Car Crash Predictions Using Regression Model Learning

by Derek Xu, Anthony O'Neil, David Xiang

Article Citation and Synopsis

Automobile crashes are one of the top causes of death and injury in the United States today. *Motor Vehicle Collisions - Crashes* describes one way in which New York City has moved to help eliminate this reoccurring issue. The article explains the way in which they utilized NYPD police reports on crashes to help develop a dataset that describes where, when, and how automobile and automobile related crashes occur within New York City. These reports (the MV104-AN report) stemmed from two programs, the CompStat program and the TrafficStat program. The CompStat program was designed to help the police “reduce crime and achieve other police department goals” utilizing accurate and timely information, rapid deployment of resources, effective methods, and thorough follow-up. TrafficStat is a subset of this program to help reduce and effectively understand the causes and effects of automobile crashes, specifically in New York City. With this process, the NYPD have gotten an immense number of data points on car crashes over $1000 within the city along with detailed data fields such as Date-Time, Borough, and Injury/Death number.

The main issue with this article and the data set that it creates is the fact that the motor vehicle collisions model only extracts data from police reports within New York City, meaning that it might be representative of crashes in other major cities, but more likely to not represent crashes within suburbs or rural areas. Another form of bias is the fact that reports are only made on crashes above $1000, meaning that the motor vehicle crashes dataset also only has data on those incidents. This means that many smaller events, such as parking collisions or slight cosmetic damage crashes will note be reported and not be present. Lastly, a major point of bias with this article and its methodology is that it utilizes police reports to contribute to the dataset. Crashes could be happening at much higher rates and in many different areas but are not reported, meaning that the dataset could be misrepresenting the actual amount of crash cases. Overall, this dataset has a multitude of detailed data points and the article well describes the methodology in obtaining these statistics along with ways that they have improved the system using technology such as digital entry and CCTV.

Dataset and Sources

Injurystats - https://injuryfacts.nsc.org/motor-vehicle/overview/introduction/

Dataset and Article: https://catalog.data.gov/dataset/motor-vehicle-collisions-crashes

Compstat Paper: https://www.ojp.gov/ncjrs/virtual-library/abstracts/compstat-its-origins-evolution-and-future-law-enforcement-agencies

Background and High-Level Explaination of Problem

According to Injuryfacts.nsc.org, over motor vehicle accidents are one of the top causes of death and injury within the United States at 21%, just under the number one spot for poisoning. These accidents are caused by a multitude of factors that can all come together to ultimately create a devastating incident. Many of these crashes can become fatal especially when it becomes car to pedestrian incidents. For our project, we have decided to analyze a government dataset on vehicular collisions in New York City in an attempt to find out what one major factor of these various variables contributes to automobile crashes, the time. The *Motor Vehicle Collisions -*

Crashes dataset includes a plethora of variables, including the weather (and more specifically, which season of the year it is), recklessness, road rage, time, and more.

In our case, we want to analyze and accurately predict the fatality of crashes based on the time since the time contributes to many factors such as the number of cars on the road, what type of drivers there are (workers, drunk drivers, etc.) and visibility. We prioritize creating a regression model that will take the time of day as the input and give us the cumulation of the number of deaths and injuries that occur within crashes in New York City. In creating a model that accurately describes the number of deaths and injuries within certain time frames, we can use this info to predict for other cities as well as other areas, what type of civil improvements that are needed. For example, if the most of the crashes occur mid-day during rush hour, then it is clear that reducing the number of cars on the road, through means of public transportation as a possible solution, would be needed to reduce automobile fatality. On the other hand, if crashes occur mostly at night, there would be other factors to look into such as decreasing the number of drunk drivers or improving visibility for drivers through means of more street lights. Overall, in solving this problem where we find the likelihood of automobile related crashes based on the time, we hope to create an understandable answer that can help drivers and pedestrians find safer times to travel and avoid incidents. More importantly, we hope that our machine learning model will help civil engineers and other city planners to create more effective infrastructure to decrease the total number of automobile related incidents as a whole.

Solution Process and Inferences

To find the answer to our problem, we first shrunk our sample size. We took one random datapoint in every 50 of the original dataset, to avoid any data bias. We then removed points missing information that may be vital: Borough, Street, and Cause of Collision. Once the data was scrubbed, we attempted to use the Classification and Regression Learners to produce valuable data; we realized that the Regression Learner would be more useful than the Classification Learner, so we stuck with that. However, our first few attempts at datamatching proved unsuccessful. The combinations of Injuries/Kills and Boroughs and Injuries/Kills and Streets didn't work, as we needed numerical values to do a true comparison. While giving the Boroughs a numerical ID would work, the result would be similar to a bar graph. Finally, we used Time against Injuries/Kills, which proved both successful and useful, so we stuck with that.

Importing the Data

```
tic
crashdata = table2struct(webread("https://data.cityofnewyork.us/api/views/h9gi-nx95/row
```

Scrubbing the Data

Determining the size of the dataset, as well as how many points are actually going to be analyzed.

```
[r,~] = size(crashdata);
fprintf("There are a total of %d datapoints\n",r)
```

```
There are a total of 1951515 datapoints
```

```
fprintf("Since we don't want nearly that many datapoints, we'll use one random point i
```

```
Since we don't want nearly that many datapoints, we'll use one random point in ever 50 in the set
```

```
format long g
maxval = floor(r/50);

%this loop eliminates all datapoints outside
for i = r:-1:(maxval*50)+1
    crashdata(i) = [];
end

%under this, the last row of cleandata is given the last row of crashdata.
%This is temporary, and used to allow preallocation.
clear cleandata
cleandata(maxval) = crashdata(maxval*50);
for i = 1:maxval
    rannum = randi([1,50]);
    cleandata(i) = crashdata(rannum + ((i-1)*50));
end

cleandata = cleandata'
```

```
cleandata = 39030×1 struct
```

...

Fields	CRASHDATE	CRASHTIME	BOROUGH	ZIPCODE	LATITUDE	LONGITUDE	LOCATION	ONSTREET...
1	1×1 datetime	'8:30'	''	NaN	NaN	NaN	''	'broadway'
2	1×1 datetime	'17:08'	'BROOKLYN'	11205	40.698463	-73.960205	'(40.698...	'FLUSHIN...
3	1×1 datetime	'22:51'	''	NaN	40.74967	-73.99531	'(40.749...	'WEST 30...
4	1×1 datetime	'14:30'	'STATEN ...	10305	40.596733	-74.07045	'(40.596...	'MCCLEAN...
5	1×1 datetime	'21:34'	'BRONX'	10469	40.87645	-73.848175	'(40.876...	'BOSTON ...
6	1×1 datetime	'10:25'	'MANHATTAN'	10128	40.781315	-73.94614	'(40.781...	'1 AVENUE'
7	1×1 datetime	'16:40'	''	NaN	40.590313	-74.09794	'(40.590...	'JEFFERS...
8	1×1 datetime	'0:00'	''	NaN	40.607754	-74.13205	'(40.607...	'BRADLEY...
9	1×1 datetime	'10:55'	'MANHATTAN'	10028	40.779602	-73.95553	'(40.779...	''
10	1×1 datetime	'14:50'	'BRONX'	10461	40.843464	-73.836	'(40.843...	''
11	1×1 datetime	'6:45'	'BROOKLYN'	11221	40.691822	-73.92223	'(40.691...	'BUSHWIC...
12	1×1 datetime	'16:55'	''	NaN	40.68039	-73.94956	'(40.680...	'FULTON ...
13	1×1 datetime	'23:55'	''	NaN	40.762676	-73.954346	'(40.762...	'FDR DRIVE'
14	1×1 datetime	'0:00'	''	NaN	40.780437	-73.94989	'(40.780...	'EAST 90...

⋮

```
[rclean,~] = size(cleandata);
fprintf("The size of the scrubbed data is now %d\n",rclean)
```

```
The size of the scrubbed data is now 39030
```

Now that the datapool has been decreased to a workable size, we're going to elminate datapoints that won't help us. This includes ones with no specified Borough, no specified causation factor (for at least 1 vehicle), and no specified streetname.

NB: Each step in the following scrubbing is run 5 times. MATLAB tends to miss a value occasionally, so repeating the process solves this issue.

```
%this loop eliminates values with no Borough
for i = 1:5
    idex = 1;
    while idex <= rclean
        if isempty(cleandata(idex).BOROUGH) || isnan(cleandata(idex).ZIPCODE)
                cleandata(idex) = [];
        end
        idex = idex + 1;
        [rclean,~] = size(cleandata);
    end
end

cleandata
```

```
cleandata = 26896×1 struct
```

...

Fields	CRASHDATE	CRASHTIME	BOROUGH	ZIPCODE	LATITUDE	LONGITUDE	LOCATION	ONSTREET...
1	1×1 datetime	'17:08'	'BROOKLYN'	11205	40.698463	-73.960205	'(40.698...	'FLUSHIN...
2	1×1 datetime	'14:30'	'STATEN ...	10305	40.596733	-74.07045	'(40.596...	'MCCLEAN...
3	1×1 datetime	'21:34'	'BRONX'	10469	40.87645	-73.848175	'(40.876...	'BOSTON ...
4	1×1 datetime	'10:25'	'MANHATTAN'	10128	40.781315	-73.94614	'(40.781...	'1 AVENUE'
5	1×1 datetime	'10:55'	'MANHATTAN'	10028	40.779602	-73.95553	'(40.779...	''
6	1×1 datetime	'14:50'	'BRONX'	10461	40.843464	-73.836	'(40.843...	''
7	1×1 datetime	'6:45'	'BROOKLYN'	11221	40.691822	-73.92223	'(40.691...	'BUSHWIC...
8	1×1 datetime	'0:00'	'BRONX'	10463	40.875294	-73.9088	'(40.875...	''
9	1×1 datetime	'14:56'	'MANHATTAN'	10029	40.800262	-73.94554	'(40.800...	''
10	1×1 datetime	'22:30'	'BROOKLYN'	11214	40.605072	-73.997795	'(40.605...	''
11	1×1 datetime	'15:00'	'QUEENS'	11374	40.72304	-73.8563	'(40.723...	'67 AVENUE'
12	1×1 datetime	'13:48'	'MANHATTAN'	10022	40.764896	-73.97256	'(40.764...	'5 AVENUE'
13	1×1 datetime	'18:40'	'QUEENS'	11368	40.756256	-73.85773	'(40.756...	''
14	1×1 datetime	'20:14'	'BROOKLYN'	11209	40.619274	-74.03203	'(40.619...	''

⋮

```
%this loop elmimnates values with no streetname
for i = 1:5
```

```
    idex = 1;
    while idex <= rclean
        if isempty(cleandata(idex).ONSTREETNAME)
                cleandata(idex) = [];
        end
        idex = idex + 1;
        [rclean,~] = size(cleandata);
    end
end
cleandata
```

```
cleandata = 21295×1 struct
```

...

Fields	CRASHDATE	CRASHTIME	BOROUGH	ZIPCODE	LATITUDE	LONGITUDE	LOCATION	ONSTREET...
1	1×1 datetime	'17:08'	'BROOKLYN'	11205	40.698463	-73.960205	'(40.698...	'FLUSHIN...
2	1×1 datetime	'14:30'	'STATEN ...	10305	40.596733	-74.07045	'(40.596...	'MCCLEAN...
3	1×1 datetime	'21:34'	'BRONX'	10469	40.87645	-73.848175	'(40.876...	'BOSTON ...
4	1×1 datetime	'10:25'	'MANHATTAN'	10128	40.781315	-73.94614	'(40.781...	'1 AVENUE'
5	1×1 datetime	'6:45'	'BROOKLYN'	11221	40.691822	-73.92223	'(40.691...	'BUSHWIC...
6	1×1 datetime	'15:00'	'QUEENS'	11374	40.72304	-73.8563	'(40.723...	'67 AVENUE'
7	1×1 datetime	'13:48'	'MANHATTAN'	10022	40.764896	-73.97256	'(40.764...	'5 AVENUE'
8	1×1 datetime	'19:37'	'QUEENS'	11413	40.665497	-73.75573	'(40.665...	'SOUTH C...
9	1×1 datetime	'5:13'	'BROOKLYN'	11211	40.703033	-73.9599	'(40.703...	'BEDFORD...
10	1×1 datetime	'20:13'	'BRONX'	10464	40.84969	-73.78744	'(40.849...	'DITMARS...
11	1×1 datetime	'12:25'	'BROOKLYN'	11207	40.65906	-73.88459	'(40.659...	'VAN SIC...
12	1×1 datetime	'16:00'	'MANHATTAN'	10023	40.77178	-73.9792	'(40.771...	'WEST 65...
13	1×1 datetime	'22:52'	'BROOKLYN'	11235	40.580193	-73.944916	'(40.580...	'HAMPTON...
14	1×1 datetime	'0:55'	'QUEENS'	11375	40.7243	-73.84517	'(40.724...	'108 STREET'

⋮

```
%finally, this loop eliminates datapoints with unspecified causes
for i = 1:5
    idex = 1;
    while idex <= rclean
        if  strcmp(cleandata(idex).CONTRIBUTINGFACTORVEHICLE1,'Unspecified') || isempt
                cleandata(idex) = [];
        end
        idex = idex + 1;
        [rclean,~] = size(cleandata);
    end
end
cleandata
```

```
cleandata = 12743×1 struct
```

...

Fields	CRASHDATE	CRASHTIME	BOROUGH	ZIPCODE	LATITUDE	LONGITUDE	LOCATION	ONSTREET...
1	1×1 datetime	'17:08'	'BROOKLYN'	11205	40.698463	-73.960205	'(40.698...	'FLUSHIN...
2	1×1 datetime	'14:30'	'STATEN ...	10305	40.596733	-74.07045	'(40.596...	'MCCLEAN...
3	1×1 datetime	'21:34'	'BRONX'	10469	40.87645	-73.848175	'(40.876...	'BOSTON ...
4	1×1 datetime	'10:25'	'MANHATTAN'	10128	40.781315	-73.94614	'(40.781...	'1 AVENUE'
5	1×1 datetime	'6:45'	'BROOKLYN'	11221	40.691822	-73.92223	'(40.691...	'BUSHWIC...
6	1×1 datetime	'15:00'	'QUEENS'	11374	40.72304	-73.8563	'(40.723...	'67 AVENUE'
7	1×1 datetime	'19:37'	'QUEENS'	11413	40.665497	-73.75573	'(40.665...	'SOUTH C...
8	1×1 datetime	'5:13'	'BROOKLYN'	11211	40.703033	-73.9599	'(40.703...	'BEDFORD...
9	1×1 datetime	'16:00'	'MANHATTAN'	10023	40.77178	-73.9792	'(40.771...	'WEST 65...
10	1×1 datetime	'22:52'	'BROOKLYN'	11235	40.580193	-73.944916	'(40.580...	'HAMPTON...
11	1×1 datetime	'0:55'	'QUEENS'	11375	40.7243	-73.84517	'(40.724...	'108 STREET'
12	1×1 datetime	'16:32'	'BROOKLYN'	11219	40.626183	-73.994156	'(40.626...	'60 STREET'
13	1×1 datetime	'0:53'	'BROOKLYN'	11236	40.627857	-73.90075	'(40.627...	'EAST 80...
14	1×1 datetime	'15:21'	'MANHATTAN'	10027	40.810276	-73.94738	'(40.810...	'7 AVENUE'

⋮

```
fprintf("The final size of the data is %d datapoints\n",rclean)
```

```
The final size of the data is 12743 datapoints
```

```
toc
```

```
Elapsed time is 486.045942 seconds.
```

```
save cleandata.mat cleandata
```

Finally, the data can be analyzed in the ML Toolbox

Vectorizing Code

In this section, three vectors are created: one is the total amount of people killed per crash, another is the total amount of those injured, and the last is the time of each of the crashes. These will be used in the machine learning toolbox.

```
load cleandata.mat
[rclean, cclean] = size(cleandata);
kills = zeros(rclean,1);
for i = 1:rclean
    kills(i) = cleandata(i).NUMBEROFPERSONSKILLED + cleandata(i).NUMBEROFPEDESTRIANSKI
end
kills
```

```
kills = 12868×1
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     ⋮
```

```
injurs = zeros(rclean,1);
for i = 1:rclean
    injurs(i) = cleandata(i).NUMBEROFPERSONSINJURED + cleandata(i).NUMBEROFPEDESTRIANSI
end
injurs
```

```
injurs = 12868×1
     2
     2
     2
     0
     6
     0
     6
     0
     0
     0
     ⋮
```

```
time = zeros(rclean,1);
for i = 1:rclean
    temp = cleandata(i).CRASHTIME;
    time(i) = str2double(erase(temp, ':'));
end
time
```

```
time = 12868×1
          102
         1755
          840
         2221
            0
          315
         1030
         1700
         1138
         1730
     ⋮
```

```
%learnerinfo = table(injurs,kills,boroughs)
save times.mat time
save injuries.mat injurs
save kills.mat kills
```

Using Kill and Injurs Data

Here, the injury and kills data are summed up.

```
suminj = zeros(1,48);
lbt = 0;
hbt = 50;
for i = 1:48
    lbt = 0 + 50 * (i-1);
    hbt = 50 + 50 * (i-1);
    for j = 1:length(cleandata)
        if (str2double(erase(cleandata(j).CRASHTIME,':')))  >=lbt-20 && (str2double(era
        suminj(i) = suminj(i) + cleandata(j).NUMBEROFPERSONSINJURED + cleandata(j).NUM
        end
    end
end

suminj = suminj';

%% This function depends on which time interval the crash happen, it adds the injuries
```

Using Time Data

Divide the whole day into 24 intervals.

```
lb = 0;
hb = 50;
num = 1;
tinterval = zeros(1,length(time));
while hb <= 2400
    for i = 1:length(time)

        if (time(i)>= lb-20 && time(i)<hb-20)
            tinterval(i) = num;
        end
    end
    num = num+1;
    lb = lb + 50;
    hb = hb + 50;
end
tinterval = tinterval';
%% this catagorize the crashe times into 48 different intervals
```

Check the count of each time interval

```
tfreq = zeros(1,48);
while num >= 1
    for i = 1: length(time)
        if (tinterval(i) == num )
            tfreq(num) = tfreq(num) + 1;
        end
    end
    num = num -1;
end
tfreq = tfreq';

%% this count how often does crashes happens in each time intervals
```

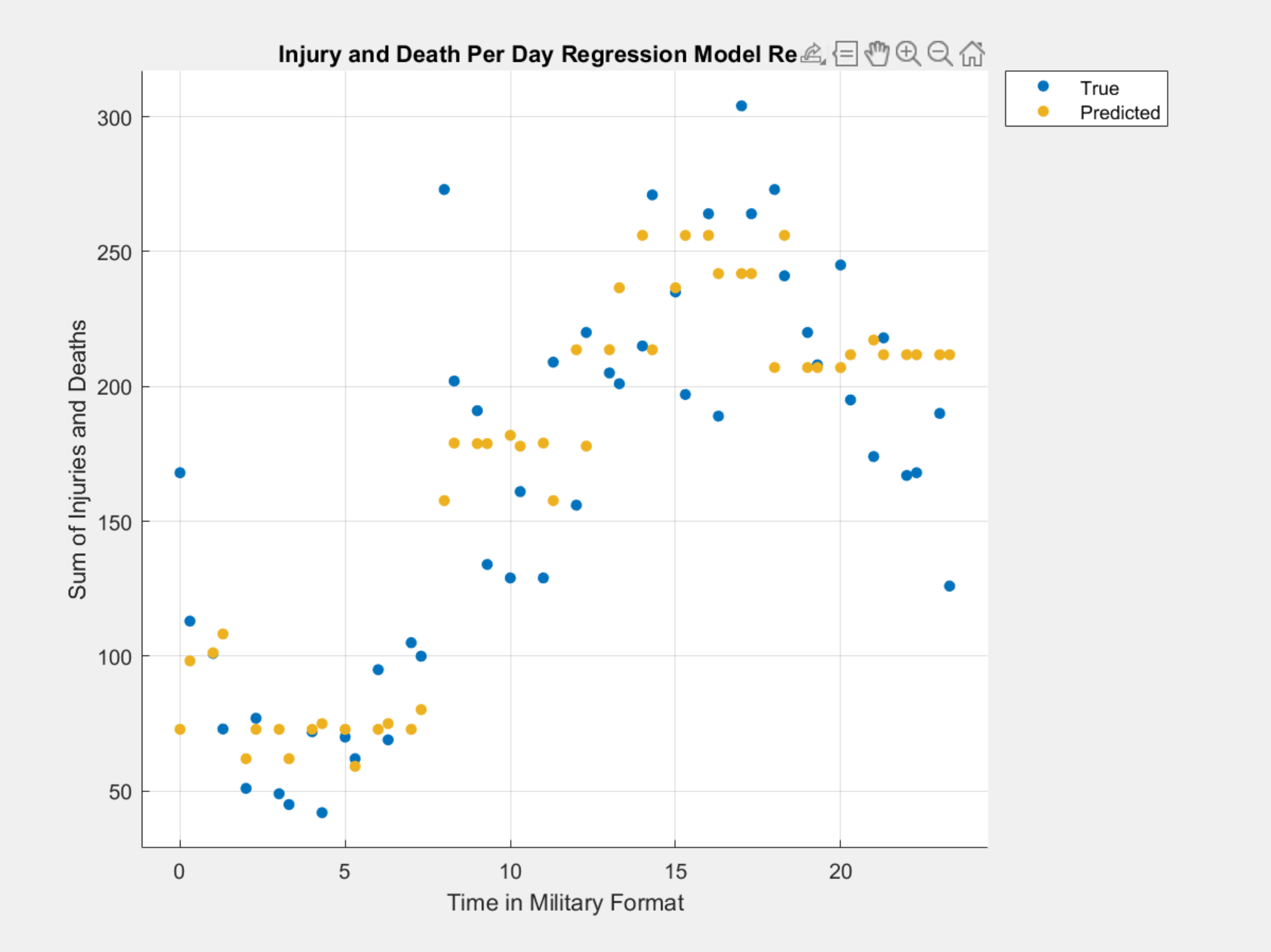
Creating the timevector to graph:

```
timevec = strings([48,1]);
hourvec = zeros(1,48);
minvec = zeros(1,48);
sub = 0;
for i = 1:2:48
    hourvec(i) = sub;
    hourvec(i+1) = sub;
    sub = sub + 1 ;
end
for i = 1:2:48
    minvec(i) = 0;
    minvec(i+1) = 30;
end
strhour = string(hourvec);
strmin = string(minvec);
for i = 1:48
    timevec(i) = strcat(strhour(i), "." , strmin(i));
end

timevec = str2double(timevec); %#ok<NASGU>

%% this creats a time interval vector for machine learning
```

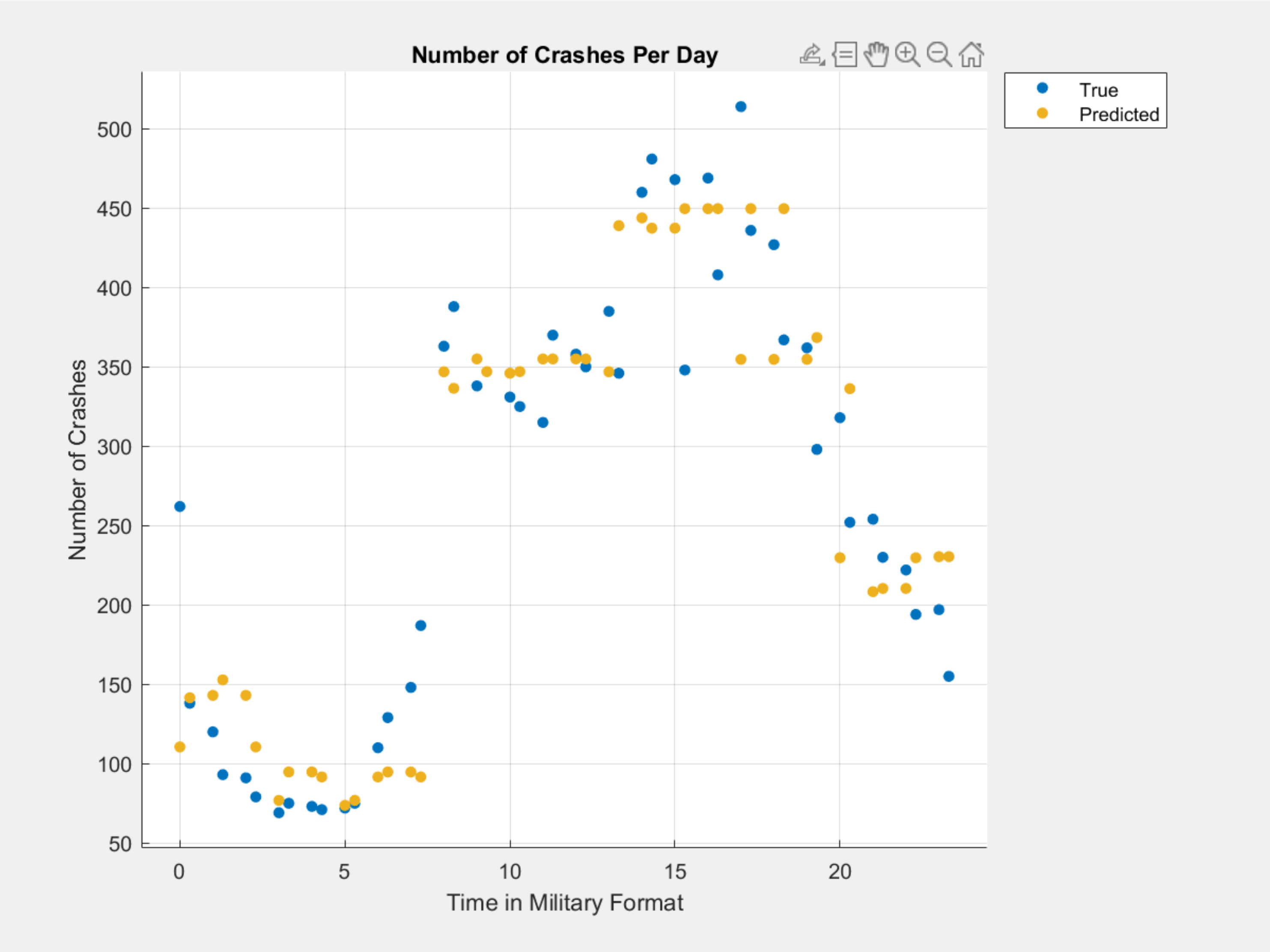
Creating Regression Model and Validating

To create the model, we use the time as the input and the cumulation of deaths + injuries as the output. The model will then allow us to predict future times and their respective fatality rates for one day.

```
% loading the predicted and true graph to show accuracy of model for
% injuries+deaths
% these two next figures help validate the accuracy of the model
load injuryModel.mat
injuryFig = openfig('modelInjuriesPlot.fig');

figure(injuryFig)
xlabel('Time in Military Format')
ylabel('Sum of Injuries and Deaths')
title('Injury and Death Per Day Regression Model Results')
load frequencyModel.mat
frequencyFig = openfig('modelFrequencyPlot.fig');
```



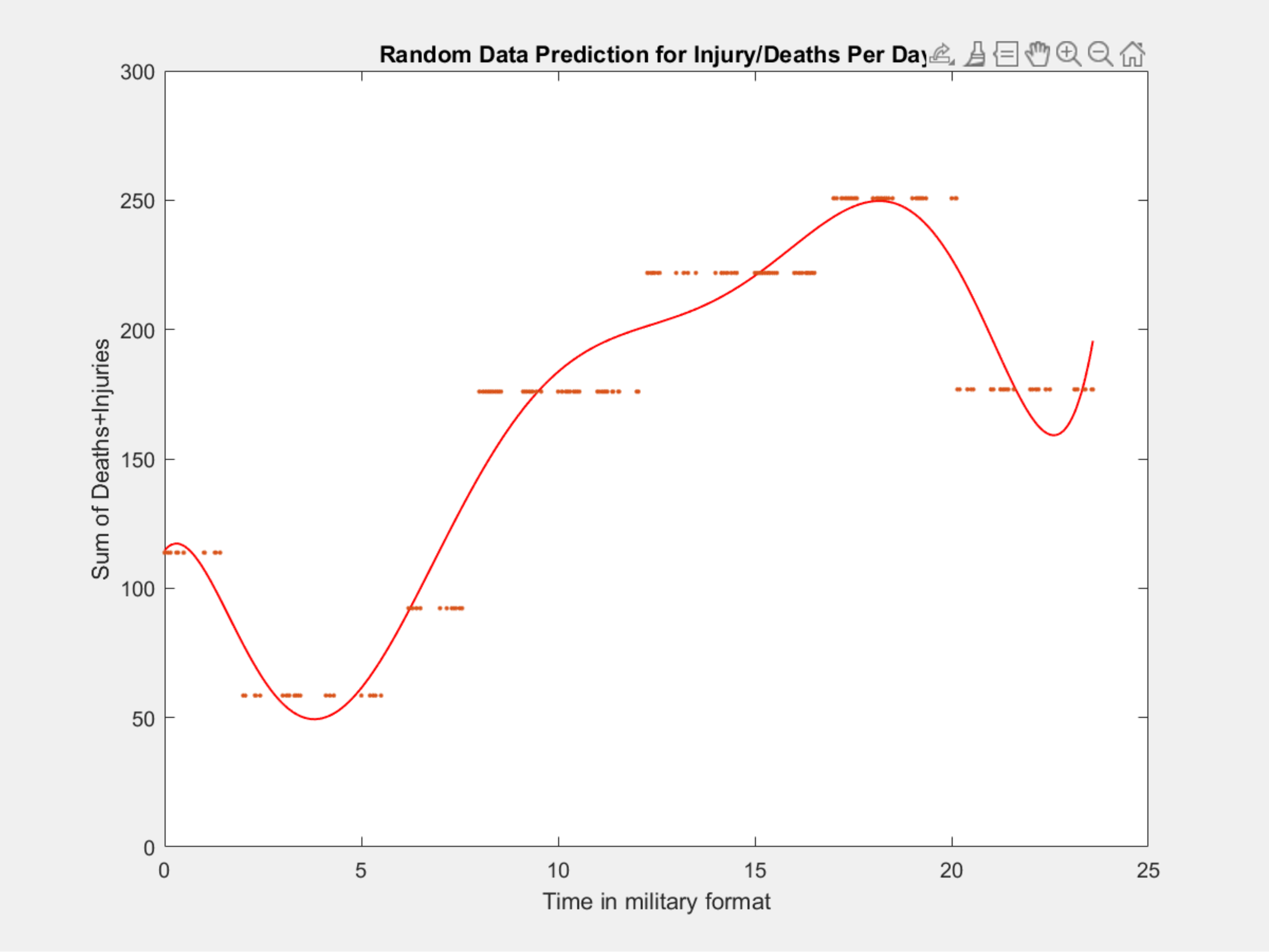
```
% doing the same with frequencies
figure(frequencyFig)
xlabel('Time in Military Format')
ylabel('Number of Crashes')
title('Number of Crashes Per Day')
```



Next section will help show a possible use case where we input random times into the model to give us results.

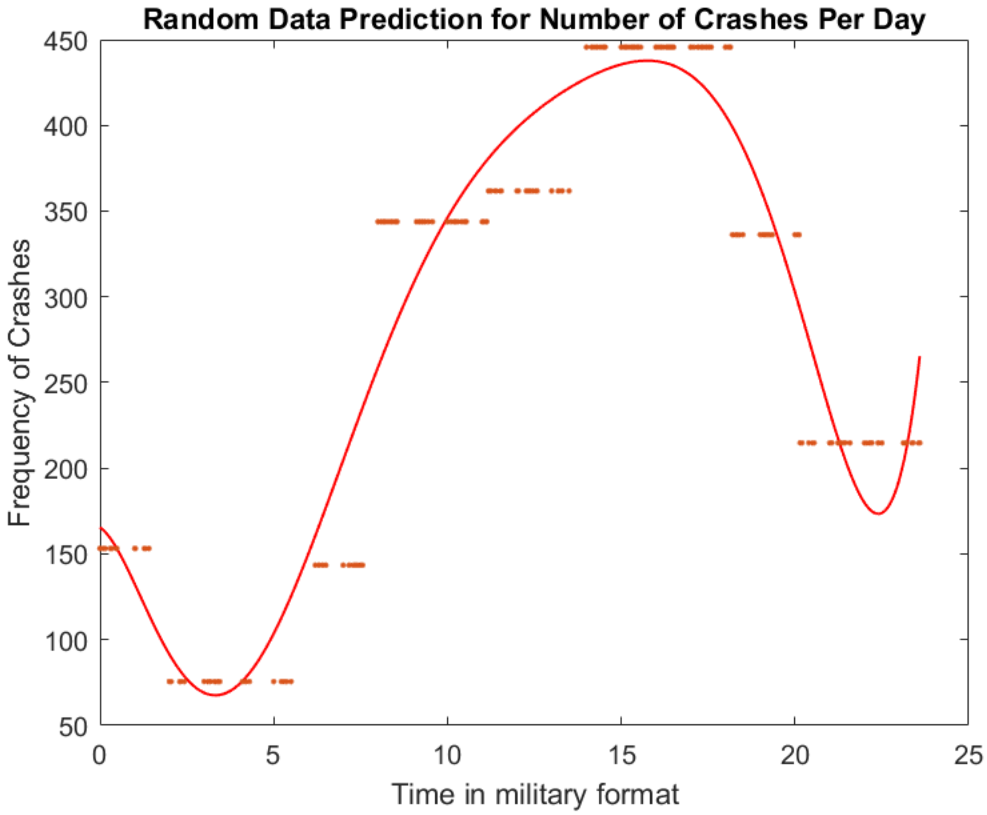
```
%creating a table of random data points to feed into model to show
% numerical results
load injuryModel.mat

% get random indices for the time
indices = randi([1, length(cleandata)], 300, 1);
timevec = zeros(300,1);
for i = 1:length(indices)
    temp = cleandata(i).CRASHTIME;
    timevec(i) = str2double(strrep(temp,':','.'));
end

% get predicted values
injuryTestOutput = injuryModel.predictFcn(timevec);

% polyfit
```

```
coefficients = polyfit(timevec, injuryTestOutput, 7);
xFit = linspace(min(timevec), max(timevec), 1000);
yFit = polyval(coefficients , xFit);
plot(xFit, yFit, 'r-', 'LineWidth', 1); % Plot fitted line.
hold on;

% plotting
plot(timevec, injuryTestOutput, '.')
xlabel('Time in military format')
ylabel('Sum of Deaths+Injuries')
title('Random Data Prediction for Injury/Deaths Per Day')
```



```
% same process for the frequency of crashes
figure
load frequencyModel.mat
frequencyTestOutput = frequencyModel.predictFcn(timevec);


% polyfit
```

```
coefficients = polyfit(timevec, frequencyTestOutput, 7);
xFit = linspace(min(timevec), max(timevec), 1000);
yFit = polyval(coefficients , xFit);
plot(xFit, yFit, 'r-', 'LineWidth', 1); % Plot fitted line.
hold on;

% plotting
plot(timevec, frequencyTestOutput, '.')
xlabel('Time in military format')
ylabel('Frequency of Crashes')
title('Random Data Prediction for Number of Crashes Per Day')
hold off
```



Conclusion

By referencing the graph, it can be inferred that the most dangerous time to drive in New York City is between 4pm and 7pm; considering that this is the usual time employees commute home, this lines up with normal logic. Furthermore, we also see that there is a decrease but still a significant amount of crashes during late night hours from 10pm to 1am. This could be also due to other factors such as drunk driving or poor visibility. When comparing the frequency of crashes to the number of injuries+deaths, we can also find that there are commonly more crashes than there are injuries+deaths. This is ratio is significantly higher especially higher during rush hours where a peak of 450 crashes have just about 250 injuries and deaths. This signifies that most of the crashes that happen within cities are small but prevalent. This could mean that there are issues with city planning in areas such as parking lots or other slower-paced areas, leading to more incidents without injury.

Overall, our model gives good estimates on the trends of car crashes throughout the day and gives good insight onto what problems might lead to these issues.