CHICAGO TRAFFIC CRASHES: AN APPROACH TO ENHANCE ROAD SAFETY

***Shweta Bankar sbankar***

***Priyanka Sreeramkavacham lsreeram***

***Kavya Elemati kavyaele***

# PROBLEM STATEMENT:

The concerning frequency of traffic accidents in Chicago underscores the need for an in-depth analysis to uncover trends, determinants, and actionable measures. This initiative seeks to explore the Traffic Crashes dataset provided by the Chicago Police Department, aiming to resolve critical inquiries and generate insights that could guide policy formulation and bolster road safety.

Background and Goals: As a vibrant urban center, Chicago experiences a substantial number of traffic incidents, leading to personal harm, fatalities, and material loss. It is imperative to dissect the contributing elements, patterns, and evolutions to forge effective countermeasures that diminish accidents and elevate communal road safety. The significance of addressing this issue stems from the substantial repercussions traffic accidents have on societal health, economic stability, and communal welfare.

Historically, the Chicago Police Department has been meticulous in gathering data regarding traffic incidents. Nonetheless, the data's vastness and intricacy call for an advanced analytical methodology. This initiative is poised to apply data science tools to distill valuable knowledge from the dataset, laying the groundwork for informed policy decisions.

Potential: This project stands to uncover obscured trends, pinpoint areas at increased risk, and illuminate factors leading to traffic incidents in Chicago. Through analytical rigor, it's possible to identify specific

scenarios, timings, and locales with a higher accident propensity, facilitating precise preventative actions.

The importance of this work is manifold. Primarily, it furnishes decision-makers and law enforcement with the data necessary for enacting efficacious preventive strategies. It also guides the strategic deployment of resources, maximizing the efficacy of road safety measures. Furthermore, the insights garnered could underpin initiatives aimed at raising public consciousness, enhancing driver education, and mobilizing community participation to nurture a culture of safety and accountability on the roads.

Ultimately, this project endeavors to markedly decrease traffic-related mishaps, injuries, and deaths in Chicago by analyzing and predicting the occurred damage during the collision and factors leading to it, fostering a more secure and sustainable cityscape. By addressing this pervasive issue, we hope to further the conversation about vehicle safety and provide inspiration for data-driven solutions that other cities facing similar problems can use..

# DATA SOURCE:

A comprehensive dataset on traffic crash patterns and contributing factors has been sourced from Kaggle for our analysis. This dataset was chosen in accordance with the requirements of our project, which calls for a comprehensive and extensive base of real-world data. It offers a strong foundation for finding trends, evaluating risk factors, and suggesting focused interventions to improve traffic safety. Because the dataset is so

comprehensive, it can be used for a wide range of analyses to better understand and lessen the effects of traffic crashes, from straightforward descriptive statistics to intricate predictive modeling. It has over 48 features and 794956 unique values. This dataset ‘Chicago Traffic Crashes - Chicago Police Dept.’ by Anoop Johny is available at [https://www.kaggle.com/datasets/anoopjohn](https://www.kaggle.com/datasets/anoopjohny/traffic-crashes-crashes) [y/traffic-crashes-crashes.](https://www.kaggle.com/datasets/anoopjohny/traffic-crashes-crashes) This is an unclean data with the features such as ‘LIGHTING\_CONDITION’,‘TRAFFICWAY\_TYPE’, ‘POSTED\_SPEED\_LIMIT’, ‘ALINGMENT’, ’DAMAGE’, ‘ROADWAY\_SURFACE\_COND’ ‘LANE\_CNT’, ‘CRASH\_DAY\_OF\_WEEK’, ‘CRASH\_HOUR’ ‘BEAT\_OF\_OCCURRENCE’, ‘STREET\_NAME’, ‘ROAD\_DEFECT’ and many

more. We will be performing the data cleaning and processing steps on the following dataset to make it compatible for our modeling of data, followed by the Exploratory Data Analysis (EDA).

# DATA CLEANING/PROCESSING:

We are cleaning up the data to gain a better understanding of traffic accidents. In order to concentrate on identifying the true causes of crashes and developing preventative measures, we are eliminating unnecessary data. Our data is now clearer, cleaner, and prepared for a thorough analysis.

Cleaning process 1: Removing features that are irrelevant to the analysis.

In the process of refining the data, several attributes were eliminated because they were considered unnecessary for relevant analysis. Items removed from the dataset encompassed identifiers and date stamps such as 'CRASH\_RECORD\_ID' and

'CRASH\_DATE', specifics about traffic signal devices, their working status, and crash scene details like 'ALIGNMENT' and 'ROADWAY\_SURFACE\_COND'.

Furthermore, administrative specifics labeled as 'REPORT\_TYPE', issues of legality like 'HIT\_AND\_RUN\_I', and precise location coordinates denoted by 'LATITUDE' and 'LONGITUDE' were also discarded. There are few other irrelevant features discarded for same reasons as shown in the subsequent image 1. Eliminating these pieces of data was critical to concentrate on the elements that significantly influence traffic accidents, thus

sharpening the effectiveness and pertinence of our analysis.



Upon eliminating unwanted features now we have 15 features for our analysis with 794956 unique entries.

Cleaning Process 2: We started a thorough data cleaning procedure with the goal of addressing the 'LANE\_CNT' feature. This procedure was necessary to deal with missing values. We initiated the process by classifying the traffic crash data based on 'TRAFFICWAY\_TYPE'. This categorization facilitated the individual treatment of each category, ensuring the accurate computation of the mode for 'LANE\_CNT' within each group. The determined mode value was then employed to fill any missing entries within the respective categories. After handling missing values, our focus shifted to identifying and addressing outliers within these specific groups. Outliers were defined as values constituting less than 1% of the group's size, considered too infrequent to significantly influence the overall trend analysis. Systematic removal of these values from the dataset was then executed.

Cleaning process 3:

To enhance readability and enable further analysis, the resulting grouped data underwent a transformation from an index to a DataFrame using the reset\_index method. This transformation is crucial as it reinstates the grouped data into a standard DataFrame format, clearly displaying the 'TRAFFICWAY\_TYPE', 'LANE\_CNT', and

the corresponding count of occurrences.

The grouped data now provides an overview of our dataset, presenting the accident count based on the type of trafficway and the number of lanes. This condensed representation serves as an insightful analysis vector for exploring patterns and trends in the dataset.

Cleaning process 4: In the 'STREET\_NAME' column, we found a specific amount of null entries. We eliminated records that lacked the street name because we knew that this important piece of information could jeopardize the accuracy and comprehensiveness of our analysis. We filtered out the records where 'STREET\_NAME' was null. Following the elimination of entries with null values in the 'STREET\_NAME' column, we validated the success of this procedure by re-evaluating the dataset for any remaining null values in that specific column. The outcome affirmed the complete removal of all null entries, resulting in a dataset where each record now possesses a specified 'STREET\_NAME'. By linking each entry in our dataset to a particular street, we laid the groundwork for obtaining more accurate and significant observations regarding the spatial distribution of traffic crashes.

Cleaning process 5: We examined the 'INTERSECTION\_RELATED\_I' column to assess its appropriateness for inclusion in our study. Initially, we investigated the presence of null values within the column and explored the range of unique values it encompassed. We noticed a lot of empty spots in the 'INTERSECTION\_RELATED\_I' column.

Because there were so many of these empty spots, we think that this column might not have enough detailed or specific information for our analysis. Based on this evaluation, we chose to get rid of the entire 'INTERSECTION\_RELATED\_I' column

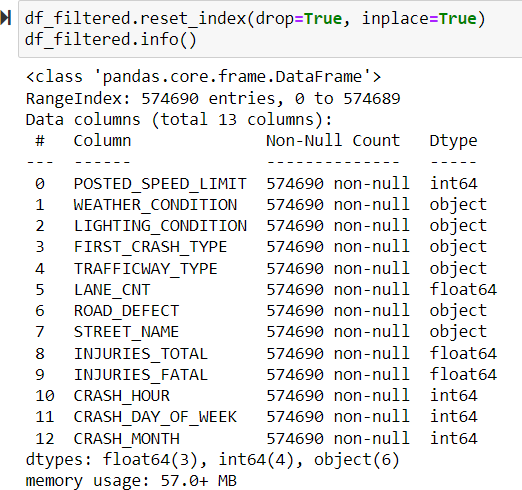
from the dataset.

Cleaning process 6: We visualized the distribution of 'POSTED\_SPEED\_LIMIT' through a histogram, dividing the data into 30 bins for a detailed view. To further refine our dataset and ensure the reliability of our analysis, we calculated the Interquartile Range (IQR) for 'POSTED\_SPEED\_LIMIT'.

We calculated the lower and upper bounds for outliers as 1.5 times the IQR below the first

quartile (Q1) and above the third quartile (Q3), respectively. Once we found the outliers, we took them out of our data. After doing that, we looked again at how the speed limits were spread out using a bar graph. By getting rid of the outliers, the new graph showed a better and truer picture of the usual speed limits. This helps us to better study and understand what things might be causing traffic crashes and how severe they are.

Cleaning process 7: After making various adjustments, including the removal of outliers and handling null values, it was necessary to refresh the dataset's indexing to reflect these changes accurately.To make our filtered data look neat and organized, we use the "reset\_index" method on our DataFrame. We include the "drop=True" option to get rid of the old index and prevent it from becoming a new column in the DataFrame. This way, our data stays clean and well-arranged.

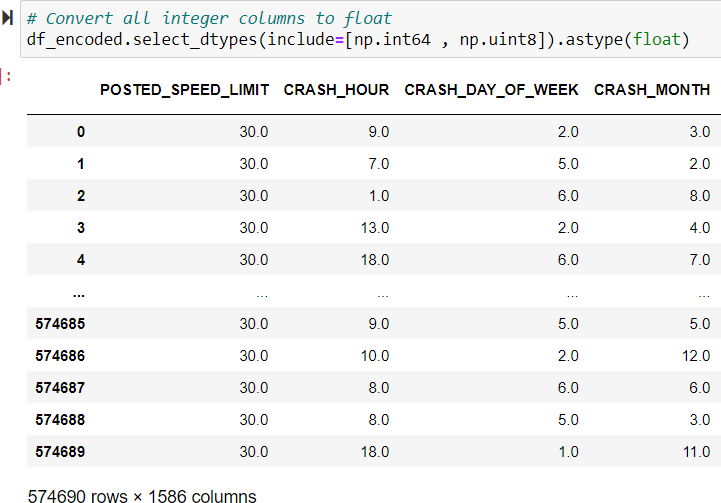


Cleaning process 8: Understanding that most machine learning programs work better with numbers, we used a method called one-hot encoding to change certain categories into a format that the computer can easily understand. We focused on specific categories like 'WEATHER\_CONDITION', 'LIGHTING\_CONDITION',

|  |  |  |  |
| --- | --- | --- | --- |
| 'FIRST\_CRASH\_TYPE', |  | the index is neat and | in order after we made |
| 'TRAFFICWAY\_TYPE', 'ROAD\_DEFECT', |  | lots of changes | while encoding and |
| and 'STREET\_NAME'. One-hot encoding |  | converting types. |  |
| changes these categories into numbers by |  |  |  |
| creating new columns. For instance, if a crash |  |  |  |
| happened in the rain, the |  |  |  |
| 'WEATHER\_CONDITION' column would | **IV.** | **EXPLORATORY** | **DATA ANALYSIS** |
| have a new column for rain, and if it was not |  | **(EDA):** |  |
| raining, it would have a separate column for |  |  |  |
| that. We used a tool called pandas' |  | During our data | exploration, we used |

get\_dummies to do this automatically for the chosen columns in our dataset. The outcome was a modified DataFrame, df\_encoded, where the categories turned into multiple columns with 1s and 0s, making it easier for the computer to work with and improve its predictions.

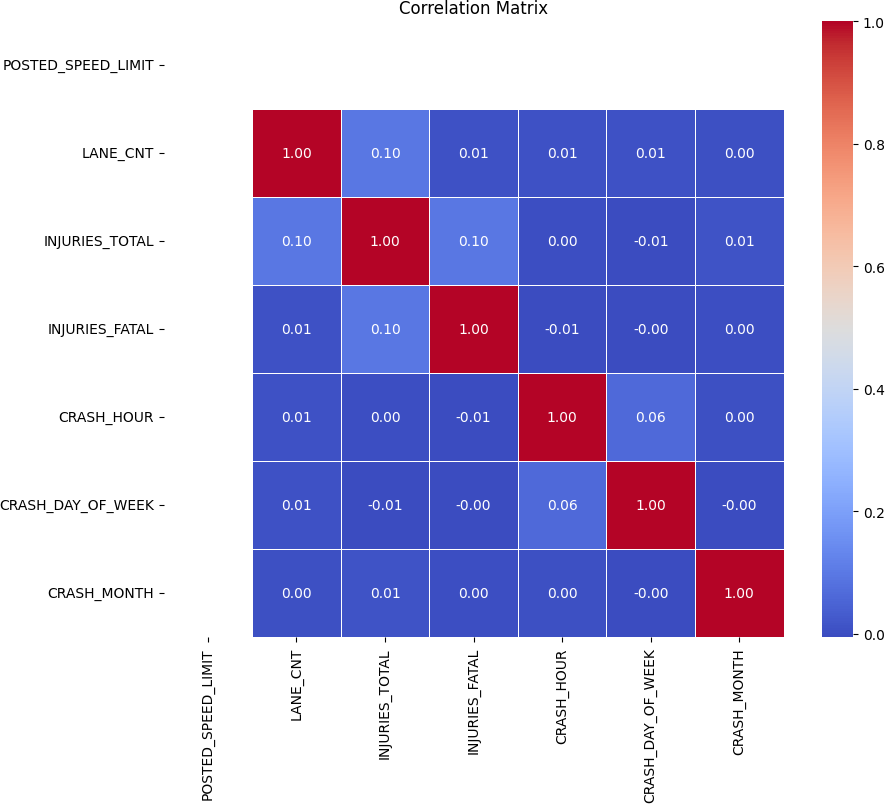
Cleaning process 9: We converted integer columns to float. To ensure consistency across numerical data types, we converted all columns with integer data types to float. This step is crucial for maintaining uniformity in our dataset. This action ensures that all numerical features in the dataset are represented as floating-point numbers, which can be important for certain types of computation and analysis, particularly those involving division or normalization where decimal precision may be necessary.



Cleaning process 10: Once we changed our data using one-hot encoding and turned numbers into decimals to keep everything consistent, the last thing we did to get our data ready was to reset the order of our ‘df\_encoded’ table. We did this to make sure

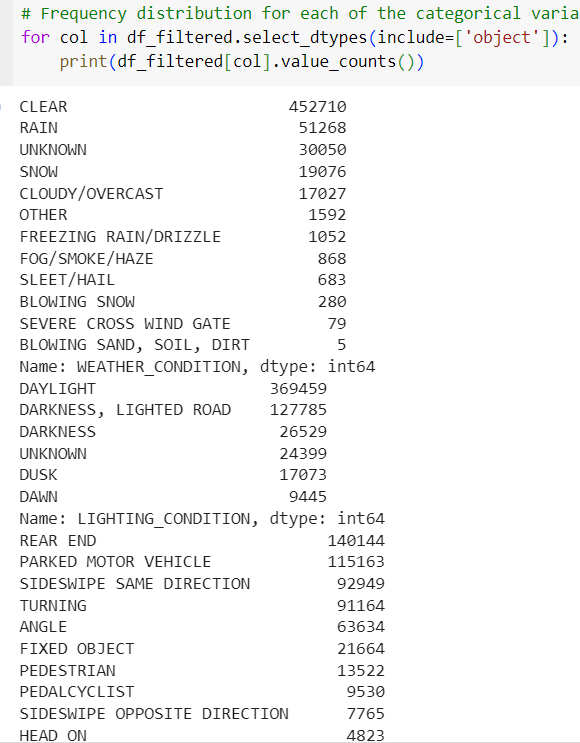
methods that follow the advice from the NIST publication, and the basic ideas explained by John Tukey.

Operation 1: Calculating the correlation matrix. The first operation in our EDA was to construct and visualize a correlation matrix for the ‘df\_filtered’ dataset. This step is important in understanding the linear relationships between numerical features within our dataset. Using our filtered DataFrame, df\_filtered, we calculated the correlation matrix using the.corr() method. This produced a matrix reflecting the Pearson correlation coefficients between all numerical features. We used the matplotlib library to create a heatmap visualization that made it easier to understand these correlations. Using this method, we were able to use color intensity to encode the strength of correlations, giving a clear visual depiction of the relationships between the variables.



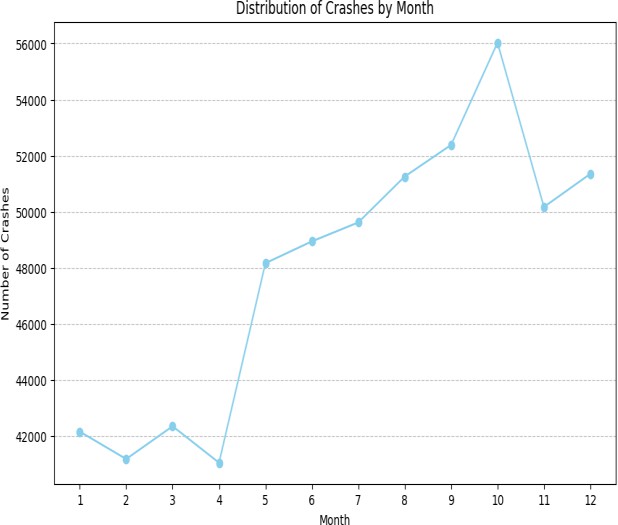
Our understanding is that none of these numeric values have a strong corelation between each other.

Operation 2: Frequency distribution for each of the categorical variables. Analyzing how often different categories appear in the data helps us explore and understand the overall makeup of the dataset better. It's like looking closely at the types of things we have in the data to get a clearer picture of what's going on. We iterated through each categorical column in the dataset, utilizing ‘select\_dtypes(include=[‘object’])’ to filter these columns specifically. For each categorical variable, we employed the ‘.value\_counts()’ method to calculate the frequency of each unique category within the variable. This approach provided a quantitative measure of how often each category occurs within the dataset. The information we got from frequency distribution analysis helped us make decisions about how to organize and modify our data. It made us clear how to strategies feature selection, handle rare categories and support the creation of new features that could better capture the underlying patterns in the data.



Operation 3: Creating a bar plot for the distribution of crashes by month to identify which months had the most crashes and which months had the least crashes.

We began by grouping the data by the 'CRASH\_MONTH' column to aggregate the total number of crashes occurring in each month. The aggregated data was then sorted by month to maintain chronological order, facilitating a coherent analysis. We used a line graph to show how many crashes happened each month. Line graphs are good for seeing how things change over time. The graph displayed how the number of crashes went up and down during the year. The graph showed that some months had more or fewer crashes than others. This suggests there could be patterns that change with the seasons. For example, months with bad weather might have more crashes because driving is harder in those conditions.

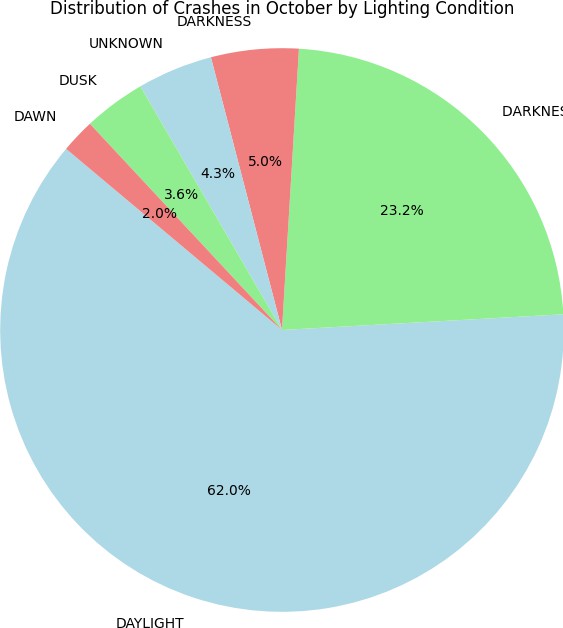


We can see that the maximum number of crashes occurred in the month of October.

Operation 4: Plotting the distribution of car crashes for the month 'October' based on the lightning conditions.

We started by looking at crash data specifically for October. This month is interesting because it marks the shift to autumn, bringing different lighting conditions. We used a pie chart to show the different lighting conditions' impact on crashes. A pie chart was chosen because it's good at representing the proportionate effect

of each lighting condition on the total number of crashes. Examining how crashes are spread out based on lighting conditions can help decide on specific safety actions. For instance, it might suggest improving street lighting or giving special warnings to drivers when the lighting is particularly risky.



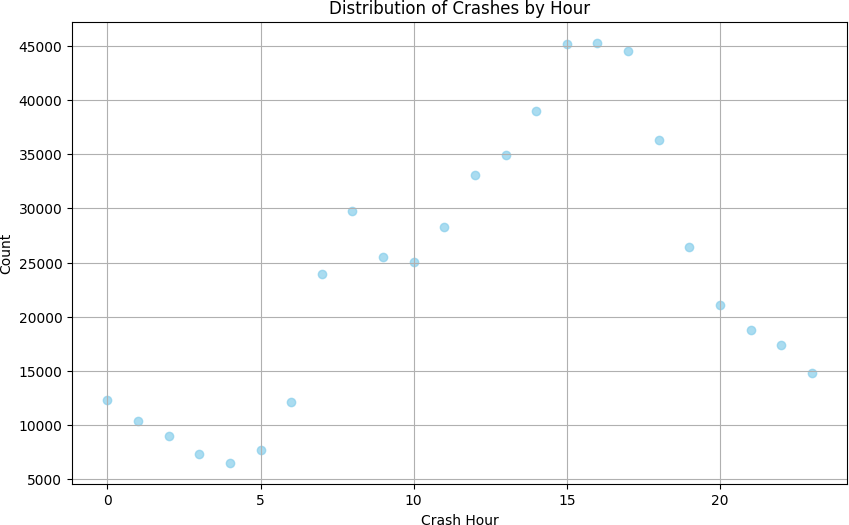
This helped us understand that most accidents occurred in the daytime. The relatively lower but still significant crash rates during dusk and dawn suggest the need for increased awareness around these times.

Operation 5: Analysis of crash frequency by Hour.

Our goal was to find out if there are specific times when car crashes happen more often. To do this, we used a scatter plot, which is like a graph that shows how many crashes occurred at each hour of the day.

We aggregated the number of crashes by hour using the value\_counts() method on the 'CRASH\_HOUR' column of the filtered dataset, ensuring that the data was sorted by the index to maintain chronological order. We extracted the hours and corresponding crash counts to prepare the data for visualization. On the scatter plot, each dot showed how many crashes happened during a certain hour. The horizontal line told us which hour it was. This helped us find hours where there were a lot or very few crashes. It gave us clues about things like busy times, how bright it was, or

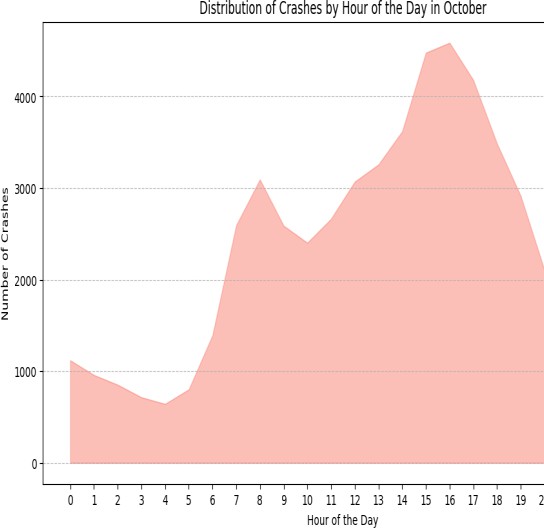
other time-related factors that might be linked to crash rates.



The highest number of crashes occurred at the 15th hour of the day (i.e. at 3pm).

Operation 6: Temporal analysis of crashes in October.

We started by selecting only the accidents that happened in October from the data. Next, we grouped the information based on the 'CRASH\_HOUR' to find out how many accidents occurred at each hour of the day in that month. Once the grouping was done, we arranged the data so that it stays in the right order of hours as they happen in a day. The information we get from this chart might indicate when it's a good idea to improve road safety or be extra careful during specific times. These outcomes could result in creating features based on time, like flags for risky hours, which could make upcoming models better at predicting outcomes.

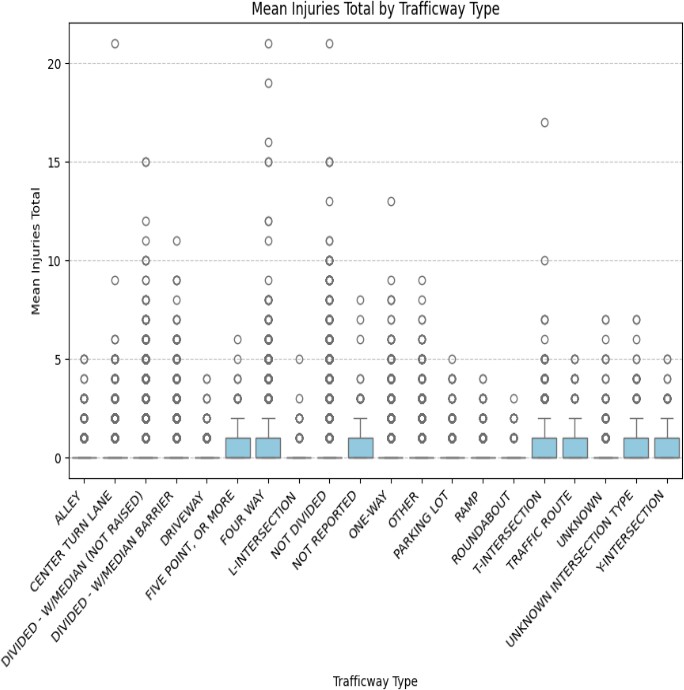


More safety measurements should be implemented during the period of 2pm to 7pm as it is highly prone to occurrence of accidents.

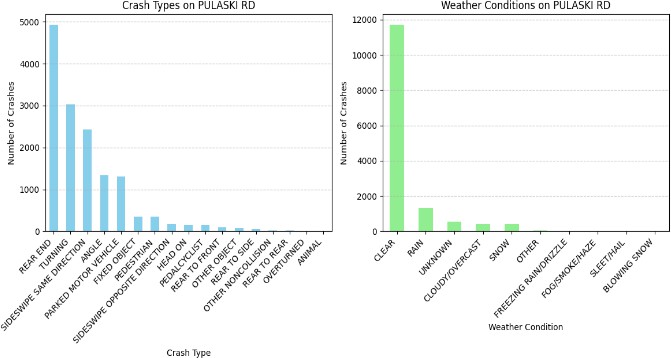
Operation 7: Evaluating the Impact of Trafficway Type on Injury Severity. We carried out a group-wise analysis by categorizing the data according to 'TRAFFICWAY\_TYPE' and calculated the mean 'INJURIES\_TOTAL' for each category. This approach allowed us to compare the average injury outcomes across different trafficway types. We chose to use a box plot, made with the Seaborn library, to show how the average total injuries are spread out for each type of trafficway. Box plots are great because they give us a good idea about where most of the data is centered and how much it varies. We looked closely at 'TRAFFICWAY\_TYPE' and

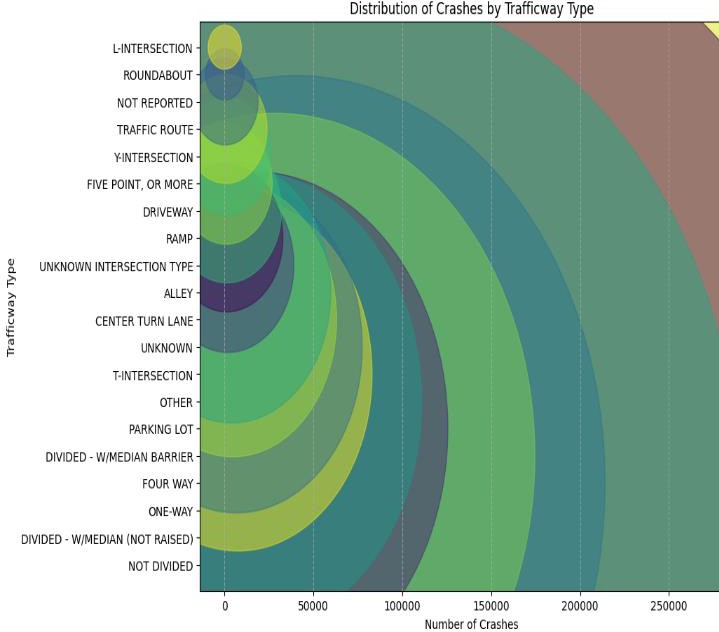
'INJURIES\_TOTAL' because these columns are directly related to our goal of figuring out what happens in terms of injuries depending on the type of trafficway. We looked at how many people got hurt in different types of roads. We grouped the data by the kind of road, like highways or streets, and found the average number of injuries. This helped us understand how severe injuries are on each type of road.

The size of the boxes, which show the middle 50% of the data called the interquartile range (IQR), tells us how spread out the injury counts are. If the box is longer, it means there is more variation in the number of injuries for that type of trafficway. The average number of injuries varies depending on the type of road. This hints that some types of roads might be more or less risky when it comes to the chances of injuries happening.

Operation 8: Analyzing which traffic way types are more prone to accidents.

We used a bubble plot to do EDA, which is good at showing where crashes happen on different types of roads. The size of each bubble shows how many crashes there are, making it easy to see the differences between the categories. The bubble plot indicates that certain trafficway types, such as 'Not Divided', 'Divided with Median', and 'One- way' roads, experience a higher frequency of crashes. This could point to inherent risk factors associated with these trafficway designs. Tiny bubbles, like the ones representing 'Roundabouts' and 'L- Intersections', imply fewer accidents. This might mean there's a lower chance of crashes or less traffic in those places.



Maximum number of accidents occured when the trafficway type is "not divided".

Operation 9: Finding out the top 5 streets where the maximum number of crashes occurred and plotting the weather conditions and type of crashes for those streets where the crashes happened.

When picking the things we look at in our analysis, we focused on the ones that give us clear information about the crashes. 'STREET\_NAME' was really important because it helped us group the crash data by where it happened. Also, 'FIRST\_CRASH\_TYPE' and

'WEATHER\_CONDITION' were crucial for understanding the kind of crash and the weather conditions during the crashes. We employed the .value\_counts() method to quantify the crashes by street, which streamlined our subsequent analyses and visualizations. Bar charts helped us see and compare different crash types and weather conditions in a straightforward way. This made it easy to figure out the common crash types and weather conditions linked to the places where crashes happen a lot.

Crashes on Pulaski road: The crashes that happen most often are "REAR END,", "TURNING" and "SIDESWIPE SAME

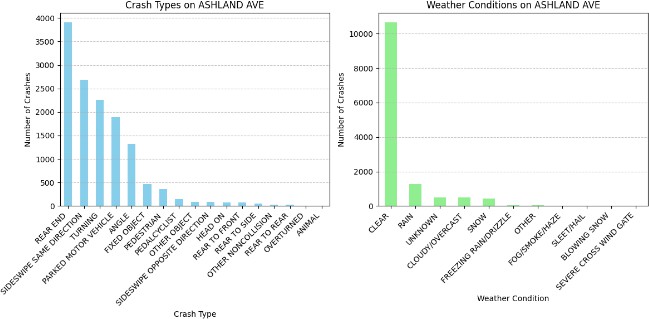
DIRECTION." The ones that don't happen very often are "REAR TO SIDE" and "REAR TO REAR,".

Weather conditions on Pulaski road: The usual weather condition was "RAIN," causing notably fewer crashes, and then came "UNKNOWN" and

"CLOUDY/OVERCAST." Weather

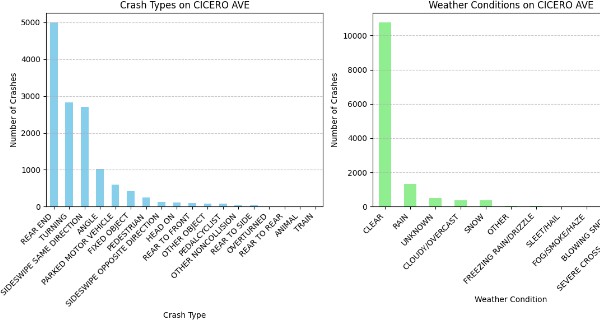
conditions like "SNOW" and others such as "FREEZING RAIN/DRIZZLE," "FOG/SMOKE/HAZE," "SLEET/HAIL,"

and "BLOWING SNOW" have way fewer crashes linked to them.



Crashes on Ashland Avenue: The crash type that happens the most is "REAR END," and it occurred more than 3500 times. After that, we have "SIDESWIPE SAME DIRECTION" and "TURNING" crashes.

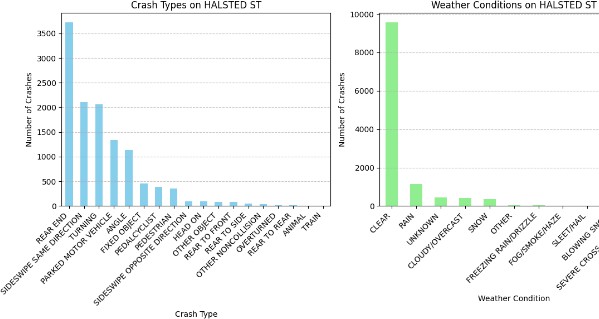
Weather conditions on Ashland Avenue: "CLEAR" weather condition is by far the most common during the crashes, with nearly 9000 crashes. "RAIN" is the next most common, but it drops significantly to under 1000 crashes.

'RAIN' come next, followed by 'UNKNOWN' and 'CLOUDY/OVERCAST'.

Inclement weather like snow, freezing rain, and fog had fewer incidents.

Crashes on Cicero Avenue: Rear-end crashes happen the most, almost 5000 times. Next are sideswipe crashes in the same direction, going over 3000. Turning crashes are also important, around 2500 times. Other types of crashes, like sideswipes in the opposite direction, angle crashes, parked car crashes, crashes with fixed objects, and pedestrian- involved crashes, become less common as you go down the list.

Weather conditions on Cicero Avenue: The most crashes happen when the weather is clear, almost reaching 9000 incidents. Rain comes next, with a bit over 1000 crashes. Crashes are less common in unknown conditions, cloudy or overcast weather, and snow. Other conditions like freezing rain or drizzle, sleet or hail, fog or smoke or haze, blowing snow, and severe crosswinds are associated with very few crashes.



Crashes on Halsted Street: The most frequent kind of crash on HALSTED ST is 'REAR END,' happening more than 3500 times. Next are 'SIDESWIPE SAME DIRECTION' and

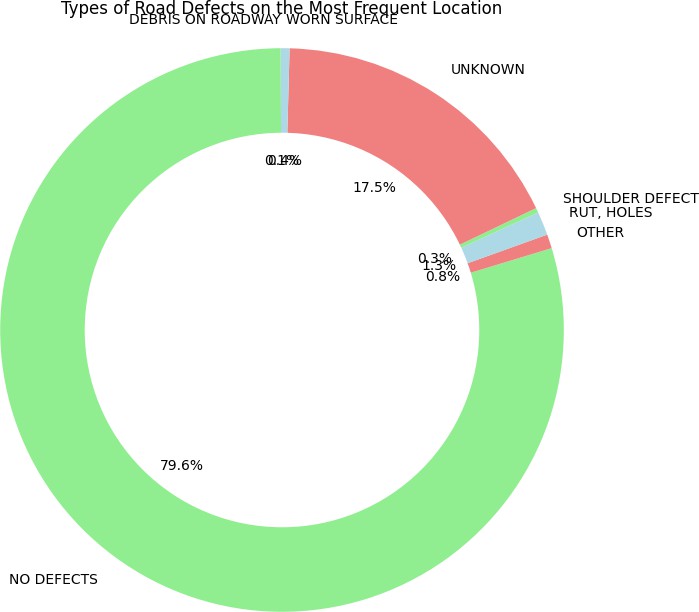
'TURNING,' with less than 1000 incidents but still quite a lot. The least usual types are 'TRAIN,' 'REAR TO REAR,' and 'ANIMAL,'

with very few occurrences compared to the other types.

Weather conditions on Halsted Street: Most crashes, around 10,000 incidents, happened in clear weather 'CLEAR'. Rainy conditions

Operation 10: Finding the street where maximum number of crashes occur and then plotting a graph based on the type of road defect.

We started exploring the data by grouping the information based on the 'STREET\_NAME.' This helped us combine the accidents and count how many happened on each street. We then sorted the results to find the street with the maximum number of crashes. The data was then further filtered to focus exclusively on the identified street. We used a donut pie chart to visually highlight the proportion of different road defects in the total dataset. This helps quickly identify and prioritize prevalent issues for safety improvements.



From the visual analysis above, it became clear which road defects are most associated with accidents on the most hazardous street.

**APPLYING MODELS:**

🡪Splitting into test and train dataset.

A screenshot of a computer program

Description automatically generated

**Algorithm 1: Linear Regression:**

**Justification:**

We chose Linear Regression for our problem because it is a simple and widely used algorithm for predicting continuous values. In our traffic crash analysis, Linear Regression can help us estimate the relationship between various contributing factors and the severity of a crash, represented by a continuous target variable such as the damage weight.

**Tuning/Training:**

In our implementation of Linear Regression, we used the default hyperparameters provided by the scikit-learn library.

A screenshot of a computer code

Description automatically generated

**Effectiveness:**

To evaluate the effectiveness of the Linear Regression model, we used the Mean Squared Error (MSE) as the performance metric. The MSE measures the average squared difference between the predicted values and the actual values.

A screenshot of a computer code

Description automatically generated

In our analysis, the Linear Regression model achieved an MSE of **46.82%.**This suggests that the model is not that good in capturing the linear relationship between the input features and the target variable.

**Visualisation:**

A close-up of a computer screen

Description automatically generated

A graph with a blue line

Description automatically generated

**Algorithm 2: Logistic Regression**

**Justification:**

We chose Logistic Regression for our problem because it is a widely used algorithm for binary classification tasks. In our traffic crash analysis, Logistic Regression can help us predict the probability of a crash being severe or not based on various contributing factors.

**Tuning/Training:**

In our implementation of Logistic Regression, we used the default hyperparameters provided by the scikit-learn library.

A screenshot of a computer program

Description automatically generated

**Effectiveness:**

To evaluate the effectiveness of the Logistic Regression model, we used several performance metrics. The model achieved a Precision of **50.223%,** indicating that  **50.23%** of the instances predicted as a particular class truly belonged to that class. The Recall of **62.49%** suggests that the model correctly identified **62.49%** of the instances that actually belonged to each class. The F1 Score, which is the harmonic mean of Precision and Recall, was **48.07%,** providing a balanced measure of the model's performance.

**Visualisation:**

We also visualized the actual vs. predicted probabilities using a scatter plot. This visualization helps us assess how well the Logistic Regression model is calibrated and how close the predicted probabilities are to the actual binary outcomes.

A graph with a red line

Description automatically generated

The Logistic Regression model DOESN’T demonstrate good performance in terms of Precision, Recall, and F1 Score, indicating its effectiveness in predicting crash severity

**Algorithm 3: Naive Bayes**

**Justification:**

We chose the Naive Bayes algorithm for our problem because it is a simple and efficient probabilistic classifier that can handle high-dimensional data. Naive Bayes is based on Bayes' theorem and assumes that the features are conditionally independent given the class label. In our traffic crash analysis, Naive Bayes can provide a quick and effective way to predict crash severity based on various contributing factors.

**Tuning/Training:**

The algorithm learns the class-conditional probabilities and prior probabilities directly from the training data. In our implementation, we used the Gaussian Naive Bayes variant, which assumes that the continuous features follow a **Gaussian (normal) distribution** within each class.

A screenshot of a computer code

Description automatically generated

**Effectiveness:**

To evaluate the effectiveness of the Naive Bayes model, we used several performance metrics. The model achieved a Precision of **61.13%,** indicating that **53.95%** of the instances predicted as a particular class truly belonged to that class. The Recall of **53.95%** suggests that the model correctly identified **53.95%** of the instances that actually belonged to each class. The F1 Score, which is the harmonic mean of Precision and Recall, was **51.69**%, providing a balanced measure of the model's performance.

**Visualization**

We also visualized the performance metrics using a bar plot to compare the Precision, Recall, and F1 Score of the Naive Bayes model.

The Naive Bayes model demonstrated decent performance in terms of Precision, Recall, and F1 Score, indicating its effectiveness in predicting crash severity based on the available features.

A screenshot of a computer program

Description automatically generated

A bar graph with different colored squares

Description automatically generated

**Algorithm 4: Decision Tree**

**Justification:**

We chose the Decision Tree algorithm for our problem because it is a simple and powerful algorithm that can model complex relationships and capture non-linear patterns in the data. Decision Trees are known for their interpretability, as they provide a clear and intuitive representation of the decision-making process. In our traffic crash analysis, Decision Trees can help us identify the most influential factors contributing to crash severity and provide a hierarchical structure of decision rules.

**Tuning/Training:**

In our implementation of the Decision Tree, we used the default hyperparameters provided by the scikit-learn library.

A screen shot of a computer code

Description automatically generated

**Effectiveness:**

To evaluate the effectiveness of the Decision Tree model, we used several performance metrics. The model achieved a Precision of **50.51%,** indicating that **50.51%,** of the instances predicted as a particular class truly belonged to that class.

The Recall of **51.70%** suggests that the model correctly identified **51.70%** of the instances that actually belonged to each class. The F1 Score, which is the harmonic mean of Precision and Recall, was **51.06%.**

**Visualisation**

A screenshot of a graph

Description automatically generated

visualized the feature importance to gain into the most influential factors driving the model's predictions. The feature importance plot shows the relative importance of each feature in the Decision Tree model.

**Algorithm 5: Gradient Boosting Machine (GBM)** **Justification:**

We chose the Gradient Boosting Machine (GBM) algorithm for our problem because it is a powerful ensemble learning technique that combines multiple weak learners to create a strong predictive model. GBM is known for its ability to handle complex relationships and capture intricate patterns in the data. In our traffic crash analysis, GBM can effectively model the non-linear interactions between various factors contributing to crash severity and provide accurate predictions.

**Tuning/Training**:

In our implementation of GBM, we used the default hyperparameters provided by the scikit-learn library.

A screenshot of a computer program

Description automatically generated

**Effectiveness:**

evaluate the effectiveness of the Gradient Boosting Machine model, we used several performance metrics and visualization techniques. The model achieved a Precision of **50.89%,** indicating that **50.89%** of the instances predicted as a particular class truly belonged to that class. The Recall of **63.38%** suggests that the model correctly identified **63.38%** of the instances that actually belonged to each class. The F1 Score, which is the harmonic mean of Precision and Recall, was **51.39%**, providing a balanced measure of the model's performance.

**Visualisation**

We also visualized the Confusion Matrix to gain into the model's classification performance across different classes. The Confusion Matrix allowed us to identify the number of true positive, true negative, false positive, and false negative predictions for each class.

A graph of a number of numbers

Description automatically generated with medium confidence

To further evaluate the model's performance, we plotted the class-wise performance metrics, including Precision, Recall, and F1 Score, using bar plots. These visualizations provided a clear comparison of the model's performance across different classes.

A graph with different colored squares

Description automatically generated

A graph with a green and blue rectangle

Description automatically generated

A graph with a green and blue rectangle

Description automatically generated

A graph showing support by class

Description automatically generated

Moreover, we employed the **One-vs-Rest (OvR)** strategy to convert the multiclass problem into multiple binary classification problems. By plotting the Receiver Operating Characteristic (ROC) curve and calculating the Area Under the Curve (AUC) for each class.

A screenshot of a computer program

Description automatically generated

A graph of a line graph

Description automatically generated with medium confidence

Overall, the Gradient Boosting Machine algorithm demonstrated its effectiveness in handling the complexity of our traffic crash data and providing accurate predictions.

**Algorithm 6: K-Nearest Neighbours (KNN):**

**Justification:**

We chose the K-Nearest Neighbours (KNN) algorithm for our problem because of its simplicity and ability to capture local patterns in the data. KNN is a non-parametric algorithm that does not make any assumptions about the underlying data distribution. It classifies a data point based on the majority class of its k nearest neighbours in the feature space.

**Tuning/Training:**

In our implementation of KNN, we set the number of neighbours (k) to 5. The choice of k is an important hyperparameter that can impact the model's performance. A smaller value of k allows the model to capture more local patterns but may lead to overfitting, while a larger value of k can smooth out the decision boundary but may result in underfitting. We chose k=5 as a reasonable balance between capturing local patterns and avoiding overfitting.

A screenshot of a computer code

Description automatically generated

**Effectiveness:**

To evaluate the effectiveness of the KNN model, we used several performance metrics. The model achieved a Precision of 51.34%, indicating that 54.22% of the instances predicted as a particular class truly belonged to that class. The Recall of 54.22% suggests that the model correctly identified 54.22% of the instances that belonged to each class. The F1 Score, which is the harmonic mean of Precision and Recall, was 52.53%, providing a balanced measure of the model's performance.

**Visualisation:**

A computer code with black and red text

Description automatically generated

A graph of confusion matrix

Description automatically generated

In addition to these metrics, we also visualized the Confusion Matrix to gain insights into the model's classification performance.

**Algorithm 7: K-Means Clustering:**

**Justification:**

We chose the K-Means clustering algorithm for our problem because it is a simple and effective unsupervised learning technique for partitioning data into distinct clusters. K-Means aims to group similar data points together based on their feature similarities, without requiring any labelled data. In our traffic crash analysis, K-Means can help us discover inherent patterns or groupings within the data, potentially revealing insights into different types of crashes or factors contributing to their severity.

**Tuning/Training:**

In our implementation of K-Means, we specified the number of clusters (k) as 3. The choice of k is the number of clusters the algorithm will try to identify in the data. We chose k=3 based on our understanding of the problem and the potential existence of three distinct groups or severity levels in the crash data.

A screenshot of a computer program

Description automatically generated

**Effectiveness:**

To evaluate the effectiveness of the K-Means clustering algorithm, we used a combination of evaluation metrics and visualization techniques. Since K-Means is an unsupervised learning algorithm, we don't have ground truth labels to directly assess its performance.

In our analysis, we calculated the accuracy of K-Means clustering by assigning class labels the clusters based on the mode of true labels in each cluster. We then compared the assigned cluster labels with the true labels and computed the accuracy. The achieved accuracy of **62.49%** .

**Visualisation**

To further visualize the clustering results, we employed Principal Component Analysis (PCA) to reduce the dimensionality of the data to two dimensions. The visualization provided insights into the distribution of data points within each cluster and the overall structure of the clustering solution.

A screenshot of a computer program

Description automatically generated

A diagram of clustering data

Description automatically generated with medium confidence

**References:**

Decision Tree Algorithm: Doing Data Science Straight Talk from the Frontline by -Rachel Schutt &Cathy O’Neil  
Gradient Boosting Machine Algorithm: A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning byJason Brownlee(https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/)

**Peer Evaluation Form for Final Group Work CSE 487/587B**

* Please write the names of your group members.

**Group member 1 : Shweta Ganesh Bankar**

**Group member 2 : Lakshmi Priyanka Sreeramkavacham Group member 3 : Kavya Elemati**

* Rate each groupmate on a scale of 5 on the following points, with 5 being HIGHEST and 1 being LOWEST.

|  |  |  |  |
| --- | --- | --- | --- |
| **Evaluation Criteria** | **Group member 1** | **Group member 2** | **Group member 3** |
| How effectively did your group mate work with you? | 5 | 5 | 5 |
| Contribution in writing the report | 5 | 5 | 5 |
| Demonstrates a cooperative and supportive attitude. | 5 | 5 | 5 |
| Contributes significantly to the success of the project . | 5 | 5 | 5 |
| **TOTAL** | 20 | 20 | 20 |

**Also please state the overall contribution of your teammate in percentage below, with total of all the three members accounting for 100% (33.33+33.33+33.33 ~ 100%) :**

**Group member 1 : 33.33**

**Group member 2 : 33.33**

**Group member 3 : 33.33**