An Investigation Into What Causes Crash Injuries, Fatalities

Analysis of crash data from the City of Chicago using data mining

Nabil Darwich
Department of Computer Science
George Mason University
Fairfax VA USA
ndarwich@gmu.edu

Hamza Mughal
Department of Computer Science
George Mason University
Fairfax VA USA
hmughal2@gmu.edu

ABSTRACT

Being able to save human life is one of the most important goals facing various communities. An unfortunately common reality in which lives are lost is due to car accidents, and there are always public service announcements to advise on avoiding them. In this paper, an analysis is conducted regarding the true factors that lead to injuries and fatalities when it comes to car crashes. Data from the City of Chicago, during the summer of 2018, is used to find out what some of these causes are, and a model is built to determine accident severity outcomes given the elements involved. The summer season was picked to minimize external factors such as snow and road conditions leading to crashes, and the most recent summer was that of the current year, 2018. We use this data to build a training model, and evaluate how it performs against a never seen before test set (Summer 2017) A goal from this paper is to raise awareness regarding what can be done to avoid such tragedies.

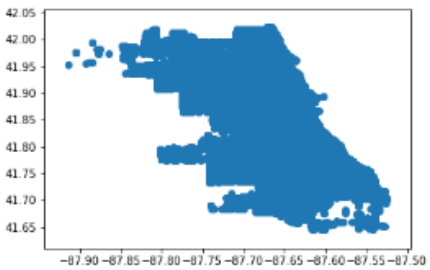
1 Introduction – Data Analysis

1.1 - Data

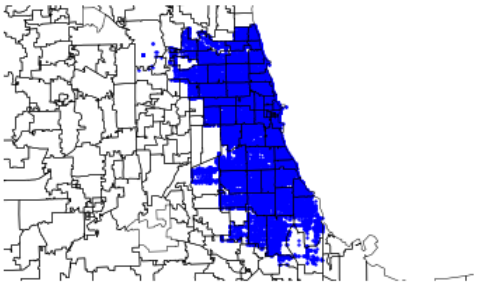
Data from the city of Chicago had 1.1M+ records split to 3 datasets (Crashes, Vehicles, People). The crash dataset has 48 features, Vehicles has 71, and People has 29. For this report, we consider records from Summer 2018 (Training) and Summer 2017 (Test).

1.2 - Visualization of Car Accidents in Chicago

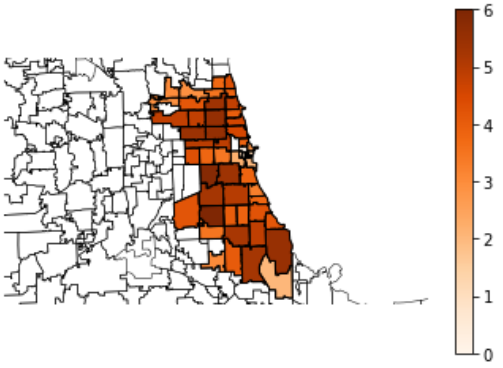
During the summer of 2018, there were approximately thirty thousand recorded car crashes in Chicago as reported by the Chicago Police Department (CPD). Recorded information about each crash ranged from data about the incident's occurrence, such as geolocation, to data about the driver, such as their age and sex. Mapping the geolocations with matplotlib's pyplot yielded an image similar to the shape of Chicago as can be seen here:



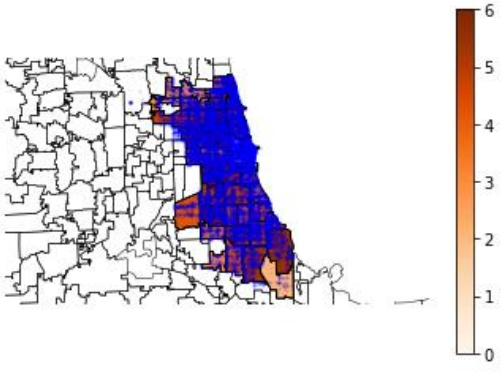
Not being able to discern much information from this image, though, we decided to plot each accident against an actual map of Chicago. This was accomplished using Basemap in Python along with associated helper files. Mapping the accidents on an actual map yielded much more informational results:



While this graph was helpful in visualizing our data, it was very intriguing how one location in the Southeast had very few crashes. Thus, an investigation into why that was the case took place. The population map of Chicago revealed that only a few individuals lived in that area, with a vast majority of them choosing to live along the center of Chicago, rather than its outskirts.



From the population density map shown above, the suspicion that most car accidents would occur throughout the densest areas of Chicago naturally came about, so we decided to map the accidents against the population densities, as shown in the following figure:



As can be seen, the outskirts of Chicago have fewer accidents, with most accidents being in the center of Chicago, where it is most dense in terms of population. This finding therefore corroborated our hypothesis.

1.3 - Analysis of Chicago Accidents

Prior to data mining work, it was also interesting to find that some obvious patterns in the data were confirmed when the data was graphed. But in other times, the answer did not seem as intuitive as originally thought. Using the Summer 2018 data, this is what we found, plotting posted speed limits against injury severities, which have been color coded as blue (no injuries), yellow (reported, yet not evident injuries), dark green (nonincapacitating injury), dark blue (incapacitating), and gold (fatal):

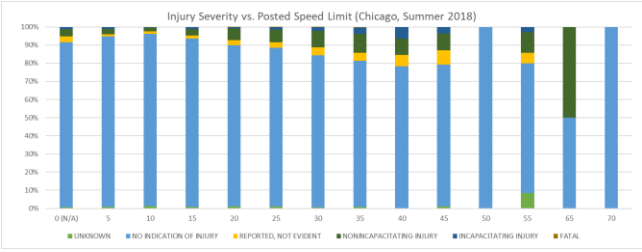


Figure 1: Injury Severity vs. Posted Speed Limit

In here, the intuitive thought that roads that have higher speed limits are involved in more severe crashes was confirmed. The x-axis in Figure 1 is the posted speed limit where a crash occurred, and the y-axis is the % occurrence of each accident severity. As can be seen, accident severities gradually get worse as the speed limit is increased. There are a few outliers when it came to higher speed limits, due to lack of data involving the same posted speed limits. As an example, only one crash happened when the speed limit was 70 during the summer of 2018, and fortunately, no serious injuries occurred. Due to this anomaly, however, a suggestion that roads that have a speed limit of 70 have a 100% survival rate may arise, which is a false conclusion.

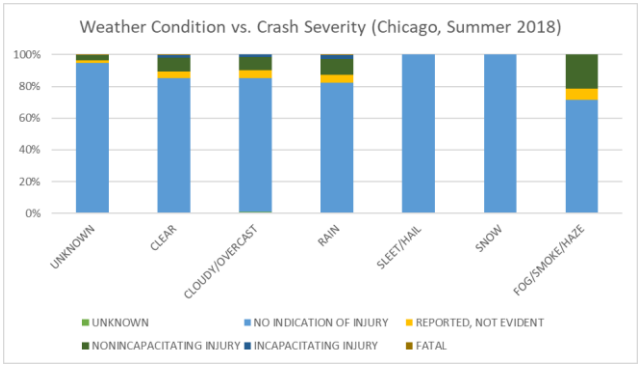


Figure 2: Weather Condition vs. Crash Severity

Another intuitive graph that was also generated involved the severity of the crash when it came to weather conditions (Fig. 2). As it can be seen, rainy days involved more injuries than clear/cloudy/overcast days. But what came in as very surprising is the amount of injuries and fatalities that occurred during foggy/smoky/hazy days, which far exceeds rainy days. This makes sense considering those days limit driver vision. This graph however may also lead to a few false conclusions, as one can draw that injuries never occur during days with snow or hail, but again, this was only due to the infrequency of data elements in that set, as snow only occurred 5 times in the 30,817 Summer 2018 crashes dataset, and sleet/hail only occurred 2 times. This may indicate that data has been improperly recorded, and it may also read to false conclusions.

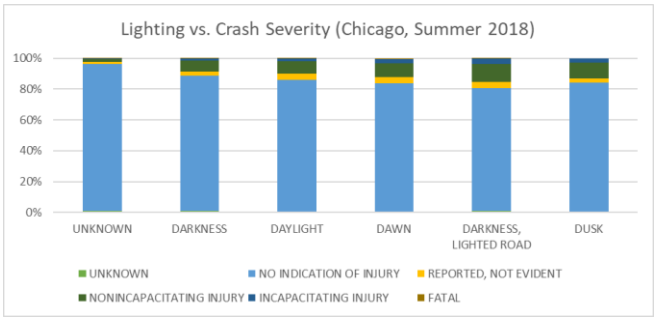


Figure 3: Lighting vs. Crash Severity

The last graph we examined during our mining analysis was one that involved the lighting conditions when the crash occurred. The results here were also consistent with natural biases, where crashes that occurred at night involved much more injuries than daylight. However, it was surprising to find out that in pitch darkness, without any streetlights to light up the path, crashes that involved injuries were less frequent when compared to crashes that occurred during daylight. A possible explanation for this phenomenon may be that streets with pitch darkness are not as crowded as streets where streetlights have been installed, and individuals at the time are much more cautious when driving.

2 Solution

2.1 - Data Preprocessing

Preprocessing was the most gruesome but crucial step in our process. Modifying the data set such that it keeps the same meaning and be interpreted by a data mining algorithm is a challenging task faced by many data scientists and statisticians.

2.1.1 – Selecting a Training Sample

The original City of Chicago car crash data set was divided into 3 separate datasets that involve more than 200,000 crash entries, and that date as far back as 2014. Those datasets contained information about the following:

1. The crash itself (date, weather, lighting, primary cause, …) [1]
2. The people involved (age, sex, injury classification, …) [2]
3. The vehicles involved in the crash (make, model, year, …) [3]

We have come to notice that vehicle data was largely sparse and infeasible to deal with and discretize without any general vehicle descriptions (i.e.: size of the vehicle). So, we agreed to only deal with information about the crashes and individuals involved instead.

Next. we have come to realize that different seasons often meant different reasons for crashes. For example, roads will be in much rougher conditions during winter months, and that information may not necessarily be captured by the datasets. We did not want these details to bias our results, so, we decided to only deal with summer data for this project, with dates ranging [6/21, 9/21]. Finally, when it came to the year, we decided to select 2018 as our training year to build the most accurate and up to date model that reflected current regulations and road conditions. We decided to use summer 2017 data for testing.

2.1.2 – Combining Crashes and People

With two datasets now, crashes and people, our next challenge was combining them into one. The two tables both featured a column called RD_NO, which was the report number of the crash. For each individual, the report number referred to the crash they were involved in, so multiple individuals may show up with the same report number. With that in mind, a left outer join was done from people onto crashes based on the report number column being equivalent. The Excel formula for accomplishing this feat was:

IFERROR(VLOOKUP($A2, crashes!$A:$AN,7,FALSE),"")

Which roughly translated to, if a report number for an individual existed in the crashes table, concatenate the crash details to the individual. Applying the formula to all individuals in the workbook created a table that had information about every individual and the details about the crash they were involved in.

2.1.3 – Discretization (binning)

With crashes and people now merged into one, allowing algorithms to operate on the data was next. For that to happen we had to transform each cell from a label {DARKNESS, DAWN, DAYLIGHT, DUSK} to a number {0, 1, 2, 3}. We decided to enumerate the values for each column, and doing so required a manual inspection of the data and recording of those unique values. Once unique strings were recorded, a mapping was created to allow for a translation from raw strings to numerical data.

Later on in our process, we felt the compulsion to also discretize age data, which was already numerical, and we did so by grouping it based on different age ranges{SMALL CHILD (3-6), YOUNG ADULT (18-21), etc..} then enumerating it as well {0, 1, …}.

2.1.5 – Feature Reduction

Numerous columns were found to have no use as they were either redundant, irrelevant, or represented a consequent that we're not aiming to determine. Examples of such columns included duplicated columns, report numbers, and if the individual was transported to an Emergency Room. As these columns hindered our training model, we simply eliminated them.

2.1.6 – Project Goal

All features by now were either antecedents or general details about the subject/environment. The one consequent that was kept was INJURY_CLASSIFICATION, which indicated the severity of the injury that was sustained by the individual, and it was one of:

0. NO INDICATION OF INJURY
0. ~~REPORTED INJURY, NO EVIDENCE~~
1. NONINCAPACITATING INJURY
2. INCAPACITATING INJURY
3. FATAL

Predicting INJURY_CLASSIFICATION became the goal for this investigation. Recently, however, we revisited the possible values for injury classification, and found that *REPORTED INJURY, NO EVIDENCE* was not a feasible class to predict for our purposes, and it only hindered our predictions. The goal of this investigation is to find when injuries do occur, not when people report them without any basis. We decided to indicate anyone classified as REPORTED INJURY, NO EVIDENCE to have NO INDICATION OF INJURY instead.

2.2 - Transformation

2.2.1 - Oversampling

When looking over the data, we noticed a large discrepancy between the number of non-injuries injuries, and fatal injuries. Analyzing this discrepancy showed that 91% of crashes didn't involve any injuries and only 0.05% involved a fatality (though we are thankful that that is the case).

To lessen the bias towards non-injuries, as well as preserve their details, we chose to oversample the data. Our dataset of 67,346 individuals with unbalanced injury classifications became one with 181,737 individuals, with repeated entries of underrepresented instances to balance all injury classifications.

2.2.2 – Refining Features

More feature trimming felt necessary as multiple features were still either infrequent or unimportant. To eliminate those remaining features, we used SKLearn's SelectKBest, scoring the features based on a chi squared statistic, and selecting the 20 best features

in the dataset. At the end of feature refinement, we had 181,737 individuals with 20 features each to determine injury classification

2.2.3 – Reverting to Strings

Our dataset consisted of oversampled numbers at this point, but since we were planning to use association analysis, we were required to revert to the original strings before the mapping. We did so by using Python Dictionaries, converting numbers back to their mapped strings. After that, we were ready to start training our model.

2.3 - Data Mining

2.3.1 – Weka

We settled on using the Weka software by the University of Waikato to continue with our investigation. WEKA provides various utilities that vastly help when it comes to association rule mining and classification.

2.3.2 – Association Analysis & Classification

Before dealing with classification, we were interested in what precursors led to fatalities. To accomplish this, Class Association Rules (CAR: subset of association rules with a specific class as a consequent) was set to True with the class index being set to 18 to indicate Injury Classification as the consequent (as a minor note, setting a specific consequent only outputs Confidence, so we did not display other metrics, such as Lift). The minimum confidence was set to 0.3, and the lower bound minimum support was set to .0001 (this meant that for an item to be considered frequent, it must have appeared in at least .01% of the data).

After running association analysis with these parameters on the training data, we came across a few interesting rules, such as:

SEX=M SAFETY_EQUIPMENT=SAFETY BELT NOT USED==> INJURY_CLASSIFICATION=FATAL conf:(0.9)

An interpretation of this rule is being a male while not wearing a safety belt (a seat belt) implies a fatality after a crash.

WEKA was then used for classification. With our dataset. we settled on using Naïve Bayes, Random Tree, and Random Forest.

2.3.3 – Naïve Bayes

Naïve Bayes is a basic probability-based classifier which assumes the independent nature of features. This independence assumption makes the classification fast. The advantage of using Naïve Bayes with is that it is linearly scalable with the number of features and can be used for multi class classification problem prediction.

Given that we have a multi class problem, we chose to see how Naïve Bayes would perform when it came to classification.

2.3.4 – Random Tree

Random Tree is a supervised classifier in which there are many individual learners deployed. Bagging is used to produce random sets which are used to construct a decision tree where each node is split on the best split among variables. The Random Tree algorithm is relatively quick and works well with many features. Considering our features, we knew Random Tree would be a good fit as our data, at a glance, fit like a tree meaning there were many different edges that led to different injury classifications.

2.3.5 – Random Forest

Random Forest chooses X number of trees to grow on different samples of data. Once all the trees are grown, an input gets assigned a label based off the majority vote of all the trees. Random Forest is an excellent classifier in terms of accuracy and handles large data quite easy however it takes some time to build the model and to classify. As with Random Tree, we expect Random Forest to perform well for the same reasons.

2.3.6 – Performance Metric

Given that our dataset has a large class imbalance between fatalities and non-injuries, we opted for F1 Score as our performance metric, as it takes into account the impact of False Positives and False Negatives.

2.3.7 – Classification on Training Data

We ran the above classifiers on the training data to see how well the classifier learned the model with a 10-fold split for testing on the training data with the label as Injury Classification.

Classifier	%Correct	%Incorrect	Precision	Recall	F1
Naïve Bayes	66.8	33.1	.66	.69	.66
Random Tree	96.9	3.1	.97	.97	.97
Random Forest	98.5	1.5	.99	.99	.99

Confusion matrixes for each classifier were as follows…

Naïve Bayes:

```
=== Confusion Matrix ===

     a     b     c     d   <-- classified as
 33452  8629  2434   919 |     a = NO INDICATION OF INJURY
 12919 19125 11543  1847 |     b = NONINCAPACITATING INJURY
  8186 10747 23406  3095 |     c = INCAPACITATING INJURY
     0     0     0 45434 |     d = FATAL
```

Random Tree:

```
=== Confusion Matrix ===

     a     b     c     d   <-- classified as
 40331  4045   979    79 |     a = NO INDICATION OF INJURY
    98 45089   247     0 |     b = NONINCAPACITATING INJURY
     0     0 45259   175 |     c = INCAPACITATING INJURY
     0     0     0 45434 |     d = FATAL
```

Random Forest:

```
=== Confusion Matrix ===

     a     b     c     d   <-- classified as
 43276  1936   215     7 |     a = NO INDICATION OF INJURY
    79 45108   247     0 |     b = NONINCAPACITATING INJURY
     0     0 45259   175 |     c = INCAPACITATING INJURY
     0     0     0 45434 |     d = FATAL
```

2.4 - Evaluation of Training Data

Out of the three classifiers, Naïve Bayes performed the worst in terms of accuracy and F1 Score, and Random Forest performed the best. Despite the F1 score for Naïve Bayes being the lowest, it managed to classify all fatalities correctly with no false negatives. The classifier looks as if it has learned the model exceptionally well as the accuracies for all three are above 50 percent however it is yet to be seen if there has been any sort of overfitting occurring.

3 Predicting Summer 2017 Labels

3.1 – Selecting a Test Sample

The classifiers performed relatively well on the training data, so we selected unseen data from Chicago's Summer 2017 to test our classifiers. This dataset contained 49,743 people entries.

3.2 – Matching Formats

For us to rerun our algorithms on the test set, the dataset had to go through a similar preprocessing phase to one that was described earlier. Summer 2017 People had to be merged with Crashes they were involved in again, and only the 20 best features were kept. It was unnecessary this time to discretize the data due to already being formatted for WEKA classification. The only column that was significantly altered was the age, where we followed the same age groups that were specified during training. By now, we were ready to measure how our trained Summer 2018 classifiers would perform against a never seen before, Summer 2017, test set.

3.3 – Prediction Results

The results of rerunning the same algorithms on the unseen test data were as follows:

Classifier	%Correct	%Incorrect	Precision	Recall	F1
Naïve Bayes	70.5	29.5	.94	.71	.79
Random Tree	88.8	11.2	.91	.89	.90
Random Forest	92.4	7.6	.92	.92	.92

3.3.1 – Confusion Matrices

Naïve Bayes:

```
=== Confusion Matrix ===

     a     b     c     d   <-- classified as
 33860 10539  2043   641 |     a = NO INDICATION OF INJURY
   460  1003   610    64 |     b = NONINCAPACITATING INJURY
    76   179   186    25 |     c = INCAPACITATING INJURY
     4     3    15     5 |     d = FATAL
```

Random Tree:

```
=== Confusion Matrix ===

     a     b     c     d   <-- classified as
 43679  2655   682    67 |     a = NO INDICATION OF INJURY
  1555   417   146    19 |     b = NONINCAPACITATING INJURY
   315   105    44     2 |     c = INCAPACITATING INJURY
    14     5     7     1 |     d = FATAL
```

Random Forest:

```
=== Confusion Matrix ===

     a     b     c     d   <-- classified as
 45347  1521   209     6 |     a = NO INDICATION OF INJURY
  1517   531    85     4 |     b = NONINCAPACITATING INJURY
   300   130    35     1 |     c = INCAPACITATING INJURY
    14     8     5     0 |     d = FATAL
```

3.4 – Evaluation of Prediction Results

Initially we were worried that the model may have overfitted the test dataset due to the very high accuracies that were achieved during training. However, that does not seem to be the case here as the classifiers performed well yet again. The accuracies and F1 scores on the unseen test dataset were relatively high which tells us that the model indeed did not overfit significantly. Naive Bayes performed the worst in terms of accuracy and F1 score, however, we found it to be our favorite model as it managed to predict most fatalities as injuries or worse. It also performed significantly better when it came to incapacitating and non-incapacitating injuries, being correct the majority of the time, unlike Random Tree/ Random Forest. So overall, when it comes to saving human life, we choose Naïve Bayes as our favorite model, as it has a high potential to mitigate crash injuries and deaths.

4 Conclusion

In conclusion, we have managed to address the problem we sought to investigate and managed to build a model out of 67,386 individuals involved in summer 2018 crashes and predicted the injury classification of 49,743 individuals from the summer of 2017. We were very surprised to see how much information the misfortune of only 36 fatalities was able to bring to a never seen before dataset of 49,743 people. The injuries and incapacitations helped significantly as well, and with this minimal model of only summer 2018 data, only 4 out of 27 summer 2017 fatalities were classified as no indication of injuries.

5 Contribution

We both feel that there was a successful exchange of responsibilities and validation in this project. Nabil contributed significantly to the lossless preprocessing phase and joining of tables. Hamza contributed significantly to running WEKA on both training and test data. Ideas were exchanged continuously regarding data selection, modifications to the data, and future steps/algorithms to pursue.

REFERENCES

[1] Traffic Crashes - Crashes | City of Chicago. (Updated Daily). Retrieved November 24, 2018, from https://data.cityofchicago.org/Transportation/Traffic-Crashes-Crashes/85ca-t3if

[2] Traffic Crashes - People | City of Chicago (Updated Daily). Retrieved November 24, 2018, from https://data.cityofchicago.org/Transportation/Traffic-Crashes-People/u6pd-qa9d

[3] Traffic Crashes - Vehicles | City of Chicago (Updated Daily). Retrieved November 24, 2018, from https://data.cityofchicago.org/Transportation/Traffic-Crashes-Vehicles/68nd-jvt3