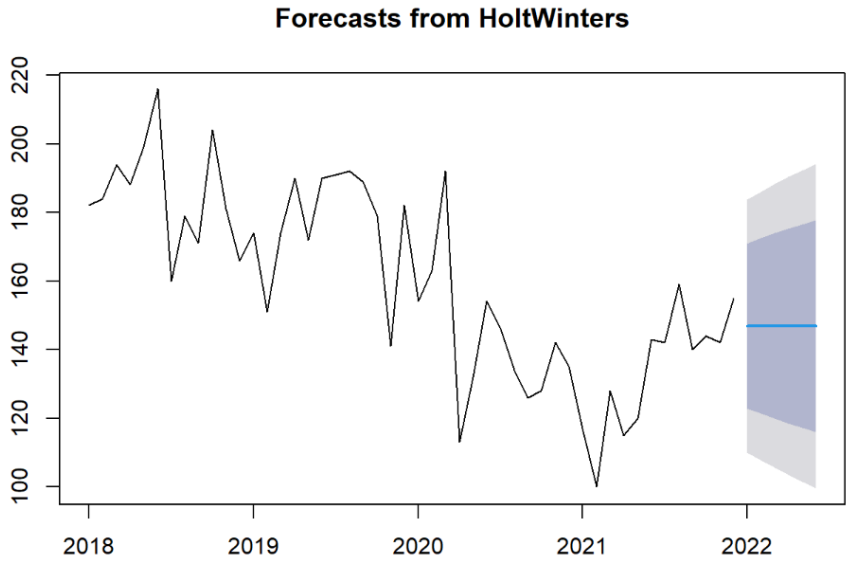

Mean Absolute Error
0
10
20
30
40
50
SMS
Holt's
A(5,3,2)
A(0,1,1)
Prophet
KNN


Running Sum of Error
-300
-200
-100
0
50
SMS
Holt's
A(5,3,2)
A(0,1,1)
Prophet
KNN

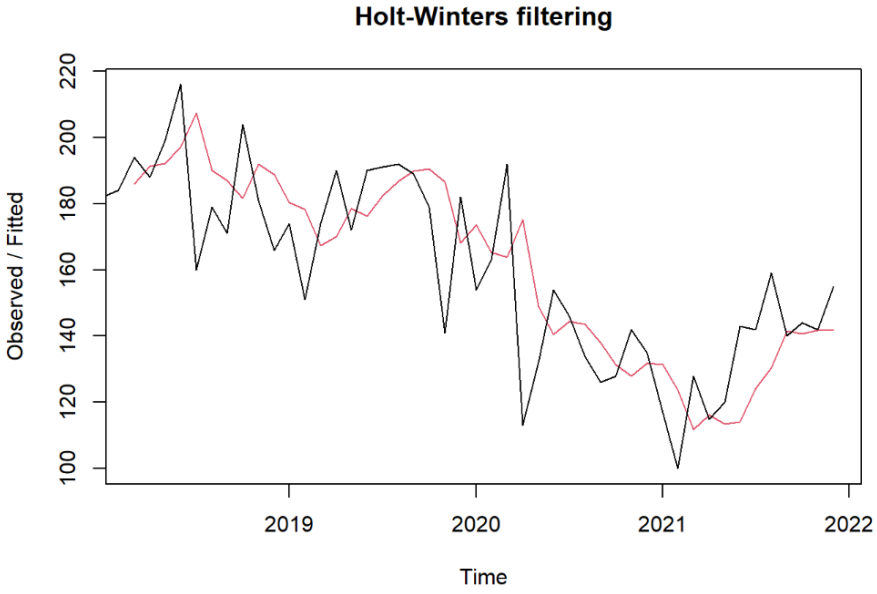
### *4.2 Section Where Jessica Adds All Her Extra Graphs*

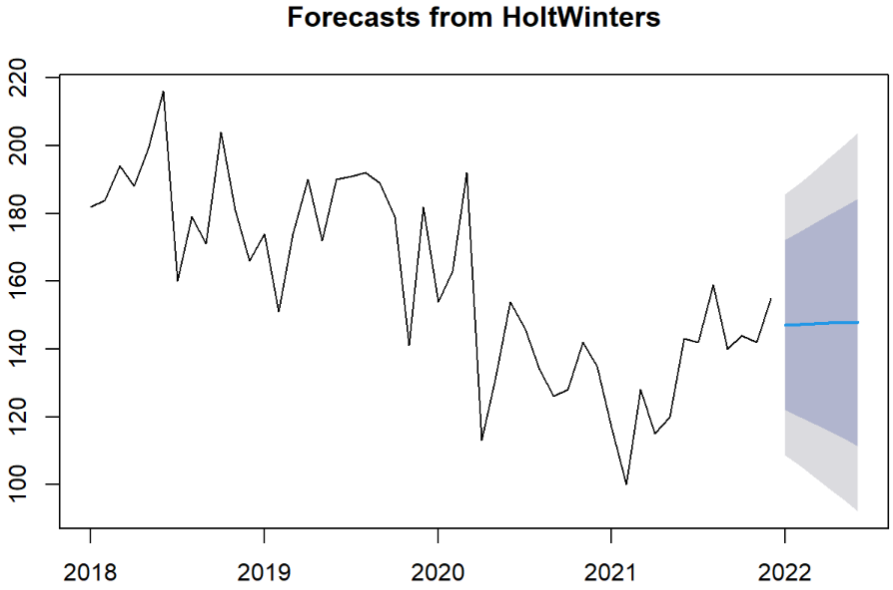
Simple Exponential Smoothing



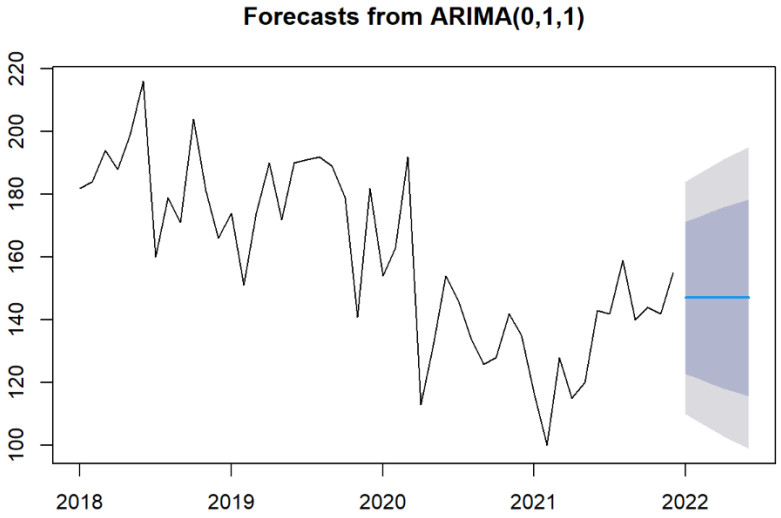




Double Exponential Smoothing

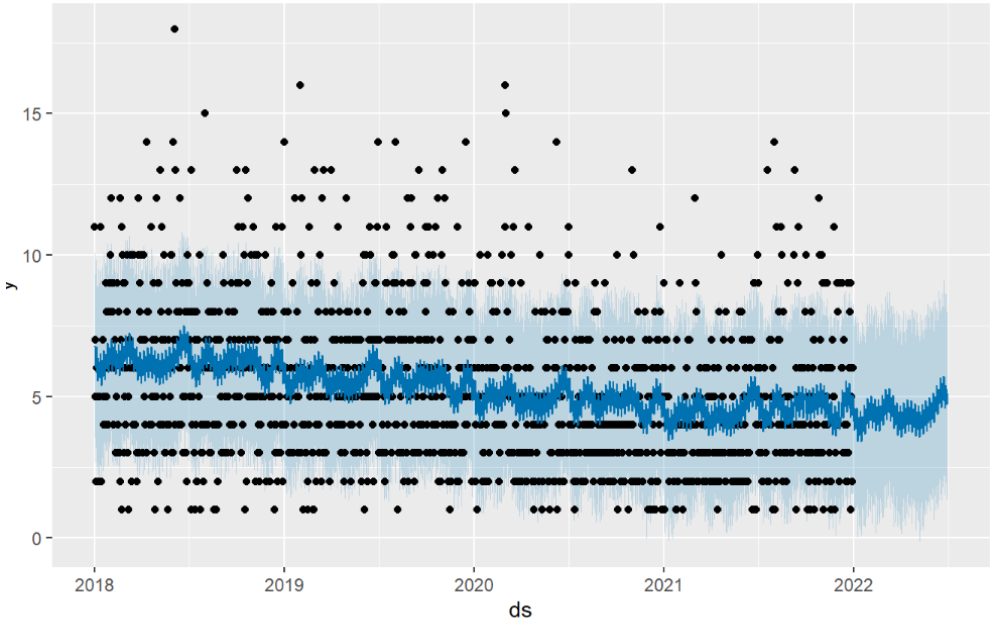

Holt-Winters filtering
Observed / Fitted
100
120
140
160
180
200
220
2019
2020
2021
2022
Time

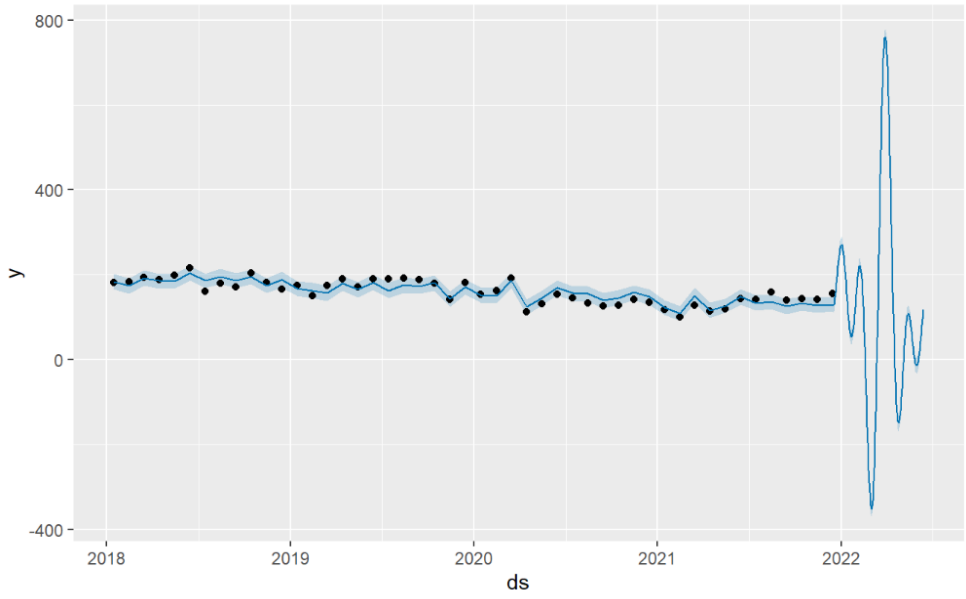

Forecasts from HoltWinters
100
120
140
160
180
200
220
2018
2019
2020
2021
2022

ARIMA(0, 1, 1)

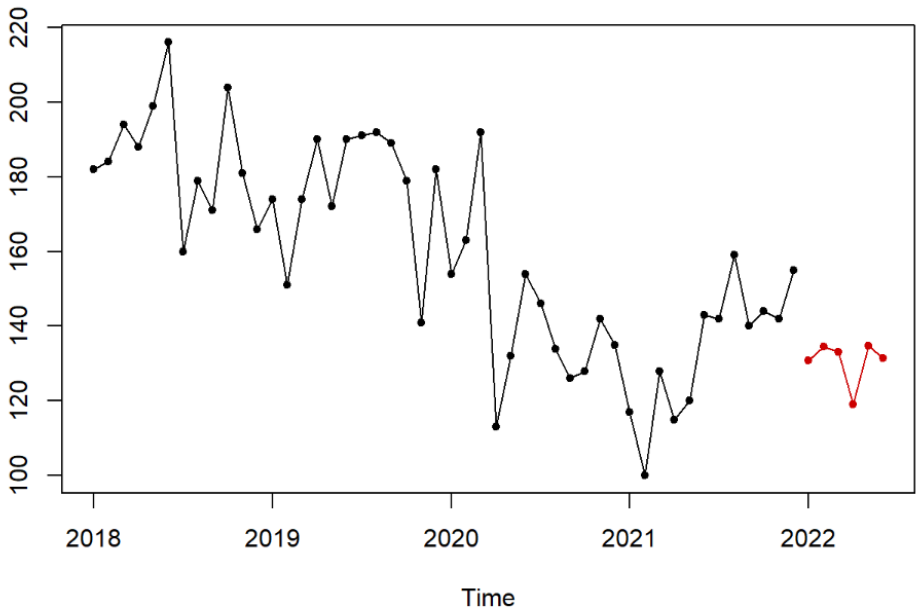



Facebook Prophet




800
400
0
-400
y
2018
2019
2020
2021
2022
ds

KNN


220
200
180
160
140
120
100
2018
2019
2020
2021
2022
Time

**RESOURCES**


https://data.sfgov.org/Public-Safety/Police-Department-Incident-Reports-2018-to-Present/wg3w-h783 [1]

https://datasf.gitbook.io/datasf-dataset-explainers/sfpd-incident-report-2018-to-present [2]

https://cran.r-project.org/web/packages/tsfknn/vignettes/tsfknn.html [3]

http://rwanjohi.rbind.io/2018/04/05/time-series-forecasting-using-lstm-in-r/ [4]

http://www.iapress.org/index.php/soic/article/view/1767/1025 [5]

https://statisticsbyjim.com/time-series/exponential-smoothing-time-series-forecasting/ [6]

https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/HoltWinters [7]

https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/arima [8]

https://youtu.be/2XFro0nIHQM?si=Wbqnvh383Xx_Y2CC [9]

https://cran.r-project.org/web/packages/tsfknn/vignettes/tsfknn.html [10]

https://www.youtube.com/watch?v=c0k-YLQGKjY&t=301s [11]