Crash Risk Prediction Model using Data Science

This manuscript (permalink) was automatically generated from uiceds/cee-492-term-project-fall-2022-swifties@a1c4794 on November 21, 2022.

Authors

- **Lara Diab**
  0000-0001-8489-2015 · diablara
  Illinois Center for Transportation (ICT); Department of Civil and Environmental Engineering, University of Illinois Urbana-Champaign · Funded by The Grainger College of Engineering

- **Renan Santos Maia**
  0000-0002-0877-4006 · renanssmaia
  Illinois Center for Transportation (ICT); Department of Civil and Environmental Engineering, University of Illinois Urbana-Champaign · Funded by The Grainger College of Engineering

- **Farid Saud**
  XXXX-XXXX-XXXX-XXXX · fsaudm
  Department of Civil and Environmental Engineering, University of Illinois Urbana-Champaign · Funded by The Grainger College of Engineering

- **Johann J Cardenas Huaman**
  0000-0002-4695-7639 · Johann-Cardenas · transporter_pe
  Illinois Center for Transportation (ICT); Department of Civil and Environmental Engineering, University of Illinois Urbana-Champaign · Funded by The Grainger College of Engineering

Project selection and Introduction

Road accidents are responsible for a significant number of injuries reported every year. According to the World Health Organization (WHO), approximately 1.3 million people die each year as a result of road traffic crashes (as of June, 2022). In addition, road traffic crashes cost countries 3% of their gross domestic product (Safarpour et al, 2020; WHO, 2022). Consequently, understanding what influences these accidents on roads is of utmost importance. However, it is not easy to decide which exact conditions lead to these accidents. Different road, climate, vehicle and driver conditions affect the likelihood of a driver to be in a fatal/serious accident.

The ability of predicting in an accurate way the potential occurence of car crashes is a valuable contribution for road safety. In an approach frequently used in the literature, crash records' data are used for the development of crash prediction models, so that agencies can allocate investments to priority areas of the roadway network. However, given that the budget for infrastructure improvements is limited, adopting countermeasures for all facilities that crashes are potentially occuring is not financially feasible. Therefore, informing drivers about the potential safety risks is a way to proactively compensate the aforementioned limitations. Moreover, with the development of connected and autonomous vehicles, this information can be provided in a more optimized way, contributing for vehicles' route decision, as well as for real-time alerts that can lead drivers to take the necessary precautions to operate more safely (Yu et al, 2021).

Project Objective and Plan Proposal

The objective of this work is to use the Illinois Department of Transportation (IDOT) extensive crash data to be analyzed and, finally, be used for a crash risk prediction model based on main categorical data that can be real-time updated (such as the weather/lighting/pavement conditions). Ideally, it could be used by navigation systems to allert drivers to adopt more cautious behavior as soon as they enter higher-risk sections.

The plan to be carried out will follow the basic steps described as follows:

1. Read the IDOT's crash data CSV files available as an open data source.
2. Clean the data by deleting unwanted columns, handling missing data, and removing irrelevant observations.
3. Tidy the data by organizing the variables into columns and the observations into rows.
4. Analyze and visualize the data by finding correlations both analytically and graphically.
5. Model a prediction algorithm for crash risk according to categorical variables.

Description of the Data Set

IDOT has generated datasets with statewide crash locations produced by the Crash Information Section of the Illinois Department of Transportation (IDOT). The accident data has been collected throughout the years using Application Programming Interfaces (APIs) that provided streaming traffic incident data. There are about 300,000 accident records per year in these datasets, and each record contains attributes that include conditions like (among others that are not listed or described here because these are not relevant for this study):

1. Time and date (day, month, year)
2. Coordinates (x,y)
3. Type of collision
4. A quantitative description of fatalities and injuries

5. Crash severity classification based on their impact on traffic
6. The road surface condition (“Dry”, “Wet”, “Snow or slush”, “Ice”, or “Sand/Dirt/Mud”)
7. Road defects (“Debris on roadway”, “Rut/Holes”, “Unkown”, or “No defects”)
8. Lightning conditions (rated in a scale from 1 to 9)
9. Geometric characteristics of the road section
10. Work Zone (“construction”, “maintenance”, “utility”, “unknown”, or “N/A”)
11. Possible causes of the accident.

The datasets for different years are available for download as .CSV files at the IDOT’s website:

https://gis-idot.opendata.arcgis.com/search?groupIds=6d2862031a6d47c7a8c211e38e423e05

Exploratory Data Analysis

Open-source crash data is published by the Illinois Department of Transportation (IDOT) yearly. Each crash report was found to have extensive entries with up to 85 attributes, which include several independent variables to describe each crash occurence. Each dataset is organized with observations filled out according to the IDOT Traffic Crash Report SR 1050 Instruction Manual (2019). The dataset for each year is available online in .CSV format at the IDOT website, and they contain observations arranged in rows and attributes in columns. The datasets from 2017, 2018, and 2019 were included in this Exploratory Data Analysis. The datasets from 2020 and 2021 were discarded in this analysis due to the COVID-19 pandemic outbreak, which altered drastically the dynamics of traffic worldwide, and thus crash-related data (Yasin, Grivna & Abu-Zidan, 2021). Regarding the dataset size, each one had originally over 300,000 rows (944,328 in total, combined). The Exploratory Data Analysis will be carried out following the steps described in the next sections.

Reading the Data

The datasets were imported to Visual Studio Code using the CSV library. It was found that the latest report contained 5 additional attributes that could not be used since they were missing in previous reports. It was verified that any other attribute was arranged in the same way for each file, thus it was decided to discard this additional information. After deleting these attributes, all 03 datasets were merged into a unified file, which contains 80 variables (columns) and 944,328 observations (rows). This final database served as the baseline to start the cleaning process.

Cleaning Process

It was found that several independent variables would not provide fruitful information due to missing, unknown or incomplete data. First, this observation was obtained by visual inspection, and later by analyzing the number of different and unique values present in each attribute. Thus, the datasets were processed to filter out irrelevant or incomplete variables. For instance, information pertaining the location (latitude & longitude, or X & Y coordinates) has not been taken into account. A map plot was initially produced to see the distribution of the data. Columns containing codes describing the city, county or ID of the location where the crash took place have also been excluded. Columns involving duplicate information (e.g. two columns describing the same independent variable with a label and a number), and traffic structures were also removed. For few other independent variables, information that could potentially be useful was found to be significantly incomplete. For instance, this was the case of attributes such as the number of lanes and the type of intersection. As a consequence, these variables were not included in the clean dataset. Finally, additional cleaning was carried out for independent variables with a high number of description labels. For example, the “Railroad Crossing Number” variable had up to 100 different values which would have not been handy information for the end-user. After filtering out all the attributes that would not be utilized for this analysis, the number of independent variables went down from 80 to 21.

When it comes to crash reports, several inconsistencies are considerably frequent. In the literature, for example, it is mentioned that “investigation of traffic safety by means of crash records is a reactive approach, where researchers need to deal with imprecise, incomplete, inconsistent, and, sometimes, inexistent records”, and that is why the acquisition of historical series to provide minimal consistency to the analysis of crashes to reduce misinterpretations and misleading conclusions is crucial (Hauer & Hakkert, 1989; Chin & Quek, 1997; Farmer, 2003). This justifies the need of dedicating a considerable amount of time, after filtering the columns (variables) of interest, to the cleaning process of the rows (observations). For each column, the observations labeled as “blank”, “unknown”, and “other” were a

matter of discussion among the group on how these inconsistencies would be handled. For all variables, the “blank” observations were immediately removed from the dataset.

Analysis and Visualization

In this section, a set of plots, charts, and visuals were developed using a set of tools: Julia, Python, and Power BI, and are presented to display the findings and trends, product of the exploratory data analysis. The visualization include distributions of car crashes over time and for the identified variables discussed in this report.

Bar Plot Crashes

Bar Plot

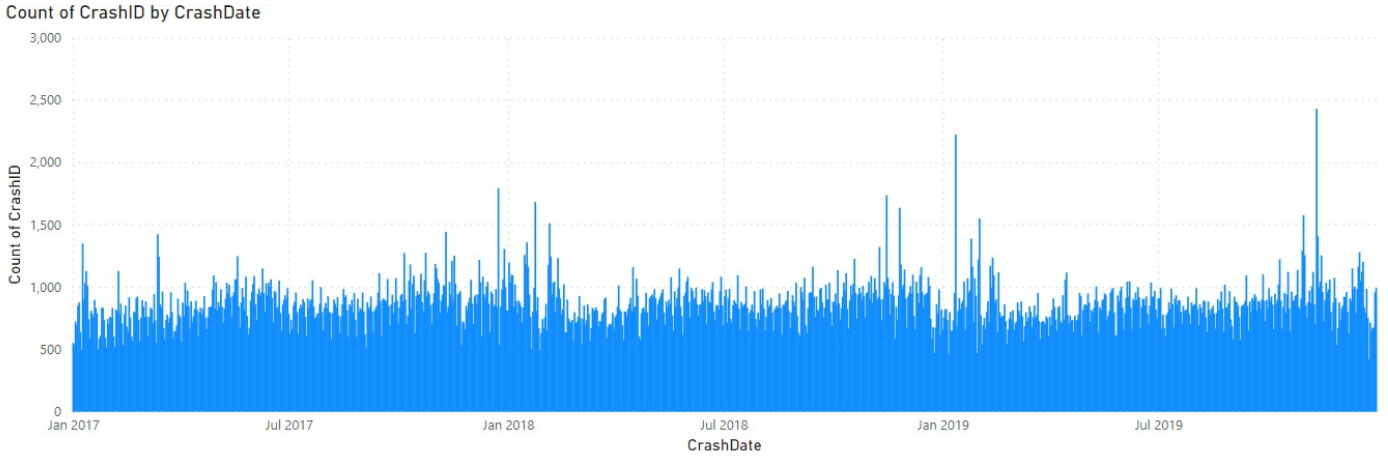

Figure 1: Distribution of accidents over time. From 2017 to 2019.

The first plot presented is a visual representation of the number of car crashes throughout time. This served as a foundational step for the later-developed visuals. Below, the data is subdivided by year, and then by month to better understand the number of car crashes through time.

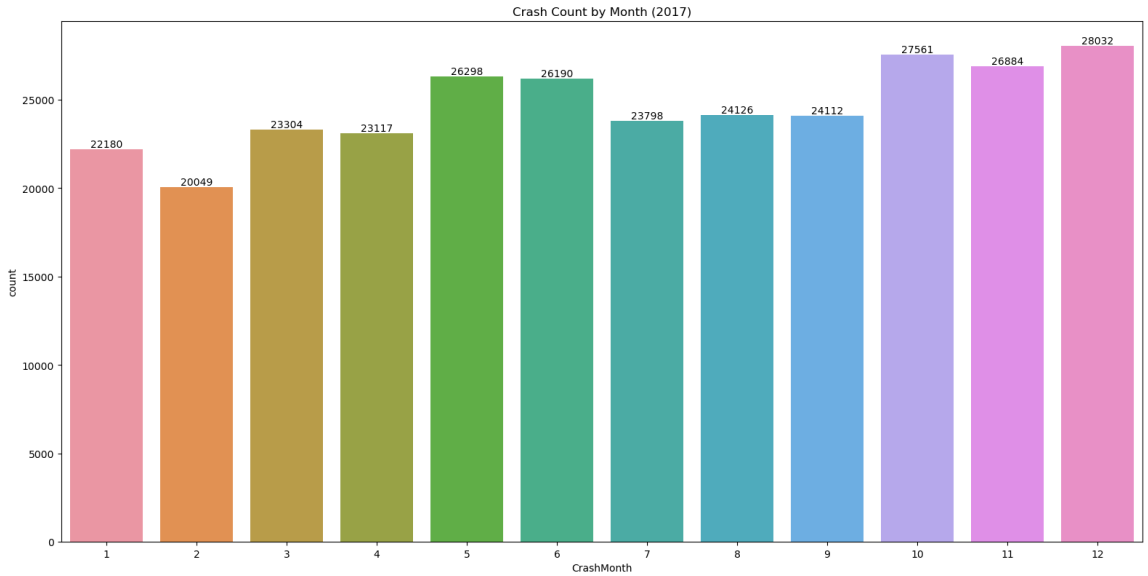

Figure 2: Bar Plot Historical Crashes. Crash reports from 2017 in the state of Illinois, USA.

For 2017, the month with the most incidents is December (“12”), with 28032 car crashes (out of 295651 for that year), and the month with the least car crashes is February (“2”), with 20049 incidents that month. These findings can represent the effect of weather conditions as well, given that during the very first and the later years of the year, the amount of accidents increases. These months correspond to the winter season and englobes certain holidays where people might be very active and potentially more car crashes might happen.

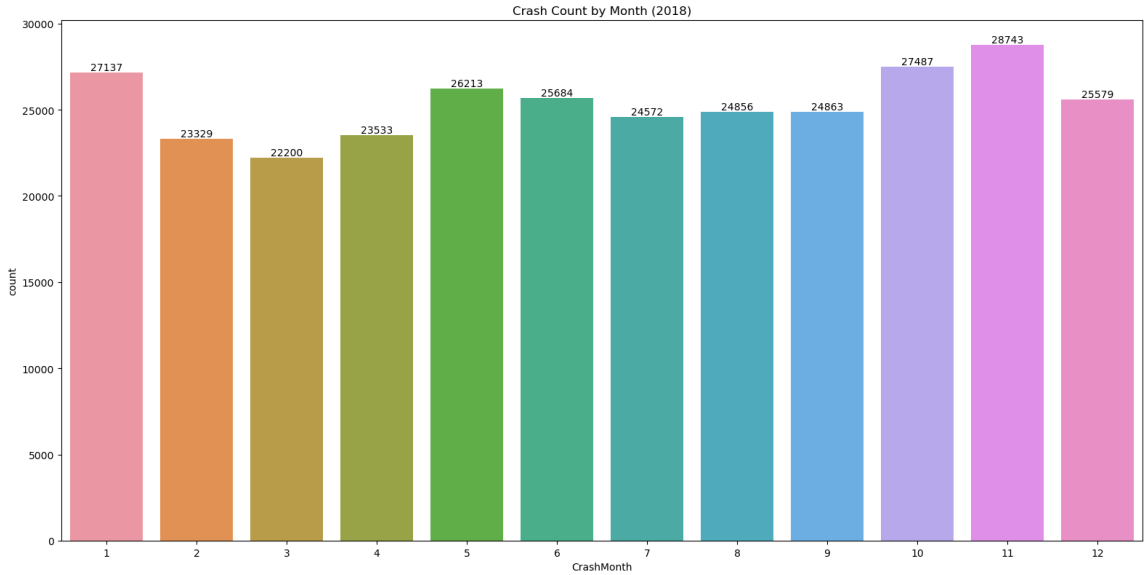

Figure 3: Bar Plot Historical Crashes. Crash reports from 2018 in the state of Illinois, USA.

Now, for 2018, different from the previous year, the month with the largest count of accidents is November (“11”), with 28743 recorded crashes (out of 304196). The month with the least amount of crashes is March (“3”), with 22200 crashes. For this year, just like for 2017, the months where peaks happen can be associated with the worse seasons for a driver.

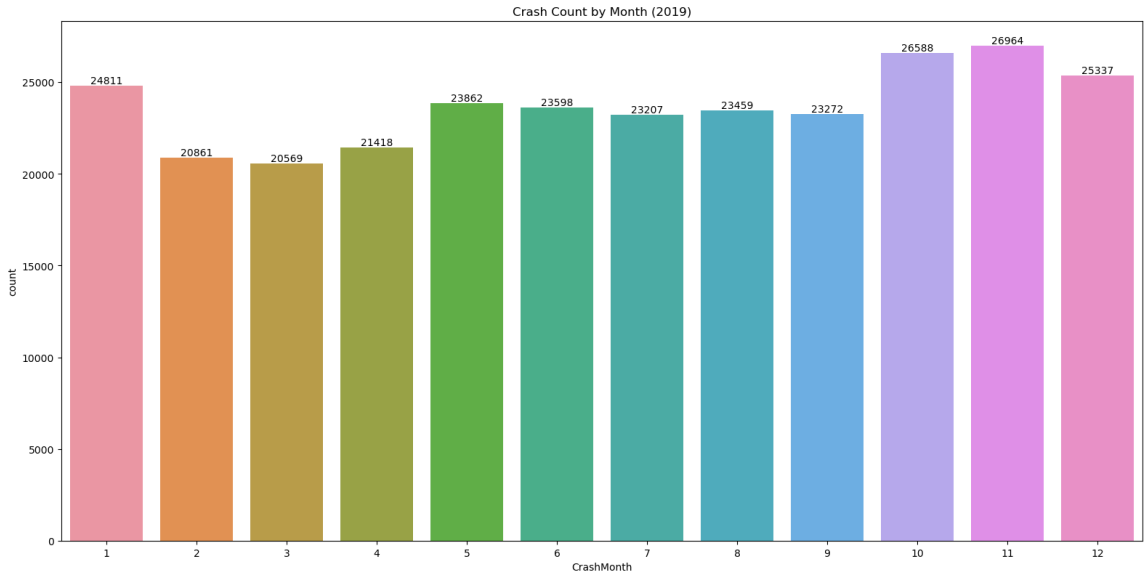

Figure 4: Bar Plot Historical Crashes. Crash reports from 2019 in the state of Illinois, USA.

For the year 2019, the month with the most accidents is November (“11”), with 26964 recorded crashes (out of 283946), and the month with the least accidents is March (“3”), with 20569 incidents. in a similar way to the years 2017 and 2018, the months with the highest count happen in the same season.

Map

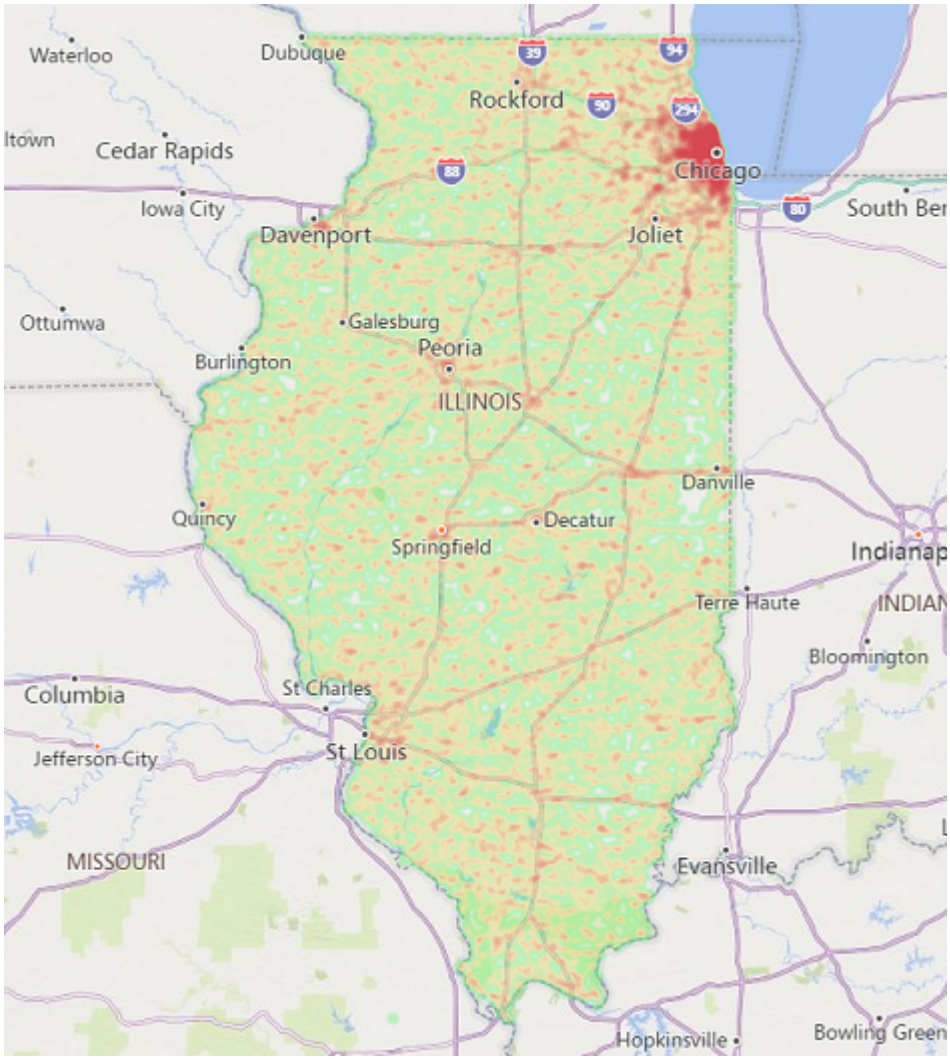

Figure 5: Distribution of crash occurrences. Crash reports from 2017,2018,2019 in the state of Illinois, USA.

Using the “X” and “Y” information present in the dataset, a visual distribution of the location of the recorded car crashes is illustrated in the map Figure. In the map of Illinois, the red dots represent a pair of “X” and “Y” coordinates, being the location where a car crash happened. The majority of car crashes appear to have happened in cities and towns. The best example would be Chicago, where all around the area, the number of occurrences, or red dots, is significantly higher, which may be presumable given that Chicago and the surrounding areas host a big fraction of the state’s population. On a smaller scale, this is also visible in other cities and towns, such as Springfield and Davenport. In other locations, the recorded car crashes are much more spaced out, with certain “hotspots” in some highways and roads.

Pie Chart | Lightning Conditions

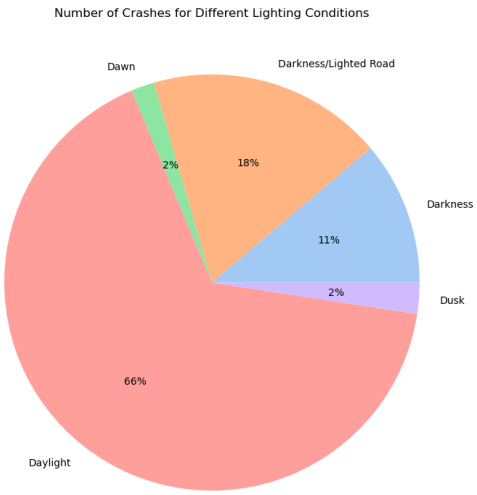

Figure 6: Distribution of accidents by Lightning Condition. From 2017 to 2019.

This figure displays the different percentages of the different lighting conditions presented in the dataset. The condition with the most car crashes associated with it is “Daylight”, representing 66% of the crashes in the dataset. During the nighttime, “Darkness/Lighted Road” accounts for 18% of the crashes, and “Darkness”, for 11%. Conditions “Dawn” and “Dusk” account for 2% each of the car crashes recorded.

Pie Chart |Road Surface Condition

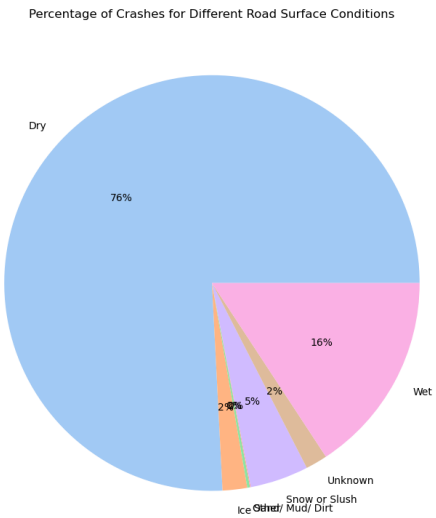

Figure 7: Distribution of accidents by Road Surface Condition. From 2017 to 2019.

The chart above summarizes the analysis performed on the data on the influence of road surface conditions on the number of car crashes. It was found that 76% of the recorded crashes correspond to a “dry” road surface, which can be thought of as the least dangerous condition. For the not-too-favorable road surface conditions, 16% of the car crashes analyzed correspond to a “wet” road surface, 5% to “snow”, 2% to “ice” and other “unknown” road surface conditions.

Pie Chart | Road Defects

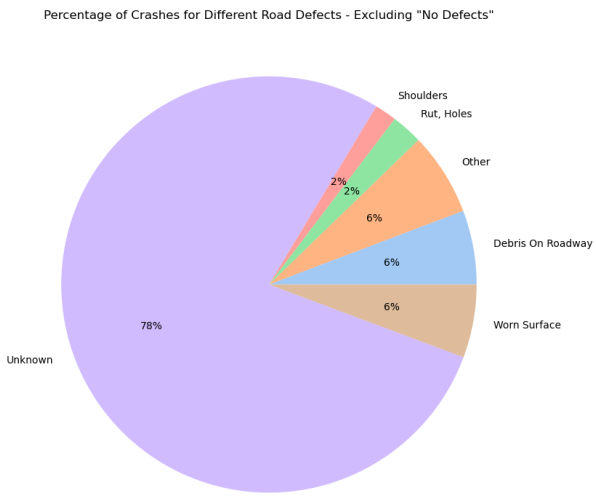

Figure 8: Distribution of accidents by Road Defects, excluding No Defects condition

While analyzing the distribution of accidents by road defects, it was found that the vast majority of the incidents (808,835 out of 883,793 observations, which accounts for 91.52%) happened were “no defects” were present in the location. Given that a great number of crashes happened without any road defects, it might be interesting to account for the likelihood of road defects being associated with a car crash. Figure 6 displays the different percentages of the road defects accounted for in the dataset, where “unknown” represents 78% of the data, followed by 6% for “worn surfaces”, “debris on the roadway” and “other” road defects, and 2% for “ruts and holes” and “shoulder” defects.

Correlation between Variables

As mentioned earlier, one of the objectives of analyzing this data is understanding how different road and environment conditions would affect crashes and their severity. This can be obtained by finding the associations between the different variables (columns) in the dataset, meaning how one variable is affected another variable. However, most of the variables are of categorical type, i.e., variables that are identified based on names or labels given to them and not based on numbers. This makes the built-in correlation functions in Python or Julia not helpful. One very commonly used method to measure the correlation between two categorical variables is Cramer’s V statistic. Cramer’s V is based on a nominal variation of Pearson’s Chi-Square Test. Like correlation, the output takes values between 0 and 1 (inclusive), with 0 corresponding to no association between the variables and 1 corresponding to one variable being completely determined by the other. On the other hand, and unlike the usual correlation, there are no negative values. For this project, a function was created in Python that calculates the association between any 2 categorical columns using confusion matrix which can be obtained via built-in pandas method for categorical columns. For this data that has 24 columns, running this function for every pair of variables would take a long time and may not give many insights. Therefore, the function was used to find how the column “CrashSeverity” is correlated with every other column. This column was chosen because finding how different conditions affect the severity of the crash is one of the most important outcomes of studying this dataset, and this would give an idea about the variables that have a significant impact on the crashes.

The output is described in the table below:

	Association with CrashSeverity
CrashYr	0.00807
CrashMonth	0.0268889
CrashDay	0.00470057
NumberOfVehicles	0.101171
DayOfWeekCode	0.0110936
CrashHour	0.0255177
CollisionTypeCode	0.259482
TotalFatals	0.707104
TotalInjured	0.705812
NoInjuries	0.355953
CrashSeverity	1
IntersectionRelated	0.141338
RoadwayFunctionalClassCode	0.0714974
WorkZoneRelated	0.00473353
TypeOfFirstCrash	0.259498
CityName	0.0738726
ClassOfTrafficWay	0.0608374
Cause1	0.165848
TrafficControlDevice	0.097696
TrafficControlDeviceCond	0.0515391
RoadSurfaceCond	0.0329927
RoadDefects	0.0424504
LightingCond	0.02852
WeatherCond	0.0307212

From the table above, it can be observed that the the factor that is the most correlated to the crash severity is the “TotalFatals” column which indicates the number of fatalities for each crash, with a value of 0.707104. Similar observations can be made for the number of injuries. This makes sense because it is expected that the higher the severity of the crash, the higher the number of fatalities and injuries is expected to be. However, this is not very helpful for understanding how different conditions affect the severity of the crash. For this purpose, the columns that can be compared are: “IntersectionRelated”, “RoadwayFunctionClassCode”, “WorkZoneRelated”, “ClassOfTrafficWay”, “TrafficControlDevice”, “TrafficControlDeviceCond”, “RoadSurfaceCond”, “RoadDefects”, “LightingCond” and “WeatherCond”. Comapring these, it can be seen that presence of intersections has the highest correlation with the severity of the crash followed by the traffic control device. In addition, the characteristics of the workzone seem to have the least correlation with the severity of the crash. This observation can be useful to understand the dataset and get an idea about which variables are important for making predictions. This was done to get a general idea about the data but for performing predictions, this will be handled in detail.

Trends

Since not all the variables have to be present for an accident to occur, it can be seen that most accidents happen in the absence of adverse conditions. However, we should take into account that this reflects the fact that adverse conditions are exceptions, and accidents happen on a daily basis with other factors as underlying reasons such as human behavior. However, adverse conditions do increase the likelihood of accidents and it can be observed an increase in the overall number of occurrences in specific hours (evening), days (weekends), and months (winter).The road type is also found to be directly correlated with the maximum speed limit, and as a consequence is tied to the number of accidents per day.

Potential Issues

As long as the number of entries containing a value for an independent variable overcome by large any other value for the same independent variable, we may experience problems related to “imbalanced data” due to the uneven distribution of observations. Similarly, it can be seen that most of the independent variables are “classifications”, and therefore their entries don’t provide meaningul numerical values to be analyzed or correlated. For some of them we could replace the text values by boolean variables, but for some others a rating system may be needed if a numerical interpretation is required.

It can be noticed that most of the attributes are subjective observations trying to describe the potential causes of an accident, and may be dependent on the observer itself. However, the casualties are a meaningful numerical observation that will be thoroughly used througout this report.

Predictive Model

Given the nature of the database and the primary established goal of predicting the severity of crashes according to different combinations of scenarios, a classification problem is faced, for which the following models will be tested:

- a. Decision Tree
- b. Random Forest
- c. Convolutional Neural Network (CNN)

The dataset in this study is large enough (with hundreds of thousands of entries), so an advantage could be taken from this by dividing it into training and validation datasets. Another option derives from the fact that the IDOT's crash databases from different years are also freely available online on .csv format. Therefore, additional hundreds of thousands of entries could be used for validation (e.g. the data from the years immediately before the ones selected for building the dataset of this report, such as the 2016 dataset). This way, the full dataset could be used to train and build the model, and the validation datasets will afterwards be useful to decide on the appropriate model parameters needed to achieve the optimal model accuracy.

Since machine learning algorithms are generally unable to work with categorical data when fed directly into the model, there is a need to convert our independent variables (inputs):RoadSurfaceCond,:RoadDefects,:LightingCond and :WeatherCond into numbers, and the same will be required for our output variable since it will also be categorical (:CrashSeverity). The task of assigning numerical values to make use of them has to be handled with the aim of avoiding undesired biases in the assignment process. If we assigned a float or a integer value, our machine learning model may wrongly allocate a higher weight to variables with higher values, affecting the accuracy of the prediction model. To avoid this issue, we will encode our categorical features as one-hot numeric arrays, a technique that is presented hereafter.

One Hot Encoding

The one-hot encoding scheme, also known as 'one-of-K' or 'dummy' creates a binary colum for each category, and returns a sparse matrix or dense array (depending on the sparse parameters).Our inputs to this transform will be strings, denoting the values taken on by our categorical (discrete) features and our output will be a binary feature for each possible category with the value of 1 to the feature of each sample that belongs to the category, and a value of 0 for any other feature (Buitinck et al, 2013). An example is shown below for the variable *:LightingCond*:

Original Feature	**One-Hot Encoded Feature**
Darkness	[1, 0, 0, 0, 0, 0]
Darkness/Lighted	[0, 1, 0, 0, 0, 0]
Dawn	[0, 0, 1, 0, 0, 0]
Daylight	[0, 0, 0, 1, 0, 0]
Dusk	[0, 0, 0, 0, 1, 0]
Unknown	[0, 0, 0, 0, 0, 1]

Although, the One Hot Encoding technique will be useful to transform and preprocess our data so that our model can understand it better and learn from it more effectively, it comes with its own

advantages and disadvantages. We'll be analyzing the results as we test our model to report our findings.

Once the data has been transformed, we can use it to train our proposed models. The description, adequacy, results, and conclusions from every model tested is discussed in the following sections.

a) Decision Tree Model

The first approach will be the development of a Decision Tree (DT) scheme. Decision Trees are non-parametric supervised learning methods, that can deal with large datasets without imposing complicated parametric structures, enabling them to predict the value of a target variable based on simple decision rules inferred from the data features. The objective is to find a set of decision rules that naturally partition the feature space to provide an informative and robust hierarchical classification model (Myles et al, 2004).

The DecisionTree.jl package available for Julia, provided us with a "DecisionTreeClassifier" model. This model requires the following hyperparameters: - pruning_purity_threshold: (post-pruning) merge leaves having >=thresh combined purity (default: no pruning) - max_depth: maximum depth of the decision tree (default: no maximum) - min_samples_leaf: the minimum number of samples each leaf needs to have (default: 1) - min_samples_split: the minimum number of samples in needed for a split (default: 2) - min_purity_increase: minimum purity needed for a split (default: 0.0) - n_subfeatures: number of features to select at random (default: keep all) - rng: the random number generator to use. Can be an Int, which will be used to seed and create a new random number generator. As the first approach, the values by default have been kept constant while the maximum depth of the decision tree is altered from scenario to scenario to evaluate the changes in the result. The first proposed model (Decision Tree) resulted in limited accuracy (78%) on the training data using the fully cleaned combined dataset. As a first estimate, this value could be seen as promising, however several issues could be immediately noticed. Despite testing several model parameters, the accuracy of the model didn't improve at all what indicates a problem in our dataset. Let's look at the general structure of our decision tree to understand this.

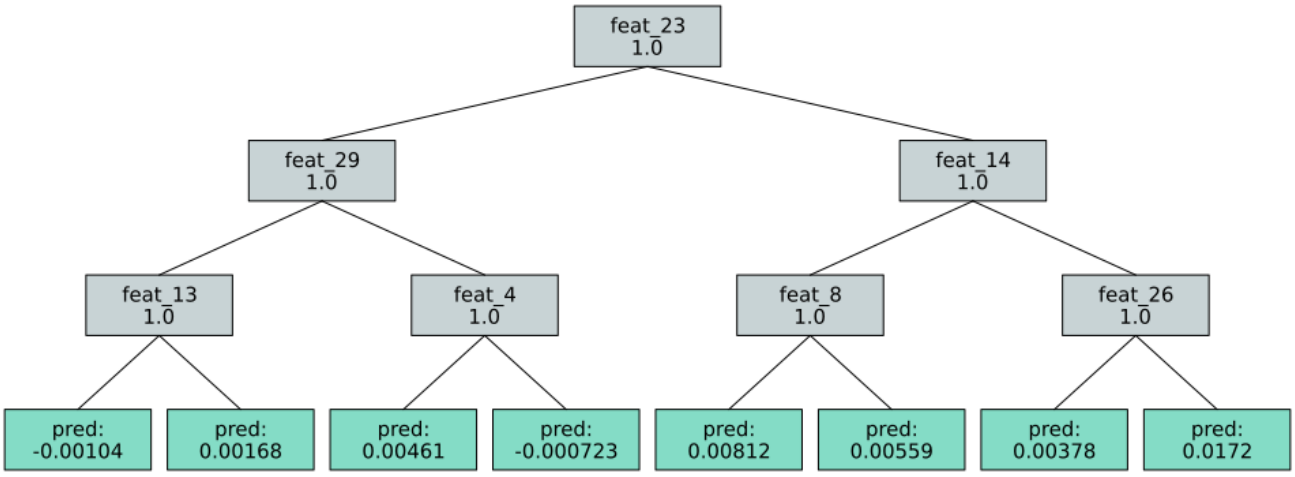

Figure 9: Decision Tree Structure. Using DecisionTreeClassifier.jl.

We could summarize this issue by stating that one of the possible outputs ("Property Damage"), the lowest severity crash type, dominates the dataset, accounting for almost 78% of the total number of cases. Consequently, a simple model that only predicts "Property Damage", independently of the entries, would have 78% accuracy. This imposes a huge bias in the criteria being used by the model to predict an output. As seen in the previous figure, most of the leaves of the decision tree get the right

output just because the likelihood of predicting one of them is enormously higher than the likelihood of any of the other two.

Alternative strategies were brainstormed to overcome the imbalanced data issue. Firstly, the DT models were re-trained using a database comprised of a sample of equal numbers of observations for the 3 different classes in the expected output vector. As this test didn't bring a considerable change either, a Random Forest (RF) model was also run for both datasets (full and reduced, equally sampled).

b) Random Forest

The Random Forest (RF) models are used to predict both categorical and continuous outputs. The background of the RF concept recalls for the presence of multiple classification trees, which partition the data using a sequence of binary splits on individual variables. The non-split nodes are called terminal nodes. Given the presence of multiple DTs, the RF models using the bagging method to build decision trees as parallel estimators, which are finally averaged to give rise to the mean predictive model. It should be noted that improved RF estimations can be obtained by taking into account uncorrelated and different between each other DTs, otherwise the final accuracy of RF and DT would be similar. In Julia, the so-called "RandomForestClassifier" object can be used to build a RF model.

For the purpose of the model definition, the Julia "DecisionTreeClassifier" model was used for the implementation of the DT. Given the limitations of the obtained DT results, a Random Forest (RF) model was implemented in order to evaluate if any increase in the model accuracy could be obtained.

c) Different Approaches for the Decision Tree and Random Forest Models

After achieving an accuracy of 78% for the decision trees and random forest models that were tried, different approaches were implemented to modify the data with the goal of achieving higher accuracy. These approaches are described below:

1- Removing rows with entries that are not very well understood

As mentioned earlier, it was decided only include the columns "RoadSurfaceCond", "RoadDefects", "LightingCond" and "WeatherCond". Each of these columns (attributes) has different possible values. Table 3 below shows the unique values for each attribute. For example, and as can be seen in the table, WeatherCond has 12 different possible values and for the one-hot encoding, this would create 12 columns just for the attribute "WeatherCond". It was thought that reducing the number of the one-hot encoded columns would make the accuracy better as the model would not have as many columns to use to make predict the labels. therefore, it was decided to delete the rows that have "WeatherCond", "RoadSurfaceCond" and "LightingCond" as "Other" or "Unknown". These two values were chosen to be removed because they do not offer any significant or specific information about the corresponding attributes.

RoadSurfaceCond	RoadDefects	LightingCond	WeatherCond
Wet	Debris on Roadway	Darkness	Blowing Sand
Dry	No Defects	Darkness/Lighted Road	Blowing Snow
Ice	Other	Dawn	Clear
Snow or Slush	Rut Holes	Daylight	Cloudy/Overcast

RoadSurfaceCond	RoadDefects	LightingCond	WeatherCond
Unknown	Shoulders	Dusk	Fog/Smoke/Haze
Other	Unknown		Freezing Rain
Sand/Mud/Dirt	Worn Surface		Other
			Rain
			Severe Cross Wind
			Sleet/Hail
			Snow
			Unknown

Table 3: Different values for each of the attributes chosen for the predictive model

After removing these rows, performing one-hot encoding again and running the decision tree/random forest models again, the same level of accuracy of 78% was achieved. This indicated that the higher number of one-hot encoded columns is not impacting the prediction.

2- Taking into consideration another attribute

It was also decided to try considering one more column in the dataset and that could maybe increase the accuracy. The reason behind this reasoning is maybe if the model was offered more insight about the data, other than the surface road conditions, road defects, weather conditions and lighting conditions, this would help the model notice more relationships between the features and the dependent variable and thus increasing the accuracy. Therefore, and based on the previous analysis of correlations between the columns, it was found that the presence or absence of road intersections was relatively highly correlated with crash severity, it was re-considered as one of the features for modelling. This attribute has 2 values: Yes or No. One-hot encoding was performed again and the decision tree model was applied again. However, the accuracy was also 78%. This showed that increasing the number of attributes is not the solution to low accuracy, in this case.

3- Addressing the issue of imbalanced data

Because the problem was not the presence of several values for each column, dealing with the imbalanced dataset was another approach that was implemented by the team. This model is trying to predict the crash severity level which has three label options: “Property Damage”, “Injury” and “Fatal”. “Property Damage” constitutes 80% of the data; 627706 rows out 809824 total rows (after cleaning the dataset), This imbalance is causing the model to predict “Propert Damage” most of the time, decreasing the accuracy of the model. One way of addressing this is running the model with the same number of rows for each label. “Fatal” had the lowest number of rows in the data which was around 3,000. Therefore, the same number of rows was randomly selected from each of the other two labels: Property Damage and Injury. However, this decreased the accuracy of te decision tree and random forest to around 40%. The reason for this may be referred to the low number of rows that the dataset was modified to have as compared to the initial data, which, although was imbalanced, had much more rows and that gave a better accuracy.

4- Removing “perfect” conditions

Another way to look at the data imbalance of this dataset is by observing the features rather than the dependent variables. Part of the reason why the data is imbalanced is the fact that most of the time

when the crashes happen, the coditions are “perfect”, meaning the weather condition is clear, the road has no defcts, the lighting condition is daylight and the road surface condition is dry. Therefore, one approach taken by the team was to remove those conditions and try to run the decision tree model for the remaining cases. This increased the model accuracy, at best, to 85%

d) Convolutional Neural Network (CNN)

As the first approach, the values by default have been kept constant while the maximum depth of the decision tree is altered from scenario to scenario to evaluate the changes in the result.

b) Random Forest

For the purpose of the model definition, the Julia “DecisionTreeClassifier” model was used for the implementation of the DT. Given the limitations of the obtained DT results, a Random Forest (RF) model was implemented in order to evaluate if any increase in the model accuracy could be obtained.

c) Convolutional Neural Network (CNN)

As it will be further discussed later, the first proposed model (Decision Tree) resulted in a limited accuracy (78%) on the training data using the full cleaned combined dataset. As a first estimation, this value could be seen as promising, however several issues could be immediately realized. In summary, “property damage”, the lowest severity crash type, dominates the dataset (also approximately 78% of the cases). Therefore, a simple model that only predicts “property damage”, independently of the entries, would have 78% accuracy. Alternative strategies were brainstormed in order to overcome the imbalanced data issue. Firstly, the DT models were re-trained using a database comprised of a sample of equal numbers of observations for the 3 different classes in the expected output vector. Other than that, the Random Forest (RF) approach was also ran for both datasets (full and reduced, equally sampled).

The RF models are used to predict both categorical and continuous outputs. The background of the RF concept recalls for the presence of multiple classification trees, which partition the data using a sequence of binary splits on individual variables. The non-split nodes are called terminal nodes. Given the presence of multiple DTs, the RF models using the bagging method to build decision trees as parallel estimators, which are finally averaged to give rise to the mean predictive model. It should be noted that improved RF estimations can be obtained by taking into account uncorrelated and different between each other DTs, otherwise the final accuracy of RF and DT would be similar. In Julia, the so-called “RandomForestClassifier” object can be used to build a RF model.

According to Brunton and Kutz, practitioners would generally refer to Deep Convolutional Neural Networks when thinking of Neural Networks. Given its popularity, flexibility and effectiveness for classification problems, DCNN were used to model this problem.

The Deep Convolutional Neural Network was constructed using the Julia Machine Learning Library, Flux. This library allows for an easier construction of neural networks with predefined functions for different kinds of layers, activation functions, model training functions, and several other functions of interest when building up a neural network.

For this model, as a first attempt, the entire dataset was used. An important consideration to account for when using the Flux library is that the neural network expects a very specific structure of the data: a 3 dimensional matrix of the input data that contains the observations divided into vectors, and one-hot encoded variables.

During the training phase, the model included convolutional layers, dense layers (to reduce the dimensionality of the ouput), the “ReLU” activation function (one of the more popular activation functions), pooling layers (for a more efficient computation and reduce overfitting), a dropout layer (to specifically target overfitting), and a “Softmax” function (a generalized version of the functions used for classification problems). These functions where all put together using a “Chain” function. In the testing phase, the Dropout functions is disabled.

The architecture of the Neural Network used is as follows:

model =Chain(Conv((3,), 1=>2, pad=(1,)), MaxPool((2,)), Conv((3,), 2=>4, pad=(1,)), MaxPool((2,)), x -> reshape(x, :, size(x, 3)), Dropout(1), Dense(28,3), softmax)

The error metric used for training this Neural Network was the logistical cross entropy loss function. This function matches the “Softmax” function output, and works as a generalized version of cross entropy loss functions used for “more-than-two” classes classification problems.

The hyper parameters used to train the neural network consisted of a batch size of approximately one third of the length of the original dataset (300000 observations), which is consistent with the amount of observations collected in one year (as described previously, the dataset consist of a compilation of data from 3 different years). The number of steps or “epochs” for training was chosen empirically, with 100 steps consistently showing a plateau in the accuracy improvement. Finally, the learning rate was defined as 0.01.

Similarly to the other predictive models used, the Convolutional Neural Network achieved an accuracy of 78.3%. The training process is illustrated in the following figure:

Convolutional Neural Network Training

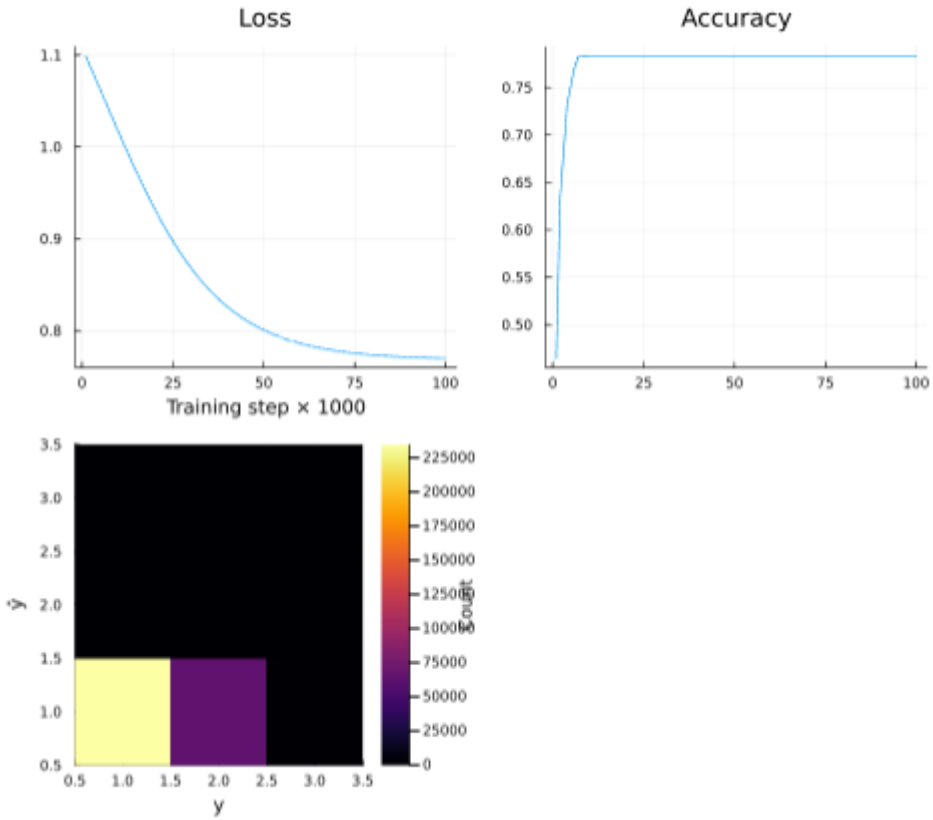

Figure 10: Using the complete dataset

Given that the results did not show a significant improvement compared to the Decision Tree predictive models, a “worst-case scenario” of the data was generated and used to train this model. In an attempt to reduce the outstanding difference of “Crash Severity” conditions, this new dataset excluded the “Clear” weather condition, “No defects” road defect condition, and “Dry” road surface condition. Since the dataset reduce its size to approximately 20000 observations, the batch size was reduced respectively to train the model.

With this “worst case scenario” dataset, the accuracy achieved reached 85.1%. The progress of training the Neural Network with this data are shown in the following figure:

Convolutional Neural Network Training

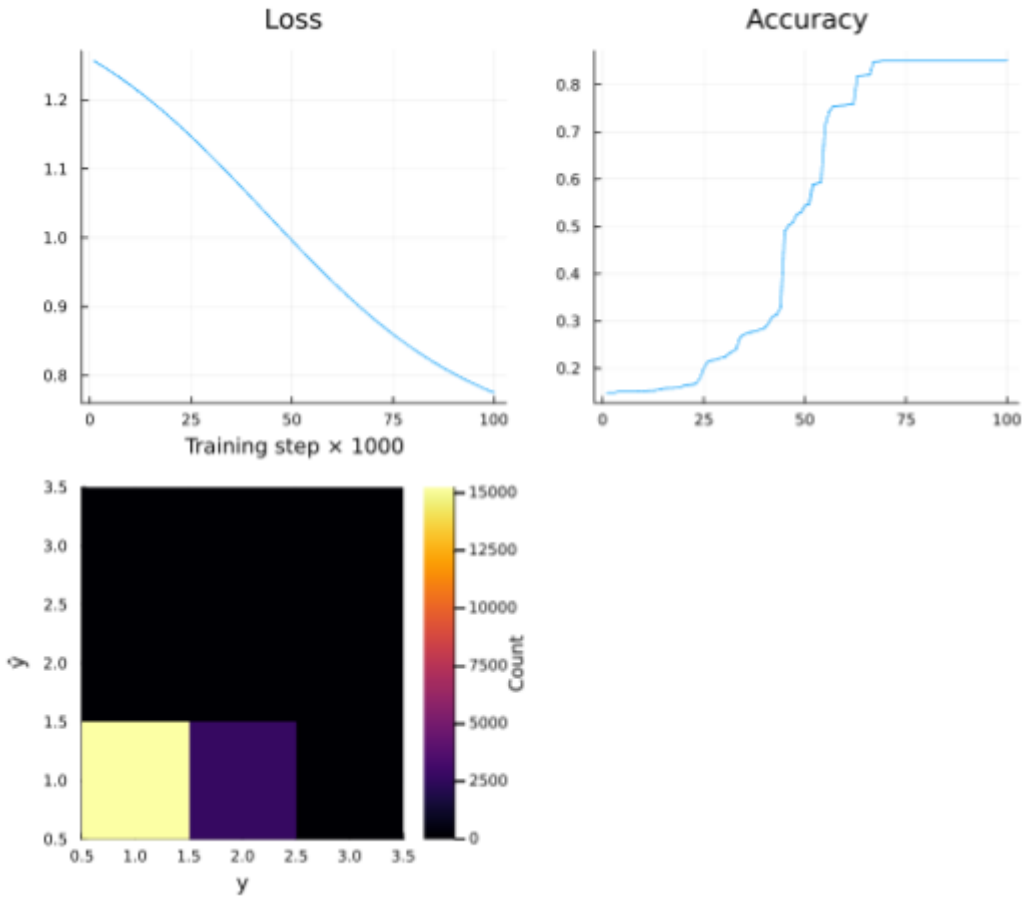

Figure 11: Using a reduced dataset

The results obtained using this predictive model may not be able to improve any further given the nature of the dataset used to train the predicted models. Since the amount of “Property Damage” as one of the possible “Crash Severity” is overwhelmingly greater than the other 2 possible classes, the models tend to learn that the majority of combinations of the selected parameters would lead to a “Property Damage” prediction.

References

Abdulhafedh, A. (2017) Road Crash Prediction Models: Different Statistical Modeling Approaches. Journal of Transportation Technologies, 7, 190-205. doi: 10.4236/jtts.2017.72014.

Akbar Danesh, Mehrdad Ehsani, Fereidoon Moghadas Nejad & Hamzeh Zakeri (2022) Prediction model of crash severity in imbalanced dataset using data leveling methods and metaheuristic optimization algorithms, International Journal of Crashworthiness, DOI: 10.1080/13588265.2022.2028471

Brunton, S., Kutz, N., (2017) Data Driven Science & Engineering: Machine Learning, Dynamical Systems, and Control.

Chin, H. C.; Quek, S. T. Measurement of Traffic Conflicts. Safety Science, Vol. 26(3), p. 169-185, 1997. DOI: https://doi.org/10.1016/S0925-7535(97)00041-6.

Farmer, C. M. Reliability of Police-Reported Information for Determining Crash and Injury Severity. Traffic Injury Prevention, 2003, n.4, p.38-44, 2003. DOI: https://doi.org/10.1080/15389580309855.

Hauer, E.; Hakkert, A. S. The Extent and Implications of Incomplete Accident Reporting. Transportation Research Record: Journal of the Transportation Research Board, n.1185, p.1-10, 1989.

Illinois Department of Transportation | IDOTAdmin https://gis-idot.opendata.arcgis.com/search?groupIds=6d2862031a6d47c7a8c211e38e423e05

Illinois Department of Transportation | Traffic Crash Report SR 1050 Instruction Manual https://idot.illinois.gov/home/resources/Manuals/Manuals-and-Guides

Lee, C., Hellinga, B., & Saccomanno, F. (2003). Real-Time Crash Prediction Model for Application to Crash Prevention in Freeway Traffic. Transportation Research Record, 1840(1), 67–77. https://doi.org/10.3141/1840-08

Mohammad Hesam Rashidi, Soheil Keshavarz, Parham Pazari, Navid Safahieh, Amir Samimi, Modeling the accuracy of traffic crash prediction models, IATSS Research, Volume 46, Issue 3, 2022, Pages 345-352, ISSN 0386-1112, https://doi.org/10.1016/j.iatssr.2022.03.004.

Myles, A.J., Feudale, R.N., Liu, Y., Woody, N.A. and Brown, S.D. (2004), An introduction to decision tree modeling. J. Chemometrics, 18: 275-285. https://doi.org/10.1002/cem.873

Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. (2011). Scikit-learn: Machine Learning in {P}ython, Journal of Machine Learning Research, Volume 12, pages 2825-2830.

Safarpour, H., Khorasani-Zavareh, D., & Mohammadi, R. (2020). The common road safety approaches: A scoping review and thematic analysis. Chinese journal of traumatology, 23(02), 113-121. https://doi.org/10.1016/j.cjtee.2020.02.005

World Health Organization (WHO). (2022). Road traffic injuries World Health Organization Regional Office for the Eastern Mediterranean. Geneva, Switzerland. https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries

Yasin, Y. J., Grivna, M., & Abu-Zidan, F. M. (2021). Global impact of COVID-19 pandemic on road traffic collisions. World journal of emergency surgery, 16(1), 1-14.

Yu, R., Han, L., & Zhang, H. (2021). Trajectory data based freeway high-risk events prediction and its influencing factors analyses. Accident Analysis & Prevention, 154, 106085. https://doi.org/10.1016/j.aap.2021.106085.