Crash Data Prediction of Delaware Counties

Soha Ahmed*, Shantanu Deshpande†, Kylie Dickinson‡, Steven Ha§, and Thinh Nguyen¶
*Statistics & Electrical Engineering, University of California Davis, Davis, USA
Email: soahmed@ucdavis.edu
†Statistics, University of California Davis, Davis, USA
Email: sdeshp@ucdavis.edu
‡Statistics, University of California Davis, Davis, USA
Email: kdickinson@ucdavis.edu
§Applied Mathematics & Statistics, University of California Davis, Davis, USA
Email: steha@ucdavis.edu
¶Applied Mathematics & Statistics, University of California Davis, Davis, USA
Email: tulnguyen@ucdavis.edu

I. Introduction

A. Motivation & Objective

Motor vehicle crashes are a persistent safety concern. To understand the potential reasons behind various crashes, in this project report, we aim to predict the severity of a crash by utilizing the conditions known at the time of the accident. The severity of the crash could vary from minor to fatal damage. By identifying factors that lead to specific outcomes, we can use this project to help improve the public safety measures. We will be using publicly available crash data from the Delaware Department of Safety and Homeland Security. This is a multi-class classification problem. We will be predicting **crash classification code**, which indicates the severity of an accident. We perform data pre-processing, exploratory data analysis, feature engineering, and the application of machine learning models. This helped us explore how various features influence the severity of a crash.

B. Dataset Description

The dataset was obtained from the Delaware Department of Homeland Security. It contains a comprehensive record of crash incidents from the year 2004, totaling up to approximately 531,000 entries and 39 columns. Each row represents a single crash event. The dataset includes features such as:

TABLE I
Summary of Dataset Features

Feature	Question	Possible Values
Crash Date and Time	When did the crash happen?	The Date and time at which the crash occurred.
Crash Classification Code / Description	What is the severity of the crash?	1 = Property Damage Only, 2 = Personal Injury, 3 = Non-Reportable, 4 = Fatality
Pedestrian Involved	Indicates if a pedestrian was involved in the collision.	No and Yes
Day of the Week Code / Description	What day of the week did the crash occur?	01 = Sunday, 02 = Monday, 03 = Tuesday, 04 = Wednesday, 05 = Thursday, 06 = Friday, 07 = Saturday
Collision on Private Property	Indicates if the collisions were on private property. Private property is defined here as location inputs of "outside of right of way" and "private parking lot" in collision reports.	No and Yes
Manner of Impact Code / Description	How the two motor vehicles initially came together, without regard to the direction of the force. Stationary motor vehicles are not included.	01 = Front to rear, 02 = Front to front, 03 = Angle, 04 = Sideswipe (same direction), 05 = Sideswipe (opposite direction), 06 = Rear to side, 07 = Rear to rear, 88 = Other, 99 = Unknown, 00 = Not a collision between two vehicles
Alcohol Involved	Indicates suspected alcohol use by any person involved in the crash, excluding passengers.	No and Yes
Drug Involved	Indicates suspected drug use by any person involved in the crash, excluding passengers.	No and Yes
Road Surface Code / Description	Describes the road conditions during the crash.	01 = Dry, 02 = Wet, 03 = Snow, 04 = Ice/Frost, 05 = Sand, 06 = Water (Standing and moving), 07 = Slush, 08 = Oil, 09 = Mud / Dirt / Gravel, 88 = Other, 99 = Unknown

Lightning Condition Code / Description	The type/level of ambient light that existed at the time of the crash	01 = Clear, 02 = Cloudy, 03 = Fog/Smog/Smoke, 04 = Rain, 05 = Sleet/Hail (freezing rain or drizzle), 06 = Snow, 07 = Blowing Snow, 08 = Severe Crosswinds, 09 = Blowing Sand/Soil/Dirt, 88 = Other, 99 = Unknown
Weather 1 Code / Description	Describes the primary weather Condition	01 = Clear, 02 = Cloudy, 03 = Fog/Smoke/Smog, 04 = Rain, 05 = Sleet/Hail (freezing rain or drizzle), 06 = Snow, 07 = Blowing Snow, 08 = Severe Crosswinds, 09 = Slowing Sand/Soil/Dirt, 88 = Other, 99 = Unknown
Weather 2 Code / Description	Describes the secondary weather condition	Same as weather 1
Seatbelt Used	Indicates if seatbelts are used by all occupants in all the vehicles involved in the crash	No and Yes
Motorcycle Involved	Indicates whether motorcycle or scooter was involved in crash	No and Yes
Motorcycle Helmet Used	Indicates whether helmet was used by any person if the crash involved a motorcycle	No and Yes
Bicycle Involved	Indicates whether a bicycle was involved in the crash.	No and Yes
Bicycle Helmet Used	Indicates whether helmet was used by any person if the crash involved a bicycle.	No and Yes
Latitude	Geographic latitude of the crash given in decimal degrees.	Numerical value representing latitude.
Longitude	The geographic longitude of the crash, given in decimal degrees.	NUmerical calue representing longitude.
School bus involved code / description	Description of school but being involved in a crash	00 = No, 01 = Yes (school bus was directly involved), 02 = Yes (school bus was indirectly involved)
Work Zone	Specifies if the crash is in a construction, maintenance, or utility work zone or if it is related to activity within the work zone.	No and Yes
Workers Present	Specifies if workers were present in the work zone at the time of the crash	01 = No, 02 = Yes, 09 = Unknown
Work Zone Location Code / Description	Specifies the section of the work zone in which the crash occurs if it is in a work zone.	01 = Before the First Work Zone Warning Sign, 02 = Advance Warning Area, 03 = Transition Area, 04 = Activity Area, 05 = Termination Area
Work Zone Type Code / Description	Specifies t he type of work zone.	01 = Lane Closure, 02 = LaneShift/Crossover, 03 = Work on Shoulder or Median, 04 = Intermittent or Moving Work, 88 = Other
Primary Contributing Circumstance Code / Description	Describe the primary contributing circumstance	01 = Speeding, 02 = Failed to yield right of way, 03 = Passed Stop Sign, 04 = Disregard Traffic Signal, 05 = Wrong side or wrong way, 06 = Improper passing, 07 = Improper lane change, 08 = Following to close, 09 = Made improper turn, 10 = Driving under the influence, 11 = Driver inattention/distraction/fatigue, 12 = Driving in a careless or reckless manner, 13 = Driving in an aggressive manner, 14 = Improper backing, 15 = Other improper driving, 16 = Mechanical defects, 17 = Animal in roadway (Deer), 18 = Animal in roadway (other animal), 19 = Other environmental circumstances (weather, glare, visibility obstructed), 20 = Roadway circumstances (debris, holes, work zone), 21 = Pedestrian, 88 = Other, 99 = Unknown
County Code / Name	The name of the country in which the crash occurred.	K = Kent, N = New Castle, S = Sussex

II. Exploratory data analysis

A. Data Cleaning & Preprocessing

Our dataset featured discrepancies and missing values across many different columns which needed to be addressed. For discrepancies, the categorical variables typically had respective columns for their codes and their descriptions. We expected the same number of missing values for these column pairs. This was not the case, we found some rows missing codes or descriptions. This was resolved by mapping respective codes to descriptions and vice versa. There were invalid

entries like '31' and '32' found in our target variable CRASH CLASSIFICATION CODE which were removed. Other typos like '.3' and '3' were parsed to '03' for consistency.

For missing values, we generalized three severities of missing values. Each of these of cases were handled as follows to retain the most useful information: For columns missing less than 1% like CRASH DATETIME and CRASH CLASSIFICATION CODE, rows with missing entries were dropped. Columns between 1% and 5% like LIGHTING CONDITION and MANNER OF IMPACT, missing values were filled in with the most frequent category, dropping rows here would affect the distribution. Columns missing more than 5% were dropped entirely, which included WEATHER 2 DESCRIPTION with over 512,000 missing values and WORK ZONE columns with over 529,000 missing values, there is not enough useful information to work with and filling them would introduce bias.

B. Crash Class Distribution

Our analysis revealed a significant class imbalance in our target variable **Crash Classification Description/Code**. It can be noted that the **Fatality Crash** group causes a high imbalance, since it has much less data than the other three categories. This imbalance has implications for impacting model performance, and therefore is later considered in later resampling steps. Figures 1 and Figure 2 both display these class imbalances by a horizontal bar chart.

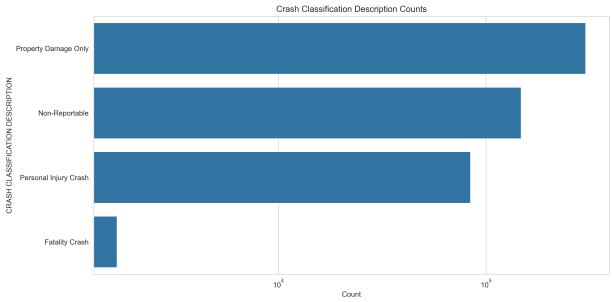


Fig. 1. Distribution of crash classifications. The horizontal bar plot shows the total number of crashes for each classification, with the x-axis scaled logarithmically.

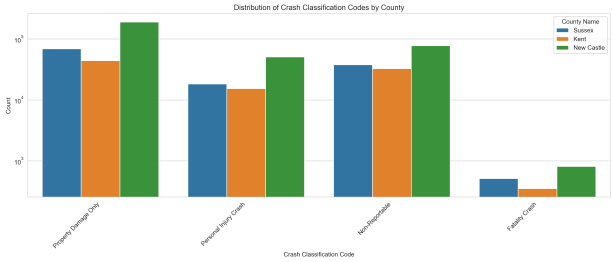


Fig. 2. Distribution of crash classification codes by county with the y-axis scaled logarithmically.

C. Temporal & External Trends

The crash class frequencies displayed temporal patterns related to time of day and year. Crashes were seen to be more frequent during dusk and nighttime, around the hours of 4-11 PM, which coincides with increasing traffic conditions and worsening lighting.

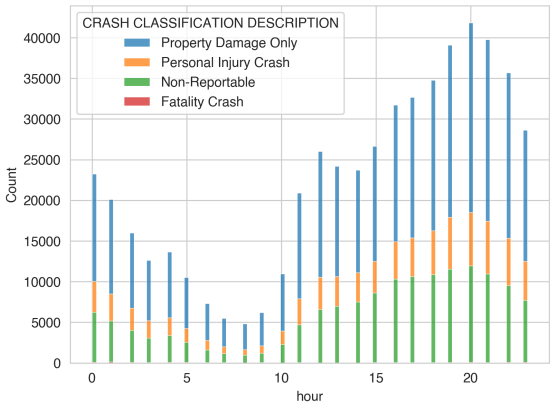


Fig. 3. Univariate distribution of crash counts over a 24-hour period, stratified by crash class classification.

Over the course of a 16-year period, the crash counts displayed in Figure 4 show an increase in property damage crashes as the years go on, peaking in 2018-2019 and then falling off in 2020, likely due to the pandemic. Non-reportable crashes have a slight decline over time, which may be due to the classification of what constitutes a reportable crash evolving over the years. Personal injury crashes stay relatively steady and fatality crashes are hardly displayed, yet stable, due to the class imbalance.

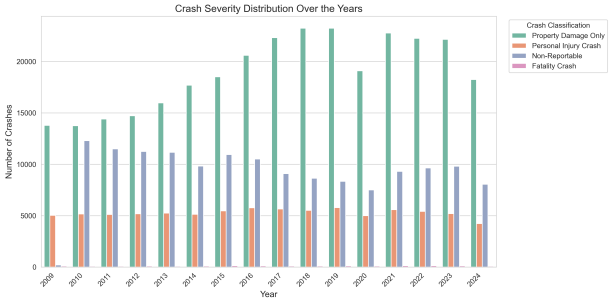


Fig. 4. Univariate distribution of crash counts over a 16-year period, stratified by crash class classification.

To explore further investigate temporal patterns in the data, we constructed a new column in the cleaned dataset to flag crash occurrences that took place around the six main U.S. holidays: New Year's Day, Memorial Day, Independence Day, Labor Day, Thanksgiving, and Christmas. For each holiday, a 5-day window was established—spanning two days before and after the holiday, including the holiday itself—to assess changes in crash frequency. While Figure 5 does not reveal

any clear temporal trends associated with these holidays, it does highlight the underlying class imbalance, even with a scaled y-axis. It can be noted that the binary categorization of crashes as simply inside or outside the holiday window may obscure finer-grained trends. A more interpretable approach could involve a zoomed-in view, such as plotting crash counts by individual days relative to each holiday (e.g., from –2 to +2), allowing for clearer comparison of pre-, during-, and post-holiday patterns.

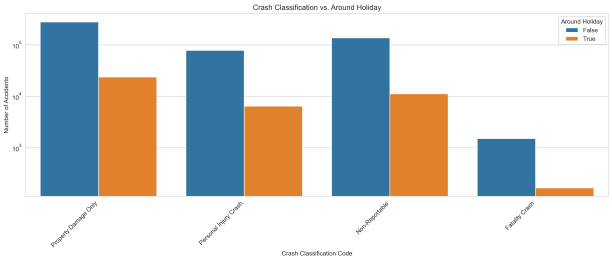


Fig. 5. Distribution of crash classification codes by whether the crash occurred around a holiday, with the y-axis scaled logarithmically. Holiday time frames are defined as a 5-day window centered around a significant US holiday.

D. Substance & Behavior Patterns

Heatmaps of substance and drug-related features indicated strong associations with crash severity. Each heatmap represents the conditional distribution of a specific crash type across alcohol and drug involvement combinations, where the four quadrants together sum to 100%. Notably, positive alcohol and drug involvement were disproportionately associated with *Fatality* and *Personal Injury* outcomes, especially when both are present. In contrast, crashes with no substance involvement were more likely to result in **Non-Reportable** or **Property Damage Only** classifications. The **Fatality Crash** category showed a significant shift, where crashes involving both alcohol and drugs occurred at a much higher percentage compared to substance-free cases. These patterns suggest that it may improve model performance in the future and will be explored further in the feature engineering phase.

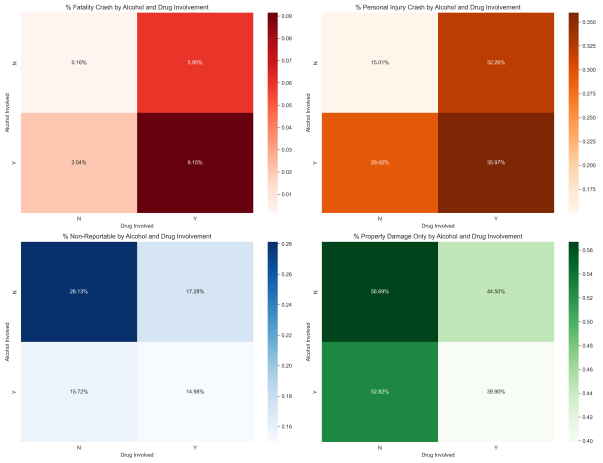


Fig. 6. Heatmap of alcohol and drug by crash classification. The heatmaps show the percentage of crashes within each classification that involved combinations of alcohol and drug involvement.

E. Environmental & Conditional Factors

Aside from the **Fatality Crash** category in Figure 7, the other three crash types exhibit similar trends in lighting conditions, with **Daylight** being the most frequent. However, this likely reflects that most driving occurs during daylight hours, rather than indicating a meaningful pattern. In contrast, the **Fatality Crash** category shows a slightly different distribution, with both **Daylight** and **Dark-Not Lighted** having nearly equal frequencies. This deviation may carry significance and will be revisited in later analysis.

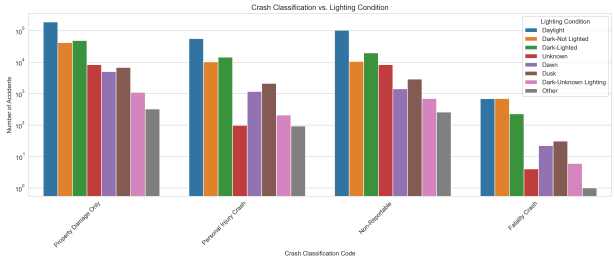


Fig. 7. Distribution of crash classification codes by lighting condition with the y-axis scaled logarithmically.

III. Methodology

A. Feature Engineering

1) Binary Mapping: Many features in the dataset were binary but stored as different categorical values such as ”Y”, ”YES”, ”N”, ”NO”, or ”0”/”1”. We applied a binary mapping strategy to convert these to a consistent numeric format, using the mapping: `binary_mapping = 'Y': 1, 'YES': 1, '1': 1, 'N': 0, 'NO': 0, '0': 0`. This will ensure that the models can interpret the binary variables correctly and consistently.

2) One-hot Encoding: For nominal categorical variables with more than two levels, we applied one-hot encoding to convert them into multiple binary columns, which allowed us

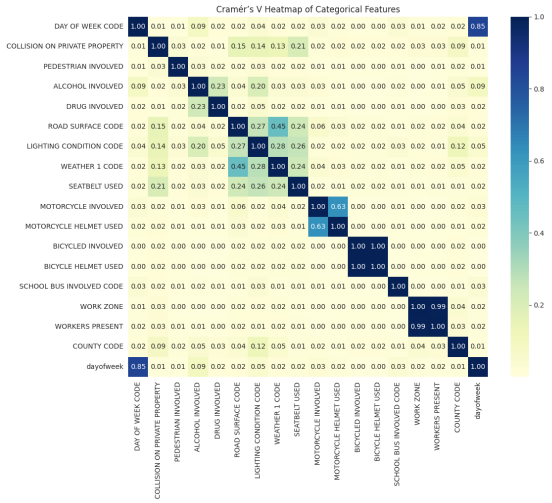


Fig. 8. Cramer's V correlation for Categorical Features

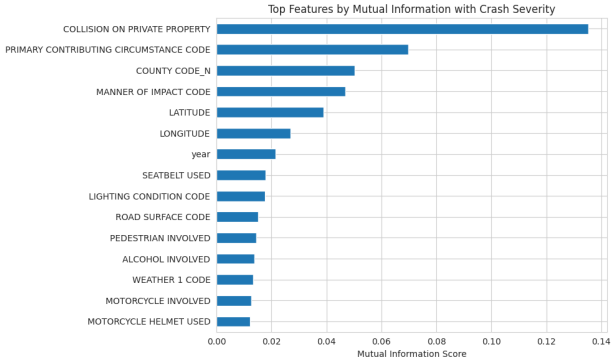


Fig. 9. Mutual Information with Crash Severity

to preserve the categorical information without imposing an artificial ordinal structure

3) Redundancy Filtering & Feature Relevance: To handle potential multicollinearity and identify most relevant features for our model training, we used Cramer's V and Mutual Information (MI) together.

Cramer's V was selected instead of Pearson correlation because majority of our features are categorical, and Pearson's coefficient is only appropriate for continuous and linear relationships. Cramer's V is a non-parametric design specifically for measuring association strength between categorical variables.

From Figure 8, we detected pairs of variables that were highly redundant. For instance,

- BICYCLE INVOLVED and BICYCLE HELMET USED had a perfect dependency of V = 1.00
- MOTORCYCLE INVOLVED and MOTORCYCLE HELMET USED had a moderately high dependency of V = 0.63
- WORK ZONE and WORKERS PRESENT also had a very high association of V= 0.99

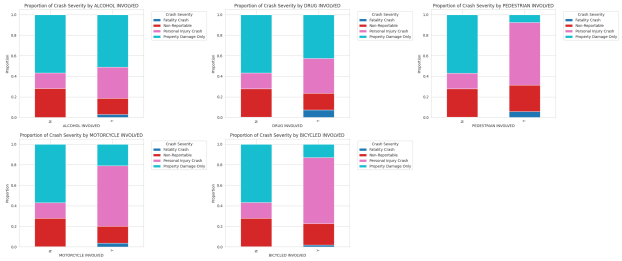


Fig. 10. Distribution of Proportions of Crash Classification

This will later be used to explain our best subset selection when modeling. We then used Mutual Information (MI) to evaluate how informative each feature was regarding the target variable. Unlike correlation, MI is more advantageous because it can capture both linear and non-linear associations and works with both categorical and numeric variables. This will allow us to see features that contributed the most predictive power. Figure 9 reveals that COLLISION ON PRIVATE PROPERTY, PRIMARY CONTRIBUTING CIRCUMSTANCES CODE, and COUNTY CODE are the top 3 contributors which makes sense since property and personal injury damage cases dominate the data.

Some features that are relevant in the real world, such as "ALCOHOL INVOLVED" and "DRUG INVOLVED," should logically be among the top contributors to our analysis; however, they were not. To investigate this further, we analyzed the distribution of these features in relation to our target variable like in Figure 10. We found that some of these features significantly contribute to our target variable by increasing the likelihood of fatal and personal injury crashes. This discrepancy is likely due to their rarity in the dataset rather than a lack of impact. As a result, we decided to retain these rare but important features for further analysis.

B. Dimensionality Reduction using PCA

To reduce the complexity of our dataset and understand the structure in a better way, we applied principal component analysis (PCA).

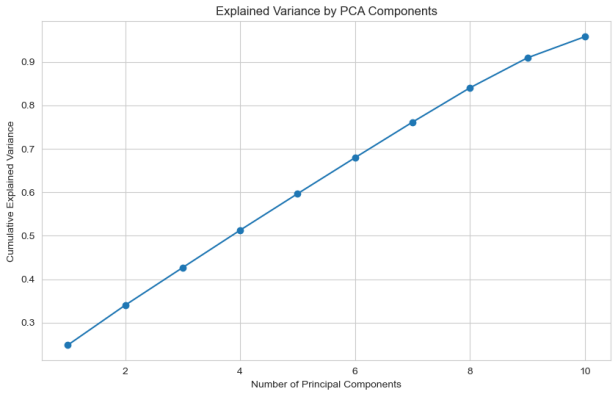


Fig. 11. Cumulative explained variance by the number of principal components.

Figure 11, the first ten principal components account for over 90% of the total variance. This justifies our use of the top components for dimensionality reduction. It indicates that a substantial amount of the dataset's structure can be preserved with fewer dimensions.

TABLE II
TOP CONTRIBUTING FEATURES TO PRINCIPAL COMPONENT 1

Feature	Absolute Loading (PC1)
WEATHER 1 CODE	0.531
ROAD SURFACE CODE	0.506
LIGHTING CONDITION CODE	0.484
MANNER OF IMPACT CODE	0.348
PRIMARY CONTRIBUTING CIRCUMSTANCE CODE	0.324

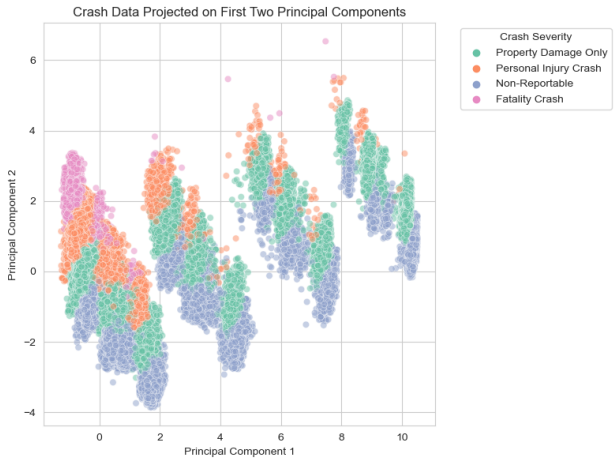


Fig. 12. Crash data projected onto the first two principal components.

Figure 12 displays a 2D scatterplot of the crash data projected onto the first two principal components. Each point is color coded for crash severity. We can see that non-reportable and property damage only categories show greater dispersion across the PCA space. This suggests a more diverse feature distribution.

C. Clustering

One-hot encoding would greatly expand our already large dataset, especially for columns with a high number of unique values. Leaving these columns alone would be problematic for model training times and memory usage. We used K-means clustering to condense the number of unique values for columns that had more than seven. The number of clusters for each high cardinality column was chosen using elbow plots and determining a cutoff for diminishing returns in within-cluster sum of squares. Figure 13 shows one of the many elbow plots. For ROAD SURFACE, we chose k to be 4.

Additionally, we engineered a geographic clustering feature by applying KMeans to the latitude and longitude of crash locations as seen in Figure 14. This allowed us to group incidents into meaningful regional clusters within Delaware to look at spatial patterns in car crashes.

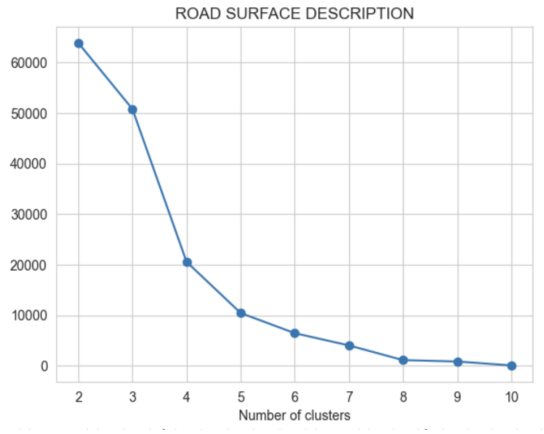


Fig. 13. ROAD SURFACE DESCRIPTION elbow plot.

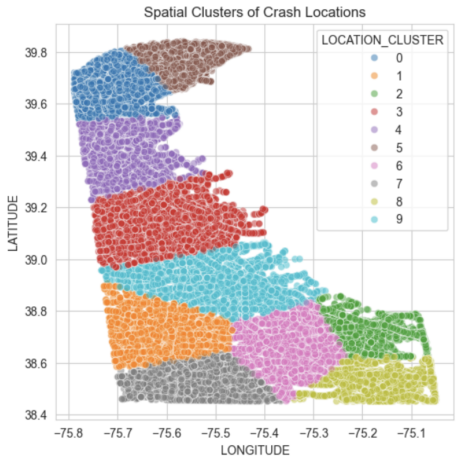


Fig. 14. Location clusters using k-means and latitude/longitude data.

D. Class Imbalance Handling

A preliminary analysis of our target variable, crash severity, revealed a significant class imbalance. Less severe outcomes, such as "Property Damage Only," constituted the vast majority of instances, while critical outcomes like "Personal Injury Crash" and "Fatality Crash" were comparatively rare. This imbalance poses a challenge as standard machine learning models tend to develop a bias towards the majority class, leading to poor predictive performance on the minority classes that are often of greatest interest.

To mitigate this issue, we explored three widely-used re-sampling and weighting techniques:

- **Undersampling:** This technique reduces the number of instances in the majority class to balance the class distribution. While it can force the model to pay more attention to minority classes, it risks discarding potentially useful information.
- **Oversampling (SMOTE):** We used the Synthetic Minority Over-sampling Technique (SMOTE), which creates new synthetic data points for the minority classes. Instead of simply duplicating existing data, SMOTE generates

new samples by interpolating between existing minority class instances, helping to create more diverse decision boundaries.
- **Class Weights:** This method involves adjusting the loss function during model training to penalize misclassifications of the minority classes more heavily than misclassifications of the majority class. This is done by assigning higher weights to the minority classes.

We evaluated these techniques using 3-fold cross-validation on a preliminary model. In our final results, SMOTE yielded the highest overall performance, notably increasing the F_1-score from a baseline of 0.4096 to 0.4379. Undersampling, while dramatically increasing recall, led to a severe drop in accuracy and the overall F_1-score, suggesting it was too aggressive for our dataset. The implications of these findings, particularly the trade-off between SMOTE's performance and its computational cost, will be discussed in the final model selection section.

IV. Experimental Results and Evaluation

A. Class Imbalance Handling

A preliminary analysis of our target variable, crash severity, revealed a significant class imbalance. Less severe outcomes, such as "Property Damage Only," constituted the vast majority of instances, while critical outcomes like "Personal Injury Crash" and "Fatality Crash" were comparatively rare. This imbalance poses a challenge as standard machine learning models tend to develop a bias towards the majority class, leading to poor predictive performance on the minority classes that are often of greatest interest.

To mitigate this issue, we explored three widely-used resampling and weighting techniques:

- **Undersampling:** This technique reduces the number of instances in the majority class to balance the class distribution. While it can force the model to pay more attention to minority classes, it risks discarding potentially useful information.
- **Oversampling (SMOTE):** We used the Synthetic Minority Over-sampling Technique (SMOTE), which creates new synthetic data points for the minority classes. Instead of simply duplicating existing data, SMOTE generates new samples by interpolating between existing minority class instances, helping to create more diverse decision boundaries.
- **Class Weights:** This method involves adjusting the loss function during model training to penalize misclassifications of the minority classes more heavily than misclassifications of the majority class. This is done by assigning higher weights to the minority classes.

We evaluated these techniques using 3-fold cross-validation on a preliminary model. As shown in Table III, SMOTE yielded the highest overall performance, notably increasing the F_1-score from a baseline of 0.4096 to 0.4379. Undersampling, while dramatically increasing recall, led to a severe drop in accuracy and the overall F_1-score, suggesting it was too aggressive for our dataset. The implications of these findings, particularly the trade-off between SMOTE's performance and its computational cost, will be discussed in the final model selection section.

TABLE III
Performance of Class Imbalance Handling Techniques

Method	Accuracy	Recall	F1-Score
Baseline	0.6324	0.3927	0.4096
Class Weights	0.6237	0.3847	0.4002
SMOTE	0.6538	0.4170	0.4379
Undersampling	0.4462	0.5118	0.3460

B. Modeling

Our modeling approach aimed to identify the most effective classifier for predicting crash severity from the prepared dataset. The process began by establishing a performance baseline using four distinct and widely-used classification algorithms. We utilized a 3-fold cross-validation strategy for all baseline model evaluations due to hardware and runtime constraints. This approach ensures that our performance estimates are robust and less prone to overfitting on a specific train-test split. The models evaluated were:

- **Extreme Gradient Boosting (XGBoost):** A powerful and efficient implementation of the gradient boosting framework, known for its high performance in classification tasks.
- **Logistic Regression:** A linear model that serves as a strong, interpretable baseline.
- **Random Forest:** An ensemble learning method that builds multiple decision trees and merges their outputs to improve predictive accuracy and control overfitting.
- **K-Nearest Neighbors (kNN):** A non-parametric, instance-based learning algorithm that classifies data points based on the majority class of their nearest neighbors.

The performance of each of these models, trained on the original, imbalanced dataset, is detailed in the following sections.

C. Performance of All Models

The following sections detail the baseline performance of each model on the test set. All models were trained without resampling techniques to establish a fair initial comparison. Performance is reported using accuracy, precision, recall, and the F_1-score, with a particular focus on the model's ability to correctly identify the minority classes.

1) XGBoost: Extreme Gradient Boosting (XGBoost) emerged as a top performer in terms of overall accuracy. The model achieved a test set accuracy of **67.84%**. It showed a strong tendency to correctly classify the majority class, "Property Damage Only," achieving a recall of 0.93. However, this high performance was not consistent across the more critical, rarer classes. The model's recall for "Personal Injury Crash" was only 0.15, and for "Fatality Crash," it was 0.16. This indicates that while the model is accurate overall, it fails

to identify over 84% of the most severe crash types, highlighting a significant weakness of the baseline configuration. The weighted-average F_1-score was 0.64, while the macro-average F_1-score was much lower at 0.46, further emphasizing the model's poor performance on the underrepresented classes.

2) Logistic Regression: The Logistic Regression model, serving as our linear baseline, achieved an overall accuracy of **67.12%**, only slightly lower than XGBoost. It exhibited a similar pattern of performance: a very high recall of 0.94 for "Property Damage Only" crashes, but extremely poor performance on severe incidents. Recall for "Personal Injury Crash" was 0.13, and for "Fatality Crash," it plummeted to a mere 0.03. This means the model correctly identified only 10 of the 335 fatal crashes in the test set. The extremely low macro-average F_1-score of 0.40 underscores the model's inability to generalize to the critical minority classes, likely due to its linear assumptions being insufficient to capture the complexity of the data.

3) Random Forest: The Random Forest classifier yielded a lower overall accuracy of **61.75%**. An interesting finding was its performance on "Personal Injury Crashes," where it achieved a recall of 0.27, notably higher than that of XGBoost or Logistic Regression in their baseline forms. This suggests the bagging mechanism of Random Forest might offer some inherent resilience to class imbalance. However, this came at the cost of lower precision (0.38) for that class, and the recall for "Fatality Crashes" remained low at 0.14. Given its significantly lower overall accuracy and weighted F_1-score (0.61), it was not considered a top contender.

4) K-Nearest Neighbors (kNN): The k-Nearest Neighbors model performed the poorest among the candidates, with a final accuracy of **61.66%**. As an instance-based learner, kNN is highly susceptible to the class distribution in the feature space. With the severe crash instances being sparse, the model's predictions were overwhelmingly influenced by the dense clusters of majority-class neighbors. This is reflected in its low recall values for severe incidents: 0.19 for "Personal Injury Crash" and a dismal 0.04 for "Fatality Crash." The low macro-average F_1-score of 0.40 confirmed its inadequacy for this specific problem.

D. Final Model Selection

After evaluating the baseline performance of all four models, we selected the final model based on overall predictive power. Table IV summarizes the key performance metric—accuracy—across all contenders.

TABLE IV
MODEL PERFORMANCE SUMMARY BY ACCURACY

Model	Accuracy
XGBoost	0.678442
Logistic Regression	0.671201
Random Forest	0.617492
K-Nearest Neighbors (kNN)	0.616597

With the highest overall accuracy, **XGBoost was chosen as the champion model** to move forward into the optimization phase. Although its baseline performance on minority classes was poor, its superior overall performance suggested it had the greatest potential to be improved.

A critical decision at this stage was how to address the class imbalance. As noted in Section 3.4, applying SMOTE yielded the best results in preliminary tests. However, it introduced significant computational overhead, with long runtimes and high memory consumption that made it unfeasible for our development cycle. We therefore made a strategic decision to forgo resampling and instead focus on optimizing the baseline XGBoost model through a more targeted approach: feature selection aimed at improving performance on the metrics that matter most.

E. Evaluation

The ultimate goal of a crash severity prediction system is not to maximize overall accuracy, but to minimize harm by correctly identifying the most dangerous incidents. The baseline model evaluations clearly demonstrated that optimizing for accuracy alone is insufficient and leads to a model that is practically useless for public safety applications. Consequently, the focus of our evaluation shifted from a general assessment to a specific, goal-oriented optimization of the selected XGBoost model.

Our primary objective became the maximization of **recall** for the "Personal Injury Crash" and "Fatality Crash" classes. A high recall in this context ensures that the model correctly identifies a high percentage of all actual severe crashes, minimizing the number of dangerous false negatives. To achieve this, we returned to the insights from our feature engineering section. We developed a strategy to train the XGBoost model on a **"best subset" of variables**. This subset was curated by:

1) Prioritizing features with high Mutual Information (MI) with respect to the target variable, such as `PRIMARY CONTRIBUTING CIRCUMSTANCE` and `SEATBELT USED`.
2) Eliminating highly redundant features identified via Cramer's V analysis (e.g., keeping only one feature from pairs like `WORK ZONE` and `WORKERS PRESENT`).
3) Intentionally retaining clinically important but rare features, such as `ALCOHOL INVOLVED`, whose impact might only be apparent when the model is no longer distracted by less relevant, noisy variables.

By training the XGBoost model on this refined, high-signal feature set, we aimed to create a more parsimonious and robust classifier. This approach forces the model to learn the more subtle patterns associated with severe crashes instead of relying on the easily learned patterns of the majority class. The desired outcome is a model with a more "balanced recall," where the sensitivity to critical minority classes is significantly improved, even if it comes at the expense of a modest decrease in overall accuracy. This trade-off is fundamental to developing a responsible and effective predictive tool for a real-world problem where the cost of a missed prediction can be catastrophic. The final performance of this optimized

model represents a more meaningful measure of success than the baseline accuracy figures.

V. Discussion

Our approach aimed to balance predictive performance, efficiency, and flexibility for future model development. The baseline XGBoost model, combined with feature subset selection, provided stable results and good recall for minority classes. However, we cannot overlook some limitations, such as class imbalance in the target variable, particularly the very few cases of "Fatality Crashes." Despite employing resampling techniques, there remains a risk of overfitting due to oversampling or potentially losing valuable data through undersampling, as well as dropping missing or incomplete data during our cleaning process.

Additionally, in terms of real-world relevance, traffic patterns and road conditions may change in the future, potentially disrupting our model's behavior and assumptions. Although XGBoost does not require strict assumptions, we have validated our input quality through multiple careful preprocessing checks.

A key challenge we faced was resource limitation. While SMOTE (Synthetic Minority Oversampling Technique) overall offered better performance, it was computationally intensive. Furthermore, real-world risk factors, such as alcohol and drug involvement, do not have a significant impact due to their low frequency. Other external factors, including road surface conditions, traffic volume, vehicle conditions, and human behavior, can directly contribute to the fatality of crashes.

Future work could build upon this foundation and enhance the analysis in several ways. For example, incorporating external data sources could provide a more comprehensive understanding of crash conditions. Real-time data on weather, traffic volume, road surface quality, and event calendars (e.g., holidays, sports events, concerts) could improve prediction accuracy and help explain variations in crash behavior that our current dataset does not capture. Additionally, expanding the dataset to include multiple states or multiple years would create a more diverse and balanced sample, enhancing model generalizability. Lastly, applying a combination of sampling techniques, such as SMOTE with class weighting or targeted undersampling, could more effectively address the severe class imbalance and improve model performance concerning rare but critical crash types.

VI. Conclusion

In this project, we aimed to predict the severity of traffic crashes using features that are known at the time of the accident. Through data cleaning, exploratory analysis, feature engineering, and modeling, we determined that the XGBoost classifier provided the best balance between performance and efficiency. Our analysis prioritized recall for severe crashes, reflecting the real-world importance of public safety.

By incorporating a thorough analysis of real-world relevance, we ensured that our models not only performed well but were also robust in a practical context. Despite some limitations, our findings offered valuable insights into the behavioral and environmental factors influencing crash outcomes.

Moving forward, integrating more comprehensive external data and expanding the analysis to larger datasets could help create even more robust and actionable predictive systems. This project enhanced our understanding of how machine learning can contribute to traffic safety and inform public policy.

FINAL_STA141C

June 9, 2025

1 Crash Dataset: Project

1.0.1 About the dataset:

- Dataset is from the department of homeland security.
- Delaware crash reports.
- 39 columns, 533115 entries.

1.0.2 Columns:

- Crash Date and Time: Date and time at which the crash occurred.
- Day of the week code: 01 - Sunday, 02 - Monday, 03 - Tuesday, 04 - Wednesday, 05 - Thursday, 06 - Friday, 07 - Saturday.
- Day of the week description: Day of the week the crash occurred.
- CRASH CLASSIFICATION CODE (TARGET VARIABLE): 01 - Non Reportable, 02 - Property damage only, 03 - personal injury crash, 04 - fatality crash.
- CRASH CLASSIFICATION DESCRIPTION (ALSO TARGET VARIABLE): check the crash classification description in the pdf.
- Collision on private property: No or Yes.
- Pedestrian involved: indicates if a pedestrian was involved.
- Manner of impact code: 01 - Front to rear, 02 - Front to front, 03 - Angle, 04 - Sideswipe (same direction), 05 - Sideswipe (opposite direction), 06 - Rear to side, 07 - Rear to rear, 88 - Other, 99 - Unknown, 00 - Not a collision between 2 vehicles.
- Manner of impact description: gives the description.
- Alcohol involved: Yes/No
- Drug Involved: Yes/No
- Road surface code: 01 - Dry, 02 - Wet, 03 - Snow, 04 - Ice/Frost, 05 - Sand, 06 - Water (standing/moving), 07 - Slush, 08 - Mud/dirt/gravel, 88- other, 99 - unknown.
- Lighting condition description: description.
- Weather 1 code: 01 - Clear, 02 - Cloudy, 03 - Fog/smoke/smog, 04 - Rain, 05 - sleet/hail (Freezing rain or drizzle), 06 - Snow, 07 - Blowing Snow, 08 - Severe Crosswinds, 09 - Blowing Sand/Soil/Dirt, 88 - Other, 99 - Unknown.
- Weather 1 description - Description.
- Weather 2 code: same thing as weather 1 (secondary weather condition)
- Weather 2 description: Same as weather 1
- Seatbelt used: Yes/No
- Motorcycle involved: Yes/No
- Motorcycle Helmet Used: Yes/No (Indicates whether a helmet was used by the people involved in the crash if a motorcycle or scooter was used.)

- Bicycle involved: Yes/No
- Bicycle helmet used: Yes/No
- Latitude: Geographic latitude for the crash given in decimal degrees.
- Longitude: Geographic longitude for the crash given in decimal degrees.
- Primary Contributing Circumstance Code: 01 - Speeding, 02 - failed to yield right of way, Passed stop sign, 04 - disregard traffic signal, 05 - wrong side or wrong way,

[4]:
```
import pandas as pd
```

[5]:
```
crash_df = pd.read_csv('Public_Crash_Data_20250529.csv', low_memory=False)
```

[6]:
```
print(crash_df.columns.tolist())
```

```
['CRASH DATETIME', 'DAY OF WEEK CODE', 'DAY OF WEEK DESCRIPTION', 'CRASH
CLASSIFICATION CODE', 'CRASH CLASSIFICATION DESCRIPTION', 'COLLISION ON PRIVATE
PROPERTY', 'PEDESTRIAN INVOLVED', 'MANNER OF IMPACT CODE', 'MANNER OF IMPACT
DESCRIPTION', 'ALCOHOL INVOLVED', 'DRUG INVOLVED', 'ROAD SURFACE CODE', 'ROAD
SURFACE DESCRIPTION', 'LIGHTING CONDITION CODE', 'LIGHTING CONDITION
DESCRIPTION', 'WEATHER 1 CODE', 'WEATHER 1 DESCRIPTION', 'WEATHER 2 CODE',
'WEATHER 2 DESCRIPTION', 'SEATBELT USED', 'MOTORCYCLE INVOLVED', 'MOTORCYCLE
HELMET USED', 'BICYCLED INVOLVED', 'BICYCLE HELMET USED', 'LATITUDE',
'LONGITUDE', 'PRIMARY CONTRIBUTING CIRCUMSTANCE CODE', 'PRIMARY CONTRIBUTING
CIRCUMSTANCE DESCRIPTION', 'SCHOOL BUS INVOLVED CODE', 'SCHOOL BUS INVOLVED
DESCRIPTION', 'WORK ZONE', 'WORK ZONE LOCATION CODE', 'WORK ZONE LOCATION
DESCRIPTION', 'WORK ZONE TYPE CODE', 'WORK ZONE TYPE DESCRIPTION', 'WORKERS
PRESENT', 'the_geom', 'COUNTY CODE', 'COUNTY NAME']
```

[7]:
```
print(crash_df.dtypes)
```

```
CRASH DATETIME                          object
DAY OF WEEK CODE                         int64
DAY OF WEEK DESCRIPTION                 object
CRASH CLASSIFICATION CODE               object
CRASH CLASSIFICATION DESCRIPTION        object
COLLISION ON PRIVATE PROPERTY           object
PEDESTRIAN INVOLVED                     object
MANNER OF IMPACT CODE                  float64
MANNER OF IMPACT DESCRIPTION            object
ALCOHOL INVOLVED                        object
DRUG INVOLVED                           object
ROAD SURFACE CODE                      float64
ROAD SURFACE DESCRIPTION                object
LIGHTING CONDITION CODE                float64
LIGHTING CONDITION DESCRIPTION          object
WEATHER 1 CODE                         float64
WEATHER 1 DESCRIPTION                   object
WEATHER 2 CODE                         float64
WEATHER 2 DESCRIPTION                   object
SEATBELT USED                           object
```

```
MOTORCYCLE INVOLVED                                 object
MOTORCYCLE HELMET USED                              object
BICYCLED INVOLVED                                   object
BICYCLE HELMET USED                                 object
LATITUDE                                           float64
LONGITUDE                                          float64
PRIMARY CONTRIBUTING CIRCUMSTANCE CODE             float64
PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION       object
SCHOOL BUS INVOLVED CODE                           float64
SCHOOL BUS INVOLVED DESCRIPTION                     object
WORK ZONE                                           object
WORK ZONE LOCATION CODE                            float64
WORK ZONE LOCATION DESCRIPTION                      object
WORK ZONE TYPE CODE                                float64
WORK ZONE TYPE DESCRIPTION                          object
WORKERS PRESENT                                     object
the_geom                                            object
COUNTY CODE                                         object
COUNTY NAME                                         object
dtype: object
```

[8]:
```
# checking the missing values
crash_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 536614 entries, 0 to 536613
Data columns (total 39 columns):
 #   Column                                  Non-Null Count   Dtype
---  ------                                  --------------   -----
 0   CRASH DATETIME                          536611 non-null  object
 1   DAY OF WEEK CODE                        536614 non-null  int64
 2   DAY OF WEEK DESCRIPTION                 536614 non-null  object
 3   CRASH CLASSIFICATION CODE               536611 non-null  object
 4   CRASH CLASSIFICATION DESCRIPTION        536585 non-null  object
 5   COLLISION ON PRIVATE PROPERTY           536614 non-null  object
 6   PEDESTRIAN INVOLVED                     536614 non-null  object
 7   MANNER OF IMPACT CODE                   516360 non-null  float64
 8   MANNER OF IMPACT DESCRIPTION            516357 non-null  object
 9   ALCOHOL INVOLVED                        536614 non-null  object
 10  DRUG INVOLVED                           536614 non-null  object
 11  ROAD SURFACE CODE                       515119 non-null  float64
 12  ROAD SURFACE DESCRIPTION                515119 non-null  object
 13  LIGHTING CONDITION CODE                 515580 non-null  float64
 14  LIGHTING CONDITION DESCRIPTION          515580 non-null  object
 15  WEATHER 1 CODE                          514724 non-null  float64
 16  WEATHER 1 DESCRIPTION                   514724 non-null  object
 17  WEATHER 2 CODE                          20291 non-null   float64
 18  WEATHER 2 DESCRIPTION                   20291 non-null   object
```

```
 19  SEATBELT USED                                    536614 non-null  object
 20  MOTORCYCLE INVOLVED                              536614 non-null  object
 21  MOTORCYCLE HELMET USED                           536614 non-null  object
 22  BICYCLED INVOLVED                                536614 non-null  object
 23  BICYCLE HELMET USED                              536614 non-null  object
 24  LATITUDE                                         536614 non-null  float64
 25  LONGITUDE                                        536614 non-null  float64
 26  PRIMARY CONTRIBUTING CIRCUMSTANCE CODE           519272 non-null  float64
 27  PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION    519271 non-null  object
 28  SCHOOL BUS INVOLVED CODE                         536599 non-null  float64
 29  SCHOOL BUS INVOLVED DESCRIPTION                  536599 non-null  object
 30  WORK ZONE                                        536614 non-null  object
 31  WORK ZONE LOCATION CODE                          3753 non-null    float64
 32  WORK ZONE LOCATION DESCRIPTION                   3753 non-null    object
 33  WORK ZONE TYPE CODE                              3755 non-null    float64
 34  WORK ZONE TYPE DESCRIPTION                       3755 non-null    object
 35  WORKERS PRESENT                                  536614 non-null  object
 36  the_geom                                         536614 non-null  object
 37  COUNTY CODE                                      536614 non-null  object
 38  COUNTY NAME                                      536614 non-null  object
dtypes: float64(11), int64(1), object(27)
memory usage: 159.7+ MB
```

[9]:
```
print(crash_df.isnull().sum())
```

```
CRASH DATETIME                                3
DAY OF WEEK CODE                              0
DAY OF WEEK DESCRIPTION                       0
CRASH CLASSIFICATION CODE                     3
CRASH CLASSIFICATION DESCRIPTION             29
COLLISION ON PRIVATE PROPERTY                 0
PEDESTRIAN INVOLVED                           0
MANNER OF IMPACT CODE                     20254
MANNER OF IMPACT DESCRIPTION              20257
ALCOHOL INVOLVED                              0
DRUG INVOLVED                                 0
ROAD SURFACE CODE                         21495
ROAD SURFACE DESCRIPTION                  21495
LIGHTING CONDITION CODE                   21034
LIGHTING CONDITION DESCRIPTION            21034
WEATHER 1 CODE                            21890
WEATHER 1 DESCRIPTION                     21890
WEATHER 2 CODE                           516323
WEATHER 2 DESCRIPTION                    516323
SEATBELT USED                                 0
MOTORCYCLE INVOLVED                           0
MOTORCYCLE HELMET USED                        0
BICYCLED INVOLVED                             0
```

```
BICYCLE HELMET USED                                  0
LATITUDE                                             0
LONGITUDE                                            0
PRIMARY CONTRIBUTING CIRCUMSTANCE CODE           17342
PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION    17343
SCHOOL BUS INVOLVED CODE                            15
SCHOOL BUS INVOLVED DESCRIPTION                     15
WORK ZONE                                            0
WORK ZONE LOCATION CODE                         532861
WORK ZONE LOCATION DESCRIPTION                  532861
WORK ZONE TYPE CODE                             532859
WORK ZONE TYPE DESCRIPTION                      532859
WORKERS PRESENT                                      0
the_geom                                             0
COUNTY CODE                                          0
COUNTY NAME                                          0
dtype: int64
```

[10]:
```
print(crash_df['CRASH CLASSIFICATION DESCRIPTION'].unique())
```

```
['Property Damage Only' 'Personal Injury Crash' 'Non-Reportable'
 'Fatality Crash' nan]
```

[11]:
```
print(crash_df['LIGHTING CONDITION DESCRIPTION'].unique())
```

```
['Daylight' nan 'Dark-Not Lighted' 'Dark-Lighted' 'Unknown' 'Dawn' 'Dusk'
 'Dark-Unknown Lighting' 'Other']
```

[12]:
```
print(crash_df['WORK ZONE TYPE DESCRIPTION'].unique())
```

```
[nan 'Lane Shift/Crossover' 'Lane Closure' 'Intermittent or Moving Work'
 'Other' 'Work on Shoulder or Median']
```

[13]:
```
print(crash_df['PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION'].unique())
```

```
['Following too close' 'Driving in a careless or reckless manner'
 'Unknown' 'Driver inattention, distraction, or fatigue' 'Other'
 'Failed to yield right of way'
 'Other environmental circumstances - weather, glare' 'Made improper turn'
 'Roadway circumstances - debris, holes, work zone,'
 'Driving under the influence' 'Improper lane change'
 'Driving in an aggressive manner' nan 'Passed Stop Sign'
 'Disregard Traffic Signal' 'Other improper driving' 'Improper backing'
 'Animal in Roadway - Deer' 'Wrong side or wrong way'
 'Animal in Roadway - Other Animal' 'Mechanical defects' 'Speeding'
 'Improper passing' 'Pedestrian']
```

[14]:
```
print(crash_df['WEATHER 1 DESCRIPTION'].unique())
```

```
['Clear' 'Cloudy' nan 'Rain' 'Snow' 'Unknown' 'Blowing Snow'
```

```
 'Severe Crosswinds' 'Fog, Smog, Smoke' 'Other'
 'Sleet, Hail (freezing rain or drizzle)' 'Blowing Sand, Soil, Dirt']
```

[15]:
```
print(crash_df['WEATHER 2 DESCRIPTION'].unique())
```

```
[nan 'Cloudy' 'Unknown' 'Severe Crosswinds' 'Clear' 'Blowing Snow' 'Rain'
 'Blowing Sand, Soil, Dirt' 'Fog, Smog, Smoke'
 'Sleet, Hail (freezing rain or drizzle)' 'Other' 'Snow']
```

[16]:
```
print(crash_df['WORK ZONE LOCATION DESCRIPTION'].unique())
```

```
[nan 'Advance Warning Area' 'Transition Area' 'Activity Area'
 'Before the First Work Zone Warning Sign' 'Termination Area']
```

[17]:
```
print(crash_df['MANNER OF IMPACT DESCRIPTION'].unique())
```

```
['Front to rear' 'Angle' 'Unknown' 'Sideswipe, same direction' nan
 'Not a collision between two vehicles' 'Front to front'
 'Sideswipe, opposite direction' 'Rear to rear' 'Rear to side' 'Other']
```

[18]:
```
print(crash_df['SCHOOL BUS INVOLVED DESCRIPTION'].unique())
```

```
['No' 'Yes, Directly Involved' 'Yes, Indirectly Involved' nan]
```

[19]:
```
crash_df['SEATBELT USED'].value_counts()
```

[19]:
```
SEATBELT USED
Y    516468
N     20146
Name: count, dtype: int64
```

[20]:
```
crash_df['DRUG INVOLVED'].value_counts()
```

[20]:
```
DRUG INVOLVED
N    530072
Y      6542
Name: count, dtype: int64
```

[21]:
```
crash_df['CRASH CLASSIFICATION DESCRIPTION'].value_counts()
```

[21]:
```
CRASH CLASSIFICATION DESCRIPTION
Property Damage Only     302594
Non-Reportable           147931
Personal Injury Crash     84382
Fatality Crash             1678
Name: count, dtype: int64
```

[22]:
```
crash_df['PEDESTRIAN INVOLVED'].value_counts()
```

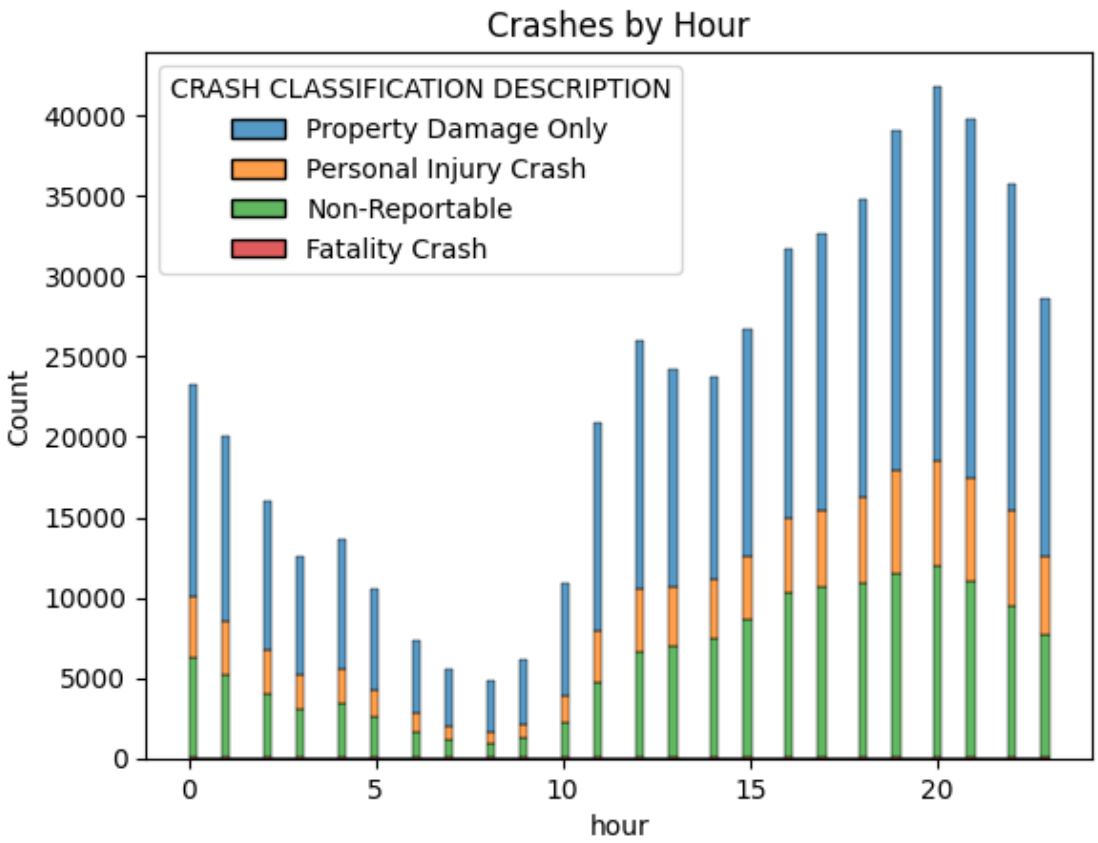
[22]:
```
PEDESTRIAN INVOLVED
N    529519
Y      7095
Name: count, dtype: int64
```

[23]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

[24]:
```
crash_df['CRASH DATETIME'] = pd.to_datetime(crash_df['CRASH DATETIME'])
crash_df['hour'] = crash_df['CRASH DATETIME'].dt.hour
crash_df['dayofweek'] = crash_df['CRASH DATETIME'].dt.day_name()
crash_df['month'] = crash_df['CRASH DATETIME'].dt.month

# Plot by hour
sns.histplot(data=crash_df, x='hour', hue='CRASH CLASSIFICATION DESCRIPTION',␣
 ↪multiple = 'stack')
plt.title("Crashes by Hour")
```

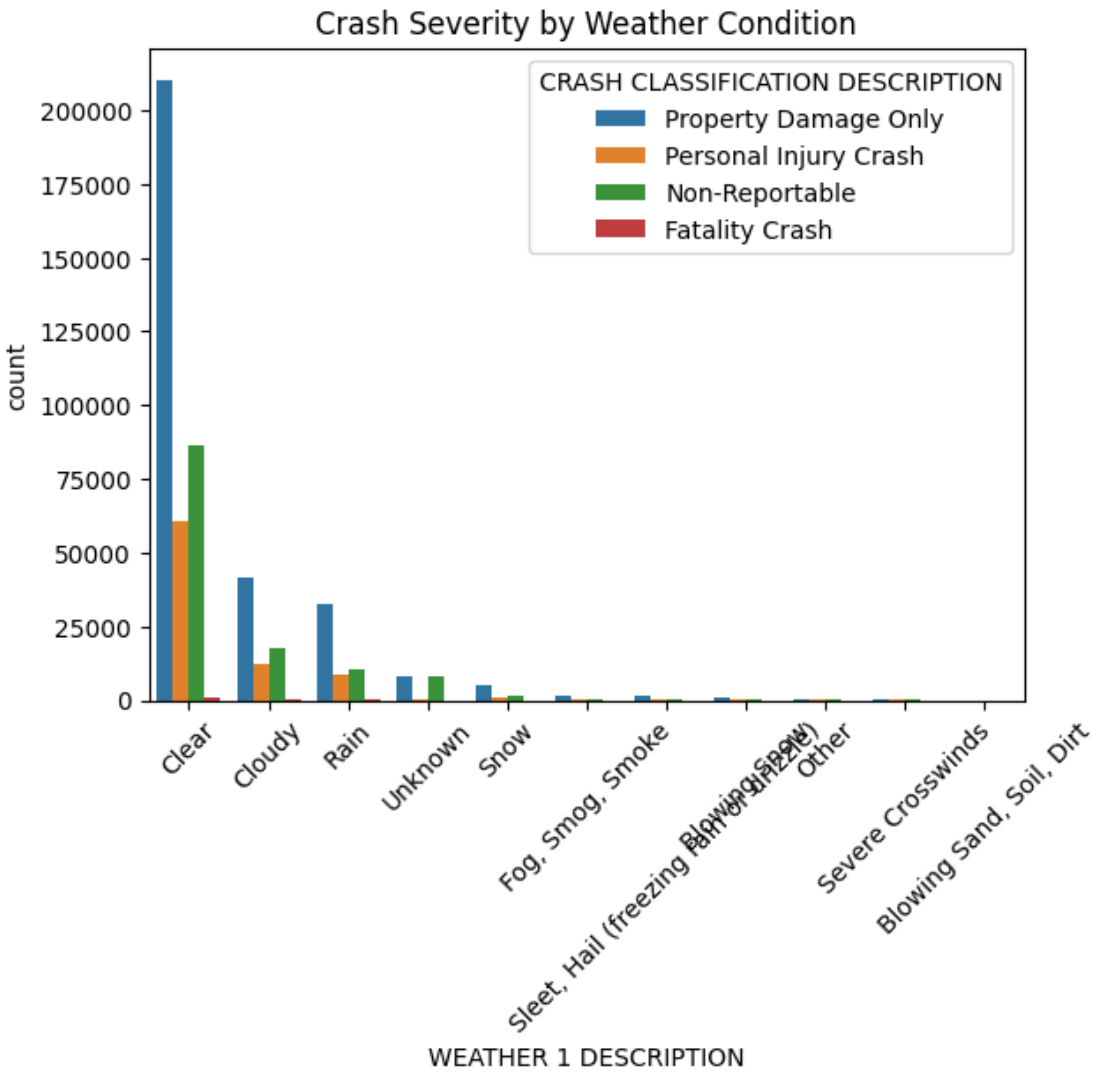
```
/var/folders/1c/47fsk5z56lq4c2qq6c1rfj7r0000gn/T/ipykernel_24863/2124001811.py:1
: UserWarning: Could not infer format, so each element will be parsed
individually, falling back to `dateutil`. To ensure parsing is consistent and
as-expected, please specify a format.
  crash_df['CRASH DATETIME'] = pd.to_datetime(crash_df['CRASH DATETIME'])
```

[24]:
```
Text(0.5, 1.0, 'Crashes by Hour')
```



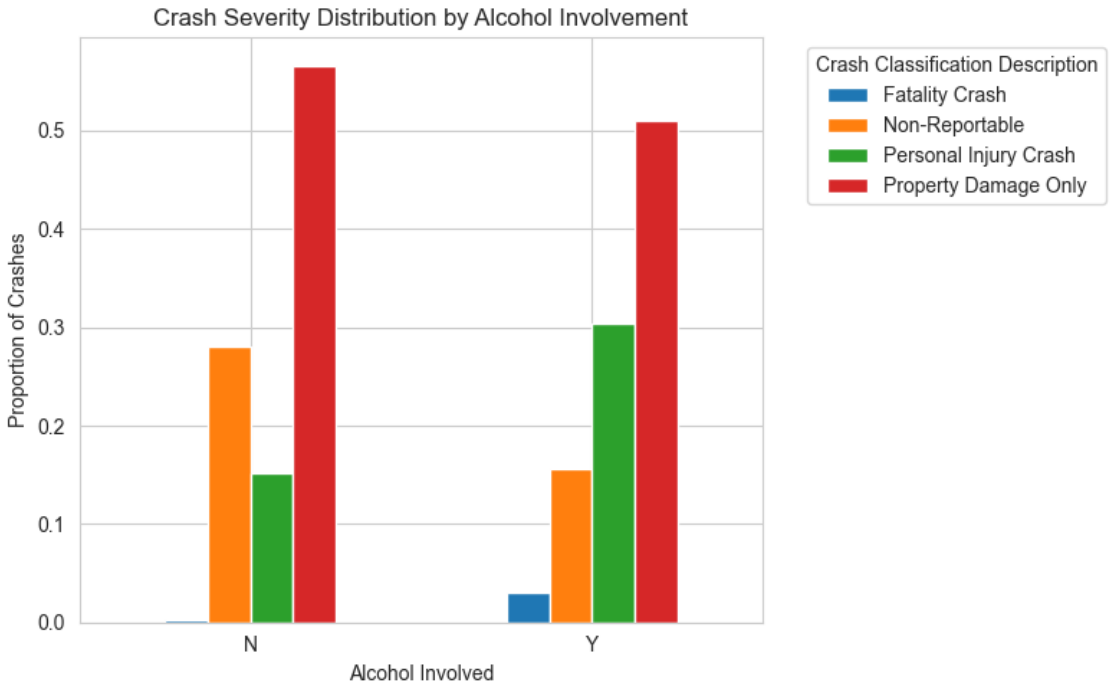
[25]:
```
weather_order = crash_df['WEATHER 1 DESCRIPTION'].value_counts().index
sns.countplot(data=crash_df, x='WEATHER 1 DESCRIPTION', hue='CRASH␣
 ↪CLASSIFICATION DESCRIPTION', order=weather_order)
plt.xticks(rotation=45)
plt.title("Crash Severity by Weather Condition")
```

[25]:
```
Text(0.5, 1.0, 'Crash Severity by Weather Condition')
```



[26]:
```
import matplotlib.pyplot as plt
sns.set_style("whitegrid")

# Your normalized DataFrame
normalized = crash_df.groupby('ALCOHOL INVOLVED')['CRASH CLASSIFICATION␣
 ↪DESCRIPTION']\
                    .value_counts(normalize=True).unstack()

# Plot with enhancements
ax = normalized.plot(kind='bar', stacked=False, figsize=(8, 5))

# Label the y-axis
ax.set_ylabel("Proportion of Crashes")
```

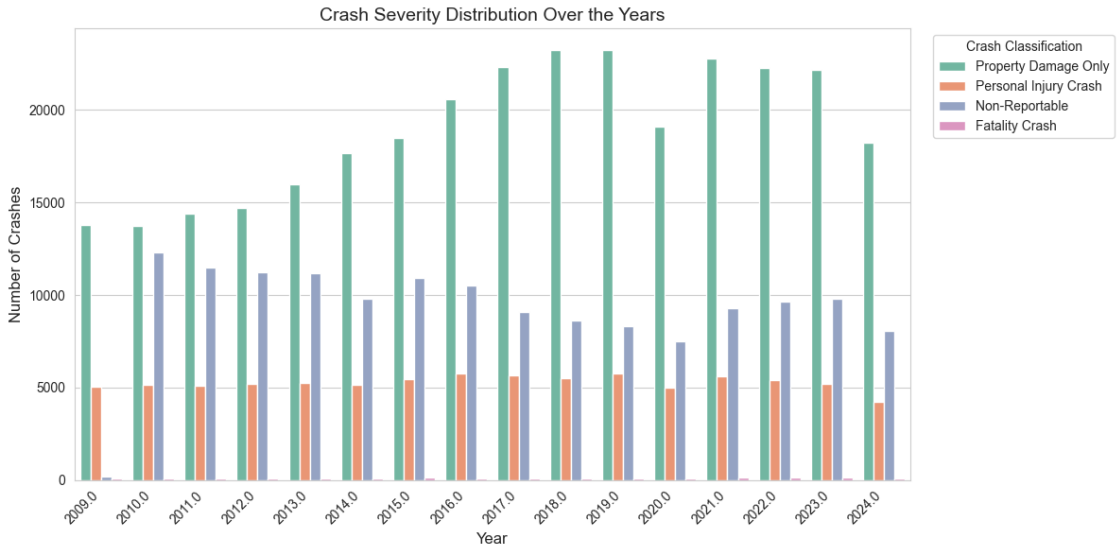
```
# Label the x-axis
ax.set_xlabel("Alcohol Involved")

# Title (optional)
ax.set_title("Crash Severity Distribution by Alcohol Involvement")

# Rotate x-axis labels (if needed)
plt.xticks(rotation=0)

# Improve legend
plt.legend(title='Crash Classification Description', bbox_to_anchor=(1.05, 1),␣
 ↪loc='upper left')

plt.tight_layout()
plt.show()
```



[27]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Ensure 'year' column exists
crash_df['CRASH DATETIME'] = pd.to_datetime(crash_df['CRASH DATETIME'],␣
 ↪errors='coerce')
crash_df['year'] = crash_df['CRASH DATETIME'].dt.year
```

```
# Drop rows with missing year just in case
crash_df = crash_df.dropna(subset=['year'])

# Plot
plt.figure(figsize=(12, 6))
sns.set_style("whitegrid")

ax = sns.countplot(data=crash_df, x='year', hue='CRASH CLASSIFICATION␣
 ↪DESCRIPTION', palette='Set2')

ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
ax.set_xlabel("Year", fontsize=12)
ax.set_ylabel("Number of Crashes", fontsize=12)
ax.set_title("Crash Severity Distribution Over the Years", fontsize=14)
plt.legend(title="Crash Classification", bbox_to_anchor=(1.02, 1), loc='upper␣
 ↪left')
plt.tight_layout()
plt.show()
```

```
/var/folders/1c/47fsk5z56lq4c2qq6c1rfj7r0000gn/T/ipykernel_24863/1556234572.py:1
7: UserWarning: set_ticklabels() should only be used with a fixed number of
ticks, i.e. after set_ticks() or using a FixedLocator.
  ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
```



[28]:
```
!pip install folium
import folium
from folium.plugins import HeatMap
```

```
sample = crash_df[crash_df['CRASH CLASSIFICATION DESCRIPTION'] == 'Fatality␣
 ↪Crash'][['LATITUDE', 'LONGITUDE']].dropna()
m = folium.Map(location=[sample['LATITUDE'].mean(), sample['LONGITUDE'].
 ↪mean()], zoom_start=8)
HeatMap(data=sample).add_to(m)
m.save('fatality_heatmap.html')
```

```
Requirement already satisfied: folium in /opt/anaconda3/lib/python3.12/site-
packages (0.19.5)
Requirement already satisfied: branca>=0.6.0 in
/opt/anaconda3/lib/python3.12/site-packages (from folium) (0.8.1)
Requirement already satisfied: jinja2>=2.9 in
/opt/anaconda3/lib/python3.12/site-packages (from folium) (3.1.4)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.12/site-
packages (from folium) (1.26.4)
Requirement already satisfied: requests in /opt/anaconda3/lib/python3.12/site-
packages (from folium) (2.32.3)
Requirement already satisfied: xyzservices in
/opt/anaconda3/lib/python3.12/site-packages (from folium) (2022.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/opt/anaconda3/lib/python3.12/site-packages (from jinja2>=2.9->folium) (2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/anaconda3/lib/python3.12/site-packages (from requests->folium) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/opt/anaconda3/lib/python3.12/site-packages (from requests->folium) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/anaconda3/lib/python3.12/site-packages (from requests->folium) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/anaconda3/lib/python3.12/site-packages (from requests->folium) (2025.4.26)
```

[29]:
```
'''
Some columns have few missing values, some have many
There also some discrepancies between respective codes and descriptions

We will proceed as follows:
Missing 0% ---> Do nothing
Missing < 1% ---> Drop rows with missing values (3)
Missing < 5% ---> Fill in with the mode/most frequent value (4)
Missing > 5% ---> Drop columns entirely (2)

Convert missing codes to descriptions or vice versa (1)
'''
```

[29]:
```
'\nSome columns have few missing values, some have many\nThere also some
discrepancies between respective codes and descriptions\n\nWe will proceed as
follows:\nMissing 0% ---> Do nothing\nMissing < 1% ---> Drop rows with missing
values (3)\nMissing < 5% ---> Fill in with the mode/most frequent value
```

(4)\nMissing > 5% ---> Drop columns entirely (2)\n\nConvert missing codes to descriptions or vice versa (1)\n'

[30]:
```
'''
Strange values in columns with discrepancies:
CRASH CODE - 00, 33, 32, 31, also needs a lot of standardization
IMPACT CODE - 30
PRIMARY CIRCUMSTANCE CODE  - 43

These need to be handled/removed before converting codes-to-descriptions/
 ↪descriptions-to-codes
'''
```

[30]: '\nStrange values in columns with discrepancies:\nCRASH CODE - 00, 33, 32, 31, also needs a lot of standardization\nIMPACT CODE - 30\nPRIMARY CIRCUMSTANCE CODE - 43\n\nThese need to be handled/removed before converting codes-to-descriptions/descriptions-to-codes\n'

[31]:
```
pd.set_option('display.max_columns', 39)
crash_df[crash_df['PRIMARY CONTRIBUTING CIRCUMSTANCE CODE'] == 43.0]
```

[31]:
```
                CRASH DATETIME  DAY OF WEEK CODE DAY OF WEEK DESCRIPTION  \
14180 2021-05-21 19:00:00+00:00                 6                  Friday

      CRASH CLASSIFICATION CODE CRASH CLASSIFICATION DESCRIPTION  \
14180                        03            Personal Injury Crash

      COLLISION ON PRIVATE PROPERTY PEDESTRIAN INVOLVED  \
14180                             N                   N

       MANNER OF IMPACT CODE MANNER OF IMPACT DESCRIPTION ALCOHOL INVOLVED  \
14180                    1.0                Front to rear                N

      DRUG INVOLVED  ROAD SURFACE CODE ROAD SURFACE DESCRIPTION  \
14180             N                1.0                      Dry

       LIGHTING CONDITION CODE LIGHTING CONDITION DESCRIPTION  WEATHER 1 CODE  \
14180                      1.0                       Daylight             1.0

      WEATHER 1 DESCRIPTION  WEATHER 2 CODE WEATHER 2 DESCRIPTION  …  \
14180                 Clear             NaN                   NaN  …

       LATITUDE LONGITUDE PRIMARY CONTRIBUTING CIRCUMSTANCE CODE  \
14180  38.51861 -75.05519                                  43.0

      PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION SCHOOL BUS INVOLVED CODE  \
14180                                           NaN                      0.0
```

```
       SCHOOL BUS INVOLVED DESCRIPTION  WORK ZONE  WORK ZONE LOCATION CODE  \
14180                               No          N                      NaN

      WORK ZONE LOCATION DESCRIPTION  WORK ZONE TYPE CODE  \
14180                            NaN                  NaN

      WORK ZONE TYPE DESCRIPTION WORKERS PRESENT  \
14180                        NaN               N

                                     the_geom COUNTY CODE  COUNTY NAME  \
14180  POINT (-75.055190000068 38.518610000202)           S       Sussex

       hour dayofweek month    year
14180  19.0    Friday   5.0  2021.0

[1 rows x 43 columns]
```

[32]:
```
crash_df[crash_df['MANNER OF IMPACT CODE'] == 30.0]
```

[32]:
```
                  CRASH DATETIME  DAY OF WEEK CODE DAY OF WEEK DESCRIPTION  \
380163 2018-05-22 20:52:00+00:00                 3                  Tuesday

       CRASH CLASSIFICATION CODE CRASH CLASSIFICATION DESCRIPTION  \
380163                        02             Property Damage Only

       COLLISION ON PRIVATE PROPERTY PEDESTRIAN INVOLVED  \
380163                             N                   N

        MANNER OF IMPACT CODE MANNER OF IMPACT DESCRIPTION ALCOHOL INVOLVED  \
380163                   30.0                          NaN                N

       DRUG INVOLVED  ROAD SURFACE CODE ROAD SURFACE DESCRIPTION  \
380163             N                1.0                      Dry

        LIGHTING CONDITION CODE LIGHTING CONDITION DESCRIPTION  \
380163                      1.0                       Daylight

        WEATHER 1 CODE WEATHER 1 DESCRIPTION  WEATHER 2 CODE  \
380163             1.0                 Clear             NaN

       WEATHER 2 DESCRIPTION  … LATITUDE LONGITUDE  \
380163                   NaN  …  38.7585  -75.6203

       PRIMARY CONTRIBUTING CIRCUMSTANCE CODE  \
380163                                   18.0
```

```
       PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION SCHOOL BUS INVOLVED CODE  \
380163              Animal in Roadway - Other Animal                       0.0

        SCHOOL BUS INVOLVED DESCRIPTION  WORK ZONE  WORK ZONE LOCATION CODE  \
380163                               No          N                      NaN

       WORK ZONE LOCATION DESCRIPTION  WORK ZONE TYPE CODE  \
380163                            NaN                  NaN

       WORK ZONE TYPE DESCRIPTION WORKERS PRESENT  \
380163                        NaN               N

                                     the_geom COUNTY CODE  COUNTY NAME  \
380163  POINT (-75.620299999689 38.758500000044)           S       Sussex

        hour dayofweek month    year
380163  20.0   Tuesday   5.0  2018.0

[1 rows x 43 columns]
```

[33]:
```
crash_df = crash_df[(crash_df['CRASH CLASSIFICATION CODE'] != '33') &
           (crash_df['CRASH CLASSIFICATION CODE'] != '32') &
           (crash_df['CRASH CLASSIFICATION CODE'] != '31') &
           (crash_df['CRASH CLASSIFICATION CODE'] != '00')]
crash_df['CRASH CLASSIFICATION CODE'] = '0' + crash_df['CRASH CLASSIFICATION␣
 ↪CODE'].str[-1]

crash_df = crash_df[crash_df['MANNER OF IMPACT CODE'] != 30.0]

crash_df = crash_df[crash_df['PRIMARY CONTRIBUTING CIRCUMSTANCE CODE'] != 43.0]

crash_df.head()
```

[33]:
```
              CRASH DATETIME  DAY OF WEEK CODE DAY OF WEEK DESCRIPTION  \
0 2023-11-16 17:10:00+00:00                 5                 Thursday
1 2022-08-21 18:51:00+00:00                 1                   Sunday
2 2018-01-19 16:30:00+00:00                 6                   Friday
3 2018-06-05 16:30:00+00:00                 3                  Tuesday
4 2023-08-07 13:02:00+00:00                 2                   Monday

  CRASH CLASSIFICATION CODE CRASH CLASSIFICATION DESCRIPTION  \
0                        02              Property Damage Only
1                        03             Personal Injury Crash
2                        02              Property Damage Only
3                        01                    Non-Reportable
4                        01                    Non-Reportable
```

```
  COLLISION ON PRIVATE PROPERTY PEDESTRIAN INVOLVED  MANNER OF IMPACT CODE  \
0                             N                   N                    1.0
1                             N                   N                    3.0
2                             N                   N                   99.0
3                             Y                   N                    4.0
4                             Y                   N                    NaN

  MANNER OF IMPACT DESCRIPTION ALCOHOL INVOLVED DRUG INVOLVED  \
0                Front to rear                N             N
1                        Angle                N             N
2                      Unknown                N             N
3   Sideswipe, same direction                N             N
4                          NaN                N             N

   ROAD SURFACE CODE ROAD SURFACE DESCRIPTION  LIGHTING CONDITION CODE  \
0                1.0                      Dry                     1.0
1                1.0                      Dry                     1.0
2                1.0                      Dry                     1.0
3                1.0                      Dry                     1.0
4                NaN                      NaN                     NaN

  LIGHTING CONDITION DESCRIPTION  WEATHER 1 CODE WEATHER 1 DESCRIPTION  \
0                      Daylight             1.0                 Clear
1                      Daylight             2.0                Cloudy
2                      Daylight             1.0                 Clear
3                      Daylight             2.0                Cloudy
4                           NaN             NaN                   NaN

   WEATHER 2 CODE WEATHER 2 DESCRIPTION  …  LATITUDE LONGITUDE  \
0             NaN                   NaN  …  38.65787 -75.59441
1             NaN                   NaN  …  38.58520 -75.29074
2             NaN                   NaN  …  38.53826 -75.05662
3             NaN                   NaN  …  39.29256 -75.64020
4             NaN                   NaN  …  38.74038 -75.14624

  PRIMARY CONTRIBUTING CIRCUMSTANCE CODE  \
0                                   8.0
1                                  12.0
2                                  99.0
3                                  11.0
4                                  88.0

  PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION SCHOOL BUS INVOLVED CODE  \
0                          Following too close                      0.0
1      Driving in a careless or reckless manner                     0.0
2                                       Unknown                     0.0
3   Driver inattention, distraction, or fatigue                     0.0
```

```
4                                        Other                        0.0

   SCHOOL BUS INVOLVED DESCRIPTION  WORK ZONE  WORK ZONE LOCATION CODE  \
0                               No          N                      NaN
1                               No          N                      NaN
2                               No          N                      NaN
3                               No          N                      NaN
4                               No          N                      NaN

  WORK ZONE LOCATION DESCRIPTION  WORK ZONE TYPE CODE  \
0                            NaN                  NaN
1                            NaN                  NaN
2                            NaN                  NaN
3                            NaN                  NaN
4                            NaN                  NaN

  WORK ZONE TYPE DESCRIPTION WORKERS PRESENT  \
0                        NaN               N
1                        NaN               N
2                        NaN               N
3                        NaN               N
4                        NaN               N

                                  the_geom COUNTY CODE  COUNTY NAME  hour  \
0  POINT (-75.594409999839 38.657869999717)           S       Sussex  17.0
1  POINT (-75.290739999562 38.585200000283)           S       Sussex  18.0
2  POINT (-75.056620000321 38.538260000222)           S       Sussex  16.0
3  POINT (-75.640199999988 39.292560000244)           K         Kent  16.0
4  POINT (-75.146239999738 38.740379999925)           S       Sussex  13.0

  dayofweek month    year
0  Thursday  11.0  2023.0
1    Sunday   8.0  2022.0
2    Friday   1.0  2018.0
3   Tuesday   6.0  2018.0
4    Monday   8.0  2023.0

[5 rows x 43 columns]
```

[34]:
```
percent_missing = (crash_df.isna().sum() / len(crash_df))[(crash_df.isna().
↪sum() / len(crash_df)) > 0]
percent_missing
```

[34]:
```
CRASH CLASSIFICATION CODE                  0.000006
CRASH CLASSIFICATION DESCRIPTION           0.000045
MANNER OF IMPACT CODE                      0.037745
MANNER OF IMPACT DESCRIPTION               0.037749
```

```
ROAD SURFACE CODE                                0.040057
ROAD SURFACE DESCRIPTION                         0.040057
LIGHTING CONDITION CODE                          0.039198
LIGHTING CONDITION DESCRIPTION                   0.039198
WEATHER 1 CODE                                   0.040794
WEATHER 1 DESCRIPTION                            0.040794
WEATHER 2 CODE                                   0.962188
WEATHER 2 DESCRIPTION                            0.962188
PRIMARY CONTRIBUTING CIRCUMSTANCE CODE           0.032312
PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION    0.032312
SCHOOL BUS INVOLVED CODE                         0.000028
SCHOOL BUS INVOLVED DESCRIPTION                  0.000028
WORK ZONE LOCATION CODE                          0.993006
WORK ZONE LOCATION DESCRIPTION                   0.993006
WORK ZONE TYPE CODE                              0.993002
WORK ZONE TYPE DESCRIPTION                       0.993002
dtype: float64
```

[35]:
```
percent_missing = (crash_df.isna().sum() / len(crash_df))[(crash_df.isna().
↪sum() / len(crash_df)) > 0]
percent_missing
```

[35]:
```
CRASH CLASSIFICATION CODE                        0.000006
CRASH CLASSIFICATION DESCRIPTION                 0.000045
MANNER OF IMPACT CODE                            0.037745
MANNER OF IMPACT DESCRIPTION                     0.037749
ROAD SURFACE CODE                                0.040057
ROAD SURFACE DESCRIPTION                         0.040057
LIGHTING CONDITION CODE                          0.039198
LIGHTING CONDITION DESCRIPTION                   0.039198
WEATHER 1 CODE                                   0.040794
WEATHER 1 DESCRIPTION                            0.040794
WEATHER 2 CODE                                   0.962188
WEATHER 2 DESCRIPTION                            0.962188
PRIMARY CONTRIBUTING CIRCUMSTANCE CODE           0.032312
PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION    0.032312
SCHOOL BUS INVOLVED CODE                         0.000028
SCHOOL BUS INVOLVED DESCRIPTION                  0.000028
WORK ZONE LOCATION CODE                          0.993006
WORK ZONE LOCATION DESCRIPTION                   0.993006
WORK ZONE TYPE CODE                              0.993002
WORK ZONE TYPE DESCRIPTION                       0.993002
dtype: float64
```

[36]:
```
percent_missing[percent_missing.between(0.01, 0.05)]
```

[36]:
```
MANNER OF IMPACT CODE                              0.037745
MANNER OF IMPACT DESCRIPTION                       0.037749
ROAD SURFACE CODE                                  0.040057
ROAD SURFACE DESCRIPTION                           0.040057
LIGHTING CONDITION CODE                            0.039198
LIGHTING CONDITION DESCRIPTION                     0.039198
WEATHER 1 CODE                                     0.040794
WEATHER 1 DESCRIPTION                              0.040794
PRIMARY CONTRIBUTING CIRCUMSTANCE CODE             0.032312
PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION      0.032312
dtype: float64
```

[37]:
```
percent_missing[percent_missing > 0.05]
```

[37]:
```
WEATHER 2 CODE                    0.962188
WEATHER 2 DESCRIPTION             0.962188
WORK ZONE LOCATION CODE           0.993006
WORK ZONE LOCATION DESCRIPTION    0.993006
WORK ZONE TYPE CODE               0.993002
WORK ZONE TYPE DESCRIPTION        0.993002
dtype: float64
```

[38]:
```
row_drop = list(percent_missing[percent_missing < 0.01].index)
row_drop
```

[38]:
```
['CRASH CLASSIFICATION CODE',
 'CRASH CLASSIFICATION DESCRIPTION',
 'SCHOOL BUS INVOLVED CODE',
 'SCHOOL BUS INVOLVED DESCRIPTION']
```

[39]:
```
row_fill = list(percent_missing[percent_missing.between(0.01, 0.05)].index)
row_fill
```

[39]:
```
['MANNER OF IMPACT CODE',
 'MANNER OF IMPACT DESCRIPTION',
 'ROAD SURFACE CODE',
 'ROAD SURFACE DESCRIPTION',
 'LIGHTING CONDITION CODE',
 'LIGHTING CONDITION DESCRIPTION',
 'WEATHER 1 CODE',
 'WEATHER 1 DESCRIPTION',
 'PRIMARY CONTRIBUTING CIRCUMSTANCE CODE',
 'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION']
```

[40]:
```
col_drop = list(percent_missing[percent_missing > 0.05].index)
col_drop
```

[40]:
```
['WEATHER 2 CODE',
 'WEATHER 2 DESCRIPTION',
 'WORK ZONE LOCATION CODE',
 'WORK ZONE LOCATION DESCRIPTION',
 'WORK ZONE TYPE CODE',
 'WORK ZONE TYPE DESCRIPTION']
```

[41]:
```
crash_df = crash_df.drop(col_drop, axis = 1)
crash_df
```

[41]:

	CRASH DATETIME	DAY OF WEEK CODE	DAY OF WEEK DESCRIPTION	\
0	2023-11-16 17:10:00+00:00	5	Thursday	
1	2022-08-21 18:51:00+00:00	1	Sunday	
2	2018-01-19 16:30:00+00:00	6	Friday	
3	2018-06-05 16:30:00+00:00	3	Tuesday	
4	2023-08-07 13:02:00+00:00	2	Monday	
…	…	…	…	
536609	2011-09-15 18:44:00+00:00	5	Thursday	
536610	2014-10-23 12:26:00+00:00	5	Thursday	
536611	2009-08-23 21:00:00+00:00	1	Sunday	
536612	2009-05-11 14:43:00+00:00	2	Monday	
536613	2015-11-13 02:14:00+00:00	5	Thursday	

	CRASH CLASSIFICATION CODE	CRASH CLASSIFICATION DESCRIPTION	\
0	02	Property Damage Only	
1	03	Personal Injury Crash	
2	02	Property Damage Only	
3	01	Non-Reportable	
4	01	Non-Reportable	
…	…	…	
536609	02	Property Damage Only	
536610	02	Property Damage Only	
536611	02	Property Damage Only	
536612	02	Property Damage Only	
536613	01	Non-Reportable	

	COLLISION ON PRIVATE PROPERTY	PEDESTRIAN INVOLVED	\
0	N	N	
1	N	N	
2	N	N	
3	Y	N	
4	Y	N	
…	…	…	
536609	N	N	
536610	N	N	
536611	N	N	
536612	N	N	

```
536613                            N                   N

        MANNER OF IMPACT CODE MANNER OF IMPACT DESCRIPTION ALCOHOL INVOLVED  \
0                         1.0               Front to rear                N
1                         3.0                       Angle                N
2                        99.0                     Unknown                N
3                         4.0   Sideswipe, same direction                N
4                         NaN                         NaN                N
…                           …                           …              …
536609                    1.0               Front to rear                N
536610                    1.0               Front to rear                N
536611                    1.0               Front to rear                N
536612                    4.0   Sideswipe, same direction                N
536613                    NaN                         NaN                N

       DRUG INVOLVED  ROAD SURFACE CODE ROAD SURFACE DESCRIPTION  \
0                  N                1.0                      Dry
1                  N                1.0                      Dry
2                  N                1.0                      Dry
3                  N                1.0                      Dry
4                  N                NaN                      NaN
…                …                  …                        …
536609             N                1.0                      Dry
536610             N                1.0                      Dry
536611             N                1.0                      Dry
536612             N                1.0                      Dry
536613             N                NaN                      NaN

        LIGHTING CONDITION CODE LIGHTING CONDITION DESCRIPTION  \
0                           1.0                       Daylight
1                           1.0                       Daylight
2                           1.0                       Daylight
3                           1.0                       Daylight
4                           NaN                            NaN
…                             …                             …
536609                      1.0                       Daylight
536610                      1.0                       Daylight
536611                      1.0                       Daylight
536612                      1.0                       Daylight
536613                      NaN                            NaN

        WEATHER 1 CODE WEATHER 1 DESCRIPTION SEATBELT USED  \
0                  1.0                 Clear             Y
1                  2.0                Cloudy             Y
2                  1.0                 Clear             Y
3                  2.0                Cloudy             Y
4                  NaN                   NaN             Y
```

```
…                 …                    …            …
536609              2.0                 Cloudy            Y
536610              2.0                 Cloudy            Y
536611              1.0                  Clear            Y
536612              2.0                 Cloudy            Y
536613              NaN                    NaN            Y

       MOTORCYCLE INVOLVED MOTORCYCLE HELMET USED BICYCLED INVOLVED  \
0                        N                       N                N
1                        N                       N                N
2                        N                       N                N
3                        N                       N                N
4                        N                       N                N
…                      …                       …                …
536609                   N                       N                N
536610                   N                       N                N
536611                   N                       N                N
536612                   N                       N                N
536613                   N                       N                N

       BICYCLE HELMET USED   LATITUDE  LONGITUDE  \
0                        N  38.657870 -75.594410
1                        N  38.585200 -75.290740
2                        N  38.538260 -75.056620
3                        N  39.292560 -75.640200
4                        N  38.740380 -75.146240
…                      …        …         …
536609                   N  39.694590 -75.719570
536610                   N  39.604460 -75.748310
536611                   N  39.628448 -75.745197
536612                   N  39.757961 -75.562487
536613                   N  38.555790 -75.247350

        PRIMARY CONTRIBUTING CIRCUMSTANCE CODE  \
0                                        8.0
1                                       12.0
2                                       99.0
3                                       11.0
4                                       88.0
…                                       …
536609                                  11.0
536610                                  11.0
536611                                   NaN
536612                                   NaN
536613                                  17.0

       PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION  \
```

```
0                                Following too close
1           Driving in a careless or reckless manner
2                                            Unknown
3         Driver inattention, distraction, or fatigue
4                                              Other
…                                                …
536609    Driver inattention, distraction, or fatigue
536610    Driver inattention, distraction, or fatigue
536611                                           NaN
536612                                           NaN
536613                         Animal in Roadway - Deer

        SCHOOL BUS INVOLVED CODE SCHOOL BUS INVOLVED DESCRIPTION WORK ZONE  \
0                            0.0                                  No         N
1                            0.0                                  No         N
2                            0.0                                  No         N
3                            0.0                                  No         N
4                            0.0                                  No         N
…                            …                                    …       …
536609                       0.0                                  No         N
536610                       0.0                                  No         N
536611                       0.0                                  No         N
536612                       0.0                                  No         N
536613                       0.0                                  No         N

       WORKERS PRESENT                                  the_geom COUNTY CODE  \
0                    N  POINT (-75.594409999839 38.657869999717)           S
1                    N  POINT (-75.290739999562 38.585200000283)           S
2                    N  POINT (-75.056620000321 38.538260000222)           S
3                    N  POINT (-75.640199999988 39.292560000244)           K
4                    N  POINT (-75.146239999738 38.740379999925)           S
…                  …                                          …       …
536609               N  POINT (-75.719570000129 39.694590000064)           N
536610               N   POINT (-75.74831000044 39.604460000234)           N
536611               N  POINT (-75.745197099927 39.628447700087)           N
536612               N  POINT (-75.562486799556 39.757961200274)           N
536613               N  POINT (-75.247349999612 38.555789999772)           S

       COUNTY NAME  hour dayofweek  month    year
0           Sussex  17.0  Thursday   11.0  2023.0
1           Sussex  18.0    Sunday    8.0  2022.0
2           Sussex  16.0    Friday    1.0  2018.0
3             Kent  16.0   Tuesday    6.0  2018.0
4           Sussex  13.0    Monday    8.0  2023.0
…             …   …       …    …     …
536609  New Castle  18.0  Thursday    9.0  2011.0
536610  New Castle  12.0  Thursday   10.0  2014.0
```

```
536611  New Castle  21.0    Sunday    8.0  2009.0
536612  New Castle  14.0    Monday    5.0  2009.0
536613      Sussex   2.0    Friday   11.0  2015.0

[536604 rows x 37 columns]
```

[42]:
```
crash_df = crash_df.dropna(subset = row_drop)
crash_df
```

[42]:
```
                  CRASH DATETIME  DAY OF WEEK CODE DAY OF WEEK DESCRIPTION  \
0      2023-11-16 17:10:00+00:00                5                Thursday
1      2022-08-21 18:51:00+00:00                1                  Sunday
2      2018-01-19 16:30:00+00:00                6                  Friday
3      2018-06-05 16:30:00+00:00                3                 Tuesday
4      2023-08-07 13:02:00+00:00                2                  Monday
…                            …                …                       …
536609 2011-09-15 18:44:00+00:00                5                Thursday
536610 2014-10-23 12:26:00+00:00                5                Thursday
536611 2009-08-23 21:00:00+00:00                1                  Sunday
536612 2009-05-11 14:43:00+00:00                2                  Monday
536613 2015-11-13 02:14:00+00:00                5                Thursday

       CRASH CLASSIFICATION CODE CRASH CLASSIFICATION DESCRIPTION  \
0                             02             Property Damage Only
1                             03            Personal Injury Crash
2                             02             Property Damage Only
3                             01                   Non-Reportable
4                             01                   Non-Reportable
…                            …                                …
536609                        02             Property Damage Only
536610                        02             Property Damage Only
536611                        02             Property Damage Only
536612                        02             Property Damage Only
536613                        01                   Non-Reportable

       COLLISION ON PRIVATE PROPERTY PEDESTRIAN INVOLVED  \
0                                  N                   N
1                                  N                   N
2                                  N                   N
3                                  Y                   N
4                                  Y                   N
…                                …                   …
536609                             N                   N
536610                             N                   N
536611                             N                   N
536612                             N                   N
536613                             N                   N
```

```
        MANNER OF IMPACT CODE MANNER OF IMPACT DESCRIPTION ALCOHOL INVOLVED  \
0                         1.0                Front to rear                N
1                         3.0                        Angle                N
2                        99.0                      Unknown                N
3                         4.0    Sideswipe, same direction                N
4                         NaN                          NaN                N
…                           …                            …              …
536609                    1.0                Front to rear                N
536610                    1.0                Front to rear                N
536611                    1.0                Front to rear                N
536612                    4.0    Sideswipe, same direction                N
536613                    NaN                          NaN                N

       DRUG INVOLVED  ROAD SURFACE CODE ROAD SURFACE DESCRIPTION  \
0                  N                1.0                      Dry
1                  N                1.0                      Dry
2                  N                1.0                      Dry
3                  N                1.0                      Dry
4                  N                NaN                      NaN
…                …                  …                        …
536609             N                1.0                      Dry
536610             N                1.0                      Dry
536611             N                1.0                      Dry
536612             N                1.0                      Dry
536613             N                NaN                      NaN

        LIGHTING CONDITION CODE LIGHTING CONDITION DESCRIPTION  \
0                           1.0                      Daylight
1                           1.0                      Daylight
2                           1.0                      Daylight
3                           1.0                      Daylight
4                           NaN                           NaN
…                             …                            …
536609                      1.0                      Daylight
536610                      1.0                      Daylight
536611                      1.0                      Daylight
536612                      1.0                      Daylight
536613                      NaN                           NaN

        WEATHER 1 CODE WEATHER 1 DESCRIPTION SEATBELT USED  \
0                  1.0                 Clear             Y
1                  2.0                Cloudy             Y
2                  1.0                 Clear             Y
3                  2.0                Cloudy             Y
4                  NaN                   NaN             Y
…                  …                     …             …
```

```
536609             2.0                Cloudy             Y
536610             2.0                Cloudy             Y
536611             1.0                 Clear             Y
536612             2.0                Cloudy             Y
536613             NaN                   NaN             Y

       MOTORCYCLE INVOLVED MOTORCYCLE HELMET USED BICYCLED INVOLVED  \
0                        N                      N                 N
1                        N                      N                 N
2                        N                      N                 N
3                        N                      N                 N
4                        N                      N                 N
…                      …                      …               …
536609                   N                      N                 N
536610                   N                      N                 N
536611                   N                      N                 N
536612                   N                      N                 N
536613                   N                      N                 N

       BICYCLE HELMET USED   LATITUDE  LONGITUDE  \
0                        N  38.657870 -75.594410
1                        N  38.585200 -75.290740
2                        N  38.538260 -75.056620
3                        N  39.292560 -75.640200
4                        N  38.740380 -75.146240
…                      …        …         …
536609                   N  39.694590 -75.719570
536610                   N  39.604460 -75.748310
536611                   N  39.628448 -75.745197
536612                   N  39.757961 -75.562487
536613                   N  38.555790 -75.247350

        PRIMARY CONTRIBUTING CIRCUMSTANCE CODE  \
0                                        8.0
1                                       12.0
2                                       99.0
3                                       11.0
4                                       88.0
…                                      …
536609                                  11.0
536610                                  11.0
536611                                   NaN
536612                                   NaN
536613                                  17.0

       PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION  \
0                              Following too close
```

```
1           Driving in a careless or reckless manner
2                                            Unknown
3         Driver inattention, distraction, or fatigue
4                                              Other
…                                                …
536609    Driver inattention, distraction, or fatigue
536610    Driver inattention, distraction, or fatigue
536611                                            NaN
536612                                            NaN
536613                      Animal in Roadway - Deer

        SCHOOL BUS INVOLVED CODE SCHOOL BUS INVOLVED DESCRIPTION WORK ZONE  \
0                            0.0                                 No         N
1                            0.0                                 No         N
2                            0.0                                 No         N
3                            0.0                                 No         N
4                            0.0                                 No         N
…                            …                                   …        …
536609                       0.0                                 No         N
536610                       0.0                                 No         N
536611                       0.0                                 No         N
536612                       0.0                                 No         N
536613                       0.0                                 No         N

       WORKERS PRESENT                                  the_geom COUNTY CODE  \
0                    N  POINT (-75.594409999839 38.657869999717)           S
1                    N  POINT (-75.290739999562 38.585200000283)           S
2                    N  POINT (-75.056620000321 38.538260000222)           S
3                    N  POINT (-75.640199999988 39.292560000244)           K
4                    N  POINT (-75.146239999738 38.740379999925)           S
…                  …                                          …        …
536609               N  POINT (-75.719570000129 39.694590000064)           N
536610               N   POINT (-75.74831000044 39.604460000234)           N
536611               N  POINT (-75.745197099927 39.628447700087)           N
536612               N  POINT (-75.562486799556 39.757961200274)           N
536613               N  POINT (-75.247349999612 38.555789999772)           S

       COUNTY NAME  hour dayofweek  month    year
0           Sussex  17.0  Thursday   11.0  2023.0
1           Sussex  18.0    Sunday    8.0  2022.0
2           Sussex  16.0    Friday    1.0  2018.0
3             Kent  16.0   Tuesday    6.0  2018.0
4           Sussex  13.0    Monday    8.0  2023.0
…             …   …       …    …     …
536609  New Castle  18.0  Thursday    9.0  2011.0
536610  New Castle  12.0  Thursday   10.0  2014.0
536611  New Castle  21.0    Sunday    8.0  2009.0
```

```
536612  New Castle  14.0    Monday    5.0  2009.0
536613      Sussex   2.0    Friday   11.0  2015.0

[536565 rows x 37 columns]
```

[43]:
```
for col in row_fill:
    crash_df[col] = crash_df[col].fillna(crash_df[col].mode(dropna = True)[0])
crash_df
```

```
/var/folders/1c/47fsk5z56lq4c2qq6c1rfj7r0000gn/T/ipykernel_24863/701388996.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  crash_df[col] = crash_df[col].fillna(crash_df[col].mode(dropna = True)[0])
```

[43]:
```
                  CRASH DATETIME  DAY OF WEEK CODE DAY OF WEEK DESCRIPTION  \
0      2023-11-16 17:10:00+00:00                5                Thursday
1      2022-08-21 18:51:00+00:00                1                  Sunday
2      2018-01-19 16:30:00+00:00                6                  Friday
3      2018-06-05 16:30:00+00:00                3                 Tuesday
4      2023-08-07 13:02:00+00:00                2                  Monday
…                            …                …                       …
536609 2011-09-15 18:44:00+00:00                5                Thursday
536610 2014-10-23 12:26:00+00:00                5                Thursday
536611 2009-08-23 21:00:00+00:00                1                  Sunday
536612 2009-05-11 14:43:00+00:00                2                  Monday
536613 2015-11-13 02:14:00+00:00                5                Thursday

       CRASH CLASSIFICATION CODE CRASH CLASSIFICATION DESCRIPTION  \
0                             02             Property Damage Only
1                             03            Personal Injury Crash
2                             02             Property Damage Only
3                             01                   Non-Reportable
4                             01                   Non-Reportable
…                            …                                …
536609                        02             Property Damage Only
536610                        02             Property Damage Only
536611                        02             Property Damage Only
536612                        02             Property Damage Only
536613                        01                   Non-Reportable

       COLLISION ON PRIVATE PROPERTY PEDESTRIAN INVOLVED  \
0                                  N                   N
1                                  N                   N
2                                  N                   N
```

```
3                                  Y                    N
4                                  Y                    N
…                                  …                    …
536609                             N                    N
536610                             N                    N
536611                             N                    N
536612                             N                    N
536613                             N                    N

        MANNER OF IMPACT CODE MANNER OF IMPACT DESCRIPTION ALCOHOL INVOLVED  \
0                         1.0                Front to rear                N
1                         3.0                        Angle                N
2                        99.0                      Unknown                N
3                         4.0    Sideswipe, same direction                N
4                         1.0                Front to rear                N
…                         …                          …              …
536609                    1.0                Front to rear                N
536610                    1.0                Front to rear                N
536611                    1.0                Front to rear                N
536612                    4.0    Sideswipe, same direction                N
536613                    1.0                Front to rear                N

       DRUG INVOLVED  ROAD SURFACE CODE ROAD SURFACE DESCRIPTION  \
0                  N                1.0                      Dry
1                  N                1.0                      Dry
2                  N                1.0                      Dry
3                  N                1.0                      Dry
4                  N                1.0                      Dry
…              …                …                        …
536609             N                1.0                      Dry
536610             N                1.0                      Dry
536611             N                1.0                      Dry
536612             N                1.0                      Dry
536613             N                1.0                      Dry

        LIGHTING CONDITION CODE LIGHTING CONDITION DESCRIPTION  \
0                           1.0                       Daylight
1                           1.0                       Daylight
2                           1.0                       Daylight
3                           1.0                       Daylight
4                           1.0                       Daylight
…                           …                          …
536609                      1.0                       Daylight
536610                      1.0                       Daylight
536611                      1.0                       Daylight
536612                      1.0                       Daylight
536613                      1.0                       Daylight
```

```
        WEATHER 1 CODE WEATHER 1 DESCRIPTION SEATBELT USED  \
0                  1.0                 Clear             Y
1                  2.0                Cloudy             Y
2                  1.0                 Clear             Y
3                  2.0                Cloudy             Y
4                  1.0                 Clear             Y
…                    …                    …             …
536609             2.0                Cloudy             Y
536610             2.0                Cloudy             Y
536611             1.0                 Clear             Y
536612             2.0                Cloudy             Y
536613             1.0                 Clear             Y

       MOTORCYCLE INVOLVED MOTORCYCLE HELMET USED BICYCLED INVOLVED  \
0                        N                       N                 N
1                        N                       N                 N
2                        N                       N                 N
3                        N                       N                 N
4                        N                       N                 N
…                      …                       …               …
536609                   N                       N                 N
536610                   N                       N                 N
536611                   N                       N                 N
536612                   N                       N                 N
536613                   N                       N                 N

       BICYCLE HELMET USED   LATITUDE  LONGITUDE  \
0                        N  38.657870 -75.594410
1                        N  38.585200 -75.290740
2                        N  38.538260 -75.056620
3                        N  39.292560 -75.640200
4                        N  38.740380 -75.146240
…                      …        …         …
536609                   N  39.694590 -75.719570
536610                   N  39.604460 -75.748310
536611                   N  39.628448 -75.745197
536612                   N  39.757961 -75.562487
536613                   N  38.555790 -75.247350

        PRIMARY CONTRIBUTING CIRCUMSTANCE CODE  \
0                                         8.0
1                                        12.0
2                                        99.0
3                                        11.0
4                                        88.0
…                                          …
```

```
536609                                  11.0
536610                                  11.0
536611                                  11.0
536612                                  11.0
536613                                  17.0

       PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION  \
0                              Following too close
1          Driving in a careless or reckless manner
2                                        Unknown
3        Driver inattention, distraction, or fatigue
4                                          Other
…                                              …
536609   Driver inattention, distraction, or fatigue
536610   Driver inattention, distraction, or fatigue
536611   Driver inattention, distraction, or fatigue
536612   Driver inattention, distraction, or fatigue
536613                     Animal in Roadway - Deer

        SCHOOL BUS INVOLVED CODE SCHOOL BUS INVOLVED DESCRIPTION WORK ZONE  \
0                            0.0                              No         N
1                            0.0                              No         N
2                            0.0                              No         N
3                            0.0                              No         N
4                            0.0                              No         N
…                              …                               …       …
536609                       0.0                              No         N
536610                       0.0                              No         N
536611                       0.0                              No         N
536612                       0.0                              No         N
536613                       0.0                              No         N

       WORKERS PRESENT                                the_geom COUNTY CODE  \
0                    N  POINT (-75.594409999839 38.657869999717)          S
1                    N  POINT (-75.290739999562 38.585200000283)          S
2                    N  POINT (-75.056620000321 38.538260000222)          S
3                    N  POINT (-75.640199999988 39.292560000244)          K
4                    N  POINT (-75.146239999738 38.740379999925)          S
…                  …                                         …       …
536609               N  POINT (-75.719570000129 39.694590000064)          N
536610               N   POINT (-75.74831000044 39.604460000234)          N
536611               N  POINT (-75.745197099927 39.628447700087)          N
536612               N  POINT (-75.562486799556 39.757961200274)          N
536613               N  POINT (-75.247349999612 38.555789999772)          S

       COUNTY NAME  hour dayofweek  month    year
0           Sussex  17.0  Thursday   11.0  2023.0
```

```
1           Sussex  18.0    Sunday    8.0  2022.0
2           Sussex  16.0    Friday    1.0  2018.0
3             Kent  16.0   Tuesday    6.0  2018.0
4           Sussex  13.0    Monday    8.0  2023.0
…              …    …        …       …     …
536609  New Castle  18.0  Thursday    9.0  2011.0
536610  New Castle  12.0  Thursday   10.0  2014.0
536611  New Castle  21.0    Sunday    8.0  2009.0
536612  New Castle  14.0    Monday    5.0  2009.0
536613      Sussex   2.0    Friday   11.0  2015.0

[536565 rows x 37 columns]
```

[44]:
```
crash_df.isna().sum()
```

[44]:
```
CRASH DATETIME                                  0
DAY OF WEEK CODE                                0
DAY OF WEEK DESCRIPTION                         0
CRASH CLASSIFICATION CODE                       0
CRASH CLASSIFICATION DESCRIPTION                0
COLLISION ON PRIVATE PROPERTY                   0
PEDESTRIAN INVOLVED                             0
MANNER OF IMPACT CODE                           0
MANNER OF IMPACT DESCRIPTION                    0
ALCOHOL INVOLVED                                0
DRUG INVOLVED                                   0
ROAD SURFACE CODE                               0
ROAD SURFACE DESCRIPTION                        0
LIGHTING CONDITION CODE                         0
LIGHTING CONDITION DESCRIPTION                  0
WEATHER 1 CODE                                  0
WEATHER 1 DESCRIPTION                           0
SEATBELT USED                                   0
MOTORCYCLE INVOLVED                             0
MOTORCYCLE HELMET USED                          0
BICYCLED INVOLVED                               0
BICYCLE HELMET USED                             0
LATITUDE                                        0
LONGITUDE                                       0
PRIMARY CONTRIBUTING CIRCUMSTANCE CODE          0
PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION   0
SCHOOL BUS INVOLVED CODE                        0
SCHOOL BUS INVOLVED DESCRIPTION                 0
WORK ZONE                                       0
WORKERS PRESENT                                 0
the_geom                                        0
COUNTY CODE                                     0
```

```
COUNTY NAME                                    0
hour                                           0
dayofweek                                      0
month                                          0
year                                           0
dtype: int64
```

[45]:
```
crash_df.to_csv('clean_crash_data.csv', index = False)
```

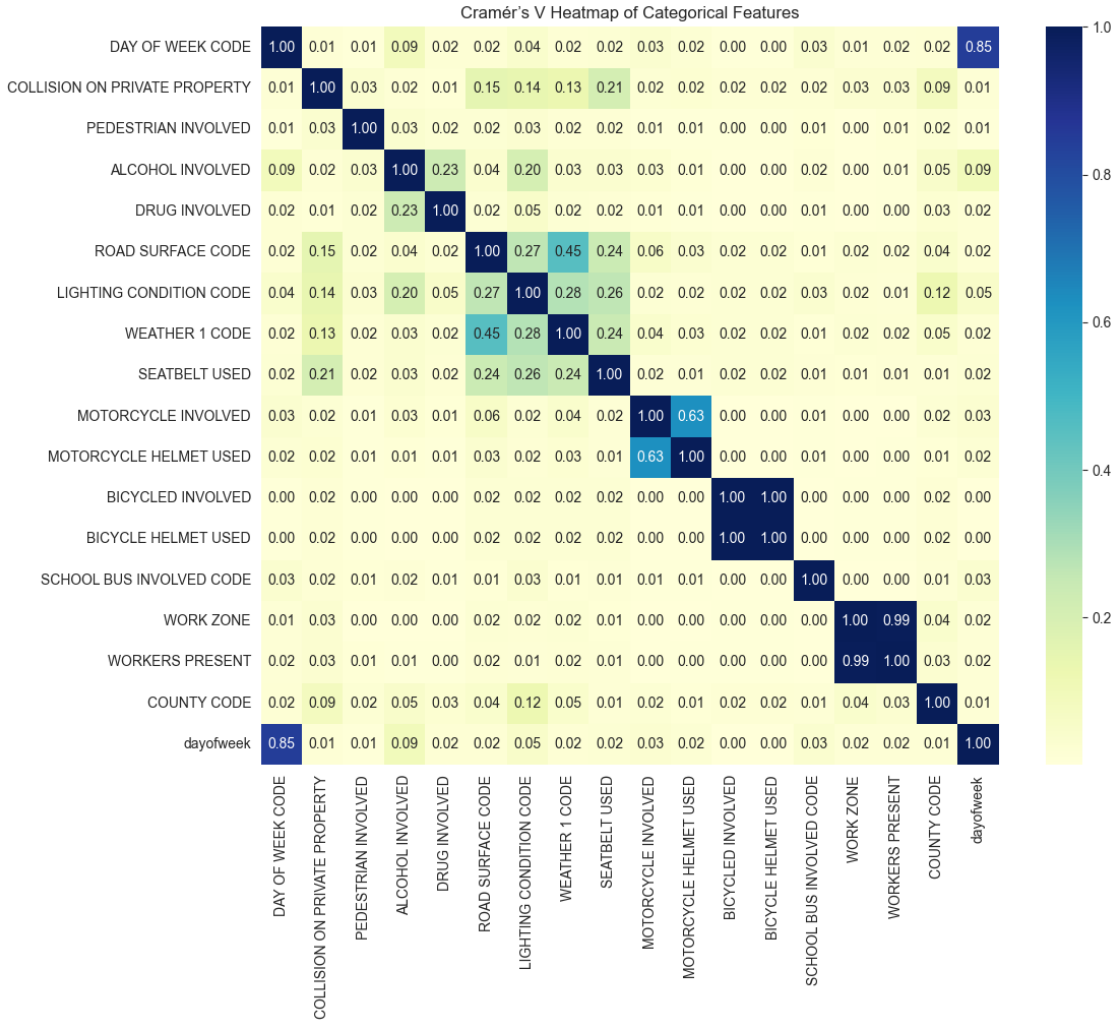
2 Feature Engineering

[47]:
```
import pandas as pd
import numpy as np
from sklearn.feature_selection import mutual_info_classif
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import chi2_contingency
```

[48]:
```
crash_df = pd.read_csv('clean_crash_data.csv')
crash_df_fe= crash_df.drop(columns=[col for col in crash_df.columns if␣
 ↪'DESCRIPTION' in col.upper()])
```

[49]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency

def cramers_v(x, y):
    confusion_matrix = pd.crosstab(x, y)
    if confusion_matrix.size == 0 or confusion_matrix.shape[0] == 1 or␣
 ↪confusion_matrix.shape[1] == 1:
        return np.nan
    chi2 = chi2_contingency(confusion_matrix)[0]
    n = confusion_matrix.sum().sum()
    return np.sqrt(chi2 / (n * (min(confusion_matrix.shape) - 1)))

# Use only original categorical features (before one-hot)
categorical_vars = [
    'DAY OF WEEK CODE', 'COLLISION ON PRIVATE PROPERTY', 'PEDESTRIAN INVOLVED',
    'ALCOHOL INVOLVED', 'DRUG INVOLVED', 'ROAD SURFACE CODE', 'LIGHTING␣
 ↪CONDITION CODE',
    'WEATHER 1 CODE', 'SEATBELT USED', 'MOTORCYCLE INVOLVED', 'MOTORCYCLE␣
 ↪HELMET USED',
    'BICYCLED INVOLVED', 'BICYCLE HELMET USED', 'SCHOOL BUS INVOLVED CODE',
    'WORK ZONE', 'WORKERS PRESENT', 'COUNTY CODE', 'dayofweek'
]
```

```
cramer_matrix = pd.DataFrame(index=categorical_vars, columns=categorical_vars)
for var1 in categorical_vars:
    for var2 in categorical_vars:
        cramer_matrix.loc[var1, var2] = cramers_v(crash_df_fe[var1],␣
 ↪crash_df_fe[var2])

plt.figure(figsize=(12, 10))
sns.heatmap(cramer_matrix.astype(float), annot=True, fmt=".2f", cmap="YlGnBu",␣
 ↪square=True)
plt.title("Cramér's V Heatmap of Categorical Features")
plt.tight_layout()
plt.show()
```



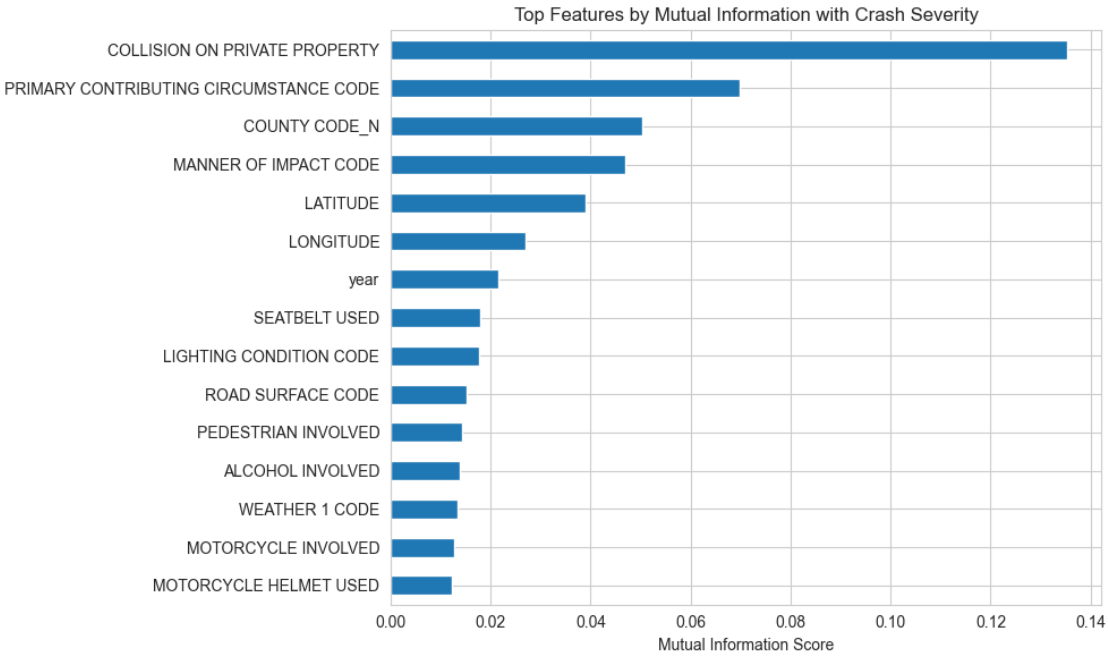
[50]:
```
binary_cols = [
    'COLLISION ON PRIVATE PROPERTY', 'PEDESTRIAN INVOLVED',
```

```
    'ALCOHOL INVOLVED', 'DRUG INVOLVED', 'SEATBELT USED',
    'MOTORCYCLE INVOLVED', 'MOTORCYCLE HELMET USED',
    'BICYCLED INVOLVED', 'BICYCLE HELMET USED',
    'WORK ZONE'
]
binary_map = {'Y': 1, 'YES': 1, '1': 1, 'N': 0, 'NO': 0, '0': 0}
for col in binary_cols:
    if col in crash_df_fe.columns:
        crash_df_fe[col] = crash_df_fe[col].astype(str).str.strip().str.upper().
 ↪map(binary_map).astype(int)
```

[51]:
```
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Define features for modeling
X_raw = crash_df_fe.drop(columns=['CRASH CLASSIFICATION CODE', 'CRASH␣
 ↪DATETIME', 'the_geom', 'COUNTY NAME', 'WORKERS PRESENT'], errors='ignore')
y = crash_df_fe['CRASH CLASSIFICATION CODE']

# Identify column types
categorical_cols = X_raw.select_dtypes(include=['object', 'category']).columns.
 ↪tolist()
numerical_cols = X_raw.select_dtypes(include=['int64', 'float64', 'bool']).
 ↪columns.tolist()

# Optional: remove ID-like or irrelevant columns from lists above if any

# Define transformers
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),
        ('cat', OneHotEncoder(drop='first', handle_unknown='ignore'),␣
 ↪categorical_cols)
    ]
)

# Fit and transform X for modeling (not affecting crash_df_fe used earlier)
X_processed = preprocessor.fit_transform(X_raw)
```

[52]:
```
from sklearn.feature_selection import mutual_info_classif
import pandas as pd

# Target
y = crash_df_fe['CRASH CLASSIFICATION CODE']

# Get feature names from ColumnTransformer
```

```
cat_features = preprocessor.named_transformers_['cat'].
  ↪get_feature_names_out(categorical_cols)
all_features = numerical_cols + cat_features.tolist()

# Now create a DataFrame for MI calculation and plotting
X_df = pd.DataFrame(X_processed, columns=all_features)

# Compute MI
mi = mutual_info_classif(X_df, y, discrete_features='auto', random_state=42)
mi_series = pd.Series(mi, index=X_df.columns).sort_values(ascending=False)

# Plot
import matplotlib.pyplot as plt

mi_series.head(15).plot(kind='barh', figsize=(10, 6))
plt.title('Top Features by Mutual Information with Crash Severity')
plt.xlabel('Mutual Information Score')
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```

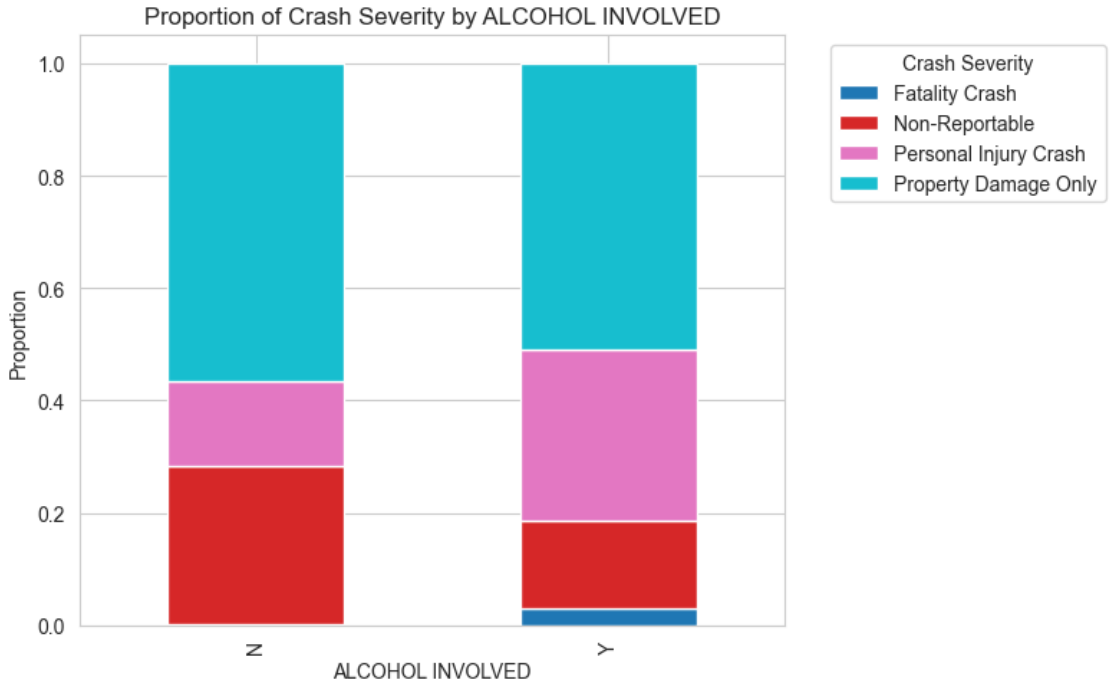


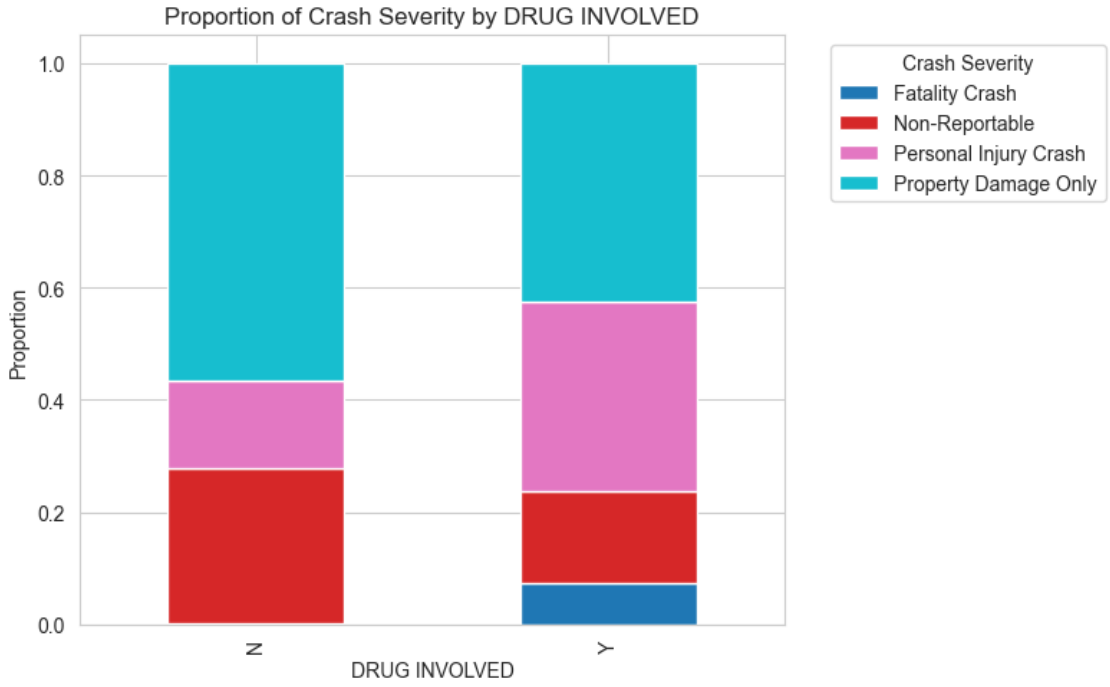
[53]:
```
print(crash_df_fe['ALCOHOL INVOLVED'].value_counts())
print(crash_df_fe['DRUG INVOLVED'].value_counts())
print(mi_series['ALCOHOL INVOLVED'])
print(mi_series['DRUG INVOLVED'])
```

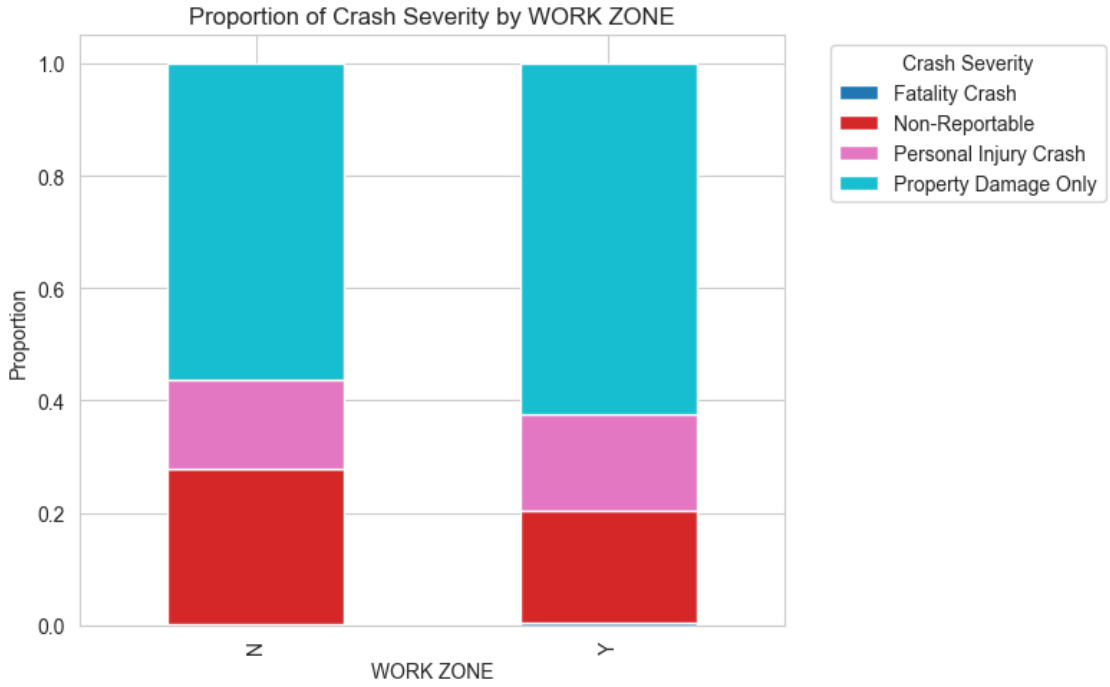
```
ALCOHOL INVOLVED
0    515641
1     20924
Name: count, dtype: int64
DRUG INVOLVED
0    530024
1      6541
Name: count, dtype: int64
0.013899895210510493
0.007731252136783828
```

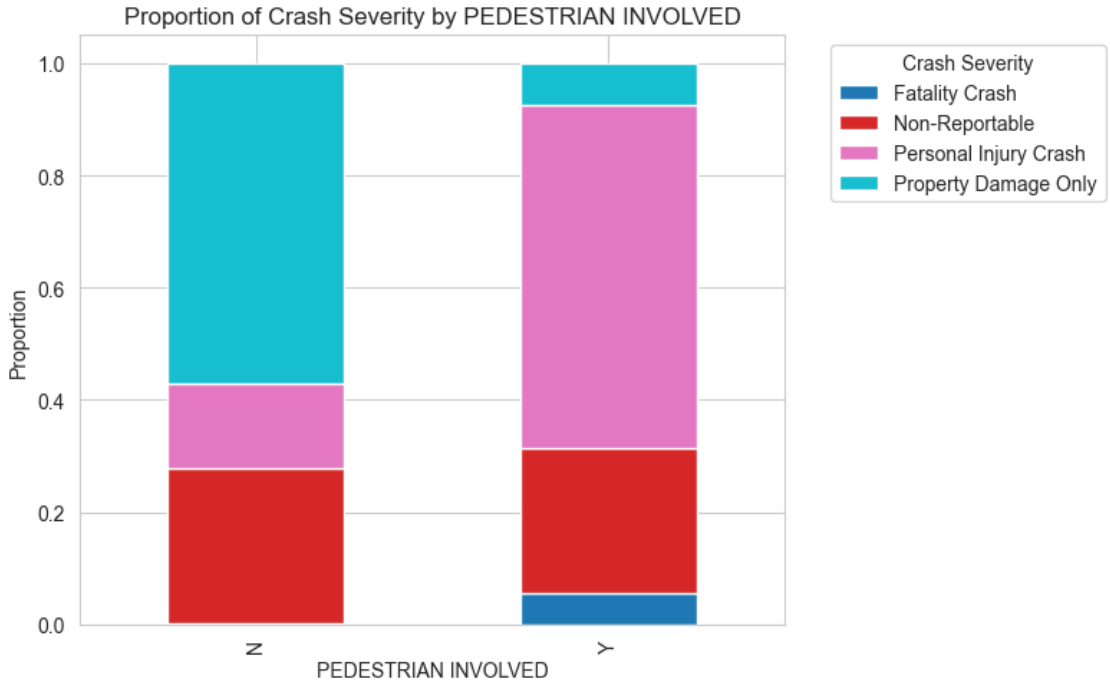
[54]:
```
features_to_plot = [
    'ALCOHOL INVOLVED',
    'DRUG INVOLVED',
    'WORK ZONE',
    'PEDESTRIAN INVOLVED',
    'MOTORCYCLE INVOLVED',
    'BICYCLED INVOLVED',
    'SCHOOL BUS INVOLVED DESCRIPTION'
]
```

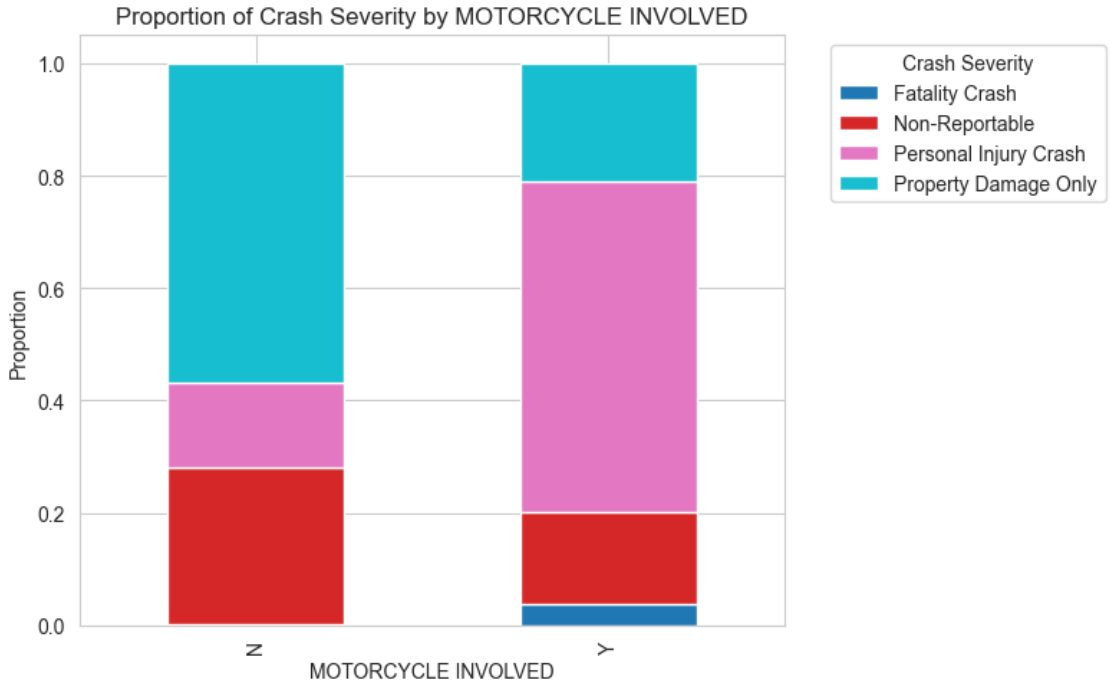
[55]:
```
for feature in features_to_plot:
    if feature in crash_df.columns:
        cross = pd.crosstab(crash_df[feature], crash_df['CRASH CLASSIFICATION␣
 ↪DESCRIPTION'], normalize='index')
        cross.plot(kind='bar', stacked=True, colormap='tab10', figsize=(8, 5))
        plt.title(f"Proportion of Crash Severity by {feature}")
        plt.ylabel("Proportion")
        plt.xlabel(feature)
        plt.legend(title="Crash Severity", bbox_to_anchor=(1.05, 1))
        plt.tight_layout()
        plt.show()
```

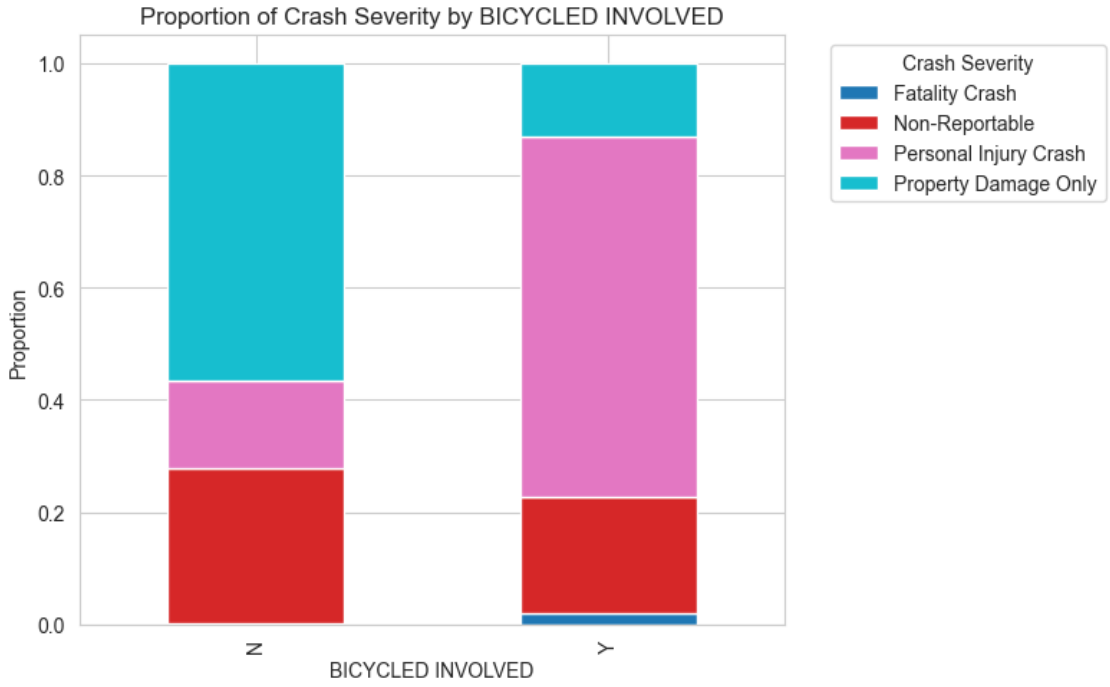

Proportion of Crash Severity by ALCOHOL INVOLVED
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
N
Y
ALCOHOL INVOLVED
Crash Severity
Fatality Crash
Non-Reportable
Personal Injury Crash
Property Damage Only

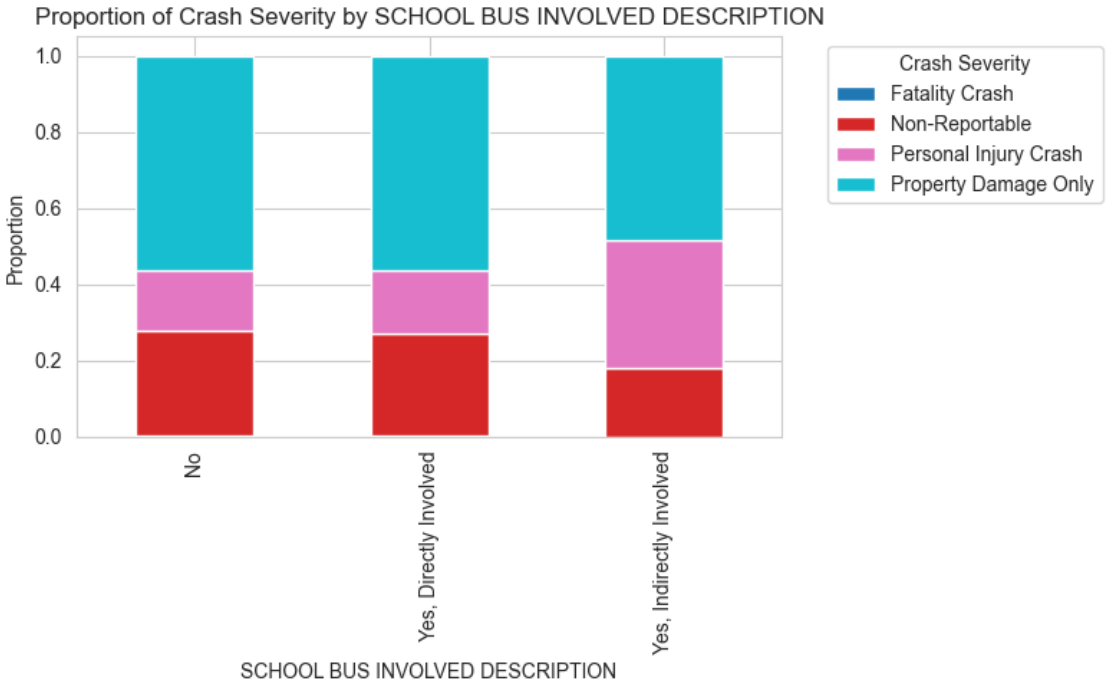

Proportion of Crash Severity by DRUG INVOLVED
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
N
Y
DRUG INVOLVED
Crash Severity
Fatality Crash
Non-Reportable
Personal Injury Crash
Property Damage Only


Proportion of Crash Severity by WORK ZONE
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
N
Y
WORK ZONE
Crash Severity
Fatality Crash
Non-Reportable
Personal Injury Crash
Property Damage Only


Proportion of Crash Severity by PEDESTRIAN INVOLVED
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
N
Y
PEDESTRIAN INVOLVED
Crash Severity
Fatality Crash
Non-Reportable
Personal Injury Crash
Property Damage Only


Proportion of Crash Severity by MOTORCYCLE INVOLVED
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
N
Y
MOTORCYCLE INVOLVED
Crash Severity
Fatality Crash
Non-Reportable
Personal Injury Crash
Property Damage Only


Proportion of Crash Severity by BICYCLED INVOLVED
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
N
Y
BICYCLED INVOLVED
Crash Severity
Fatality Crash
Non-Reportable
Personal Injury Crash
Property Damage Only



3 PCA Analysis

[57]:
```
# importing libraries
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
```

[58]:
```
clean_df = pd.read_csv('clean_crash_data.csv')
```

[59]:
```
# dropping the non numeric columns
non_numeric = ['CRASH CLASSIFICATION DESCRIPTION', 'CRASH DATETIME', 'COUNTY␣
 ↪NAME']
X = clean_df.drop(columns=non_numeric, errors='ignore')
```

[60]:
```
# identifying high cardinality columns
high_card = X.nunique().sort_values(ascending=False)
print(high_card.head(15))
```

```
the_geom                                467943
LATITUDE                                114191
LONGITUDE                                78973
hour                                        24
PRIMARY CONTRIBUTING CIRCUMSTANCE CODE      23
```

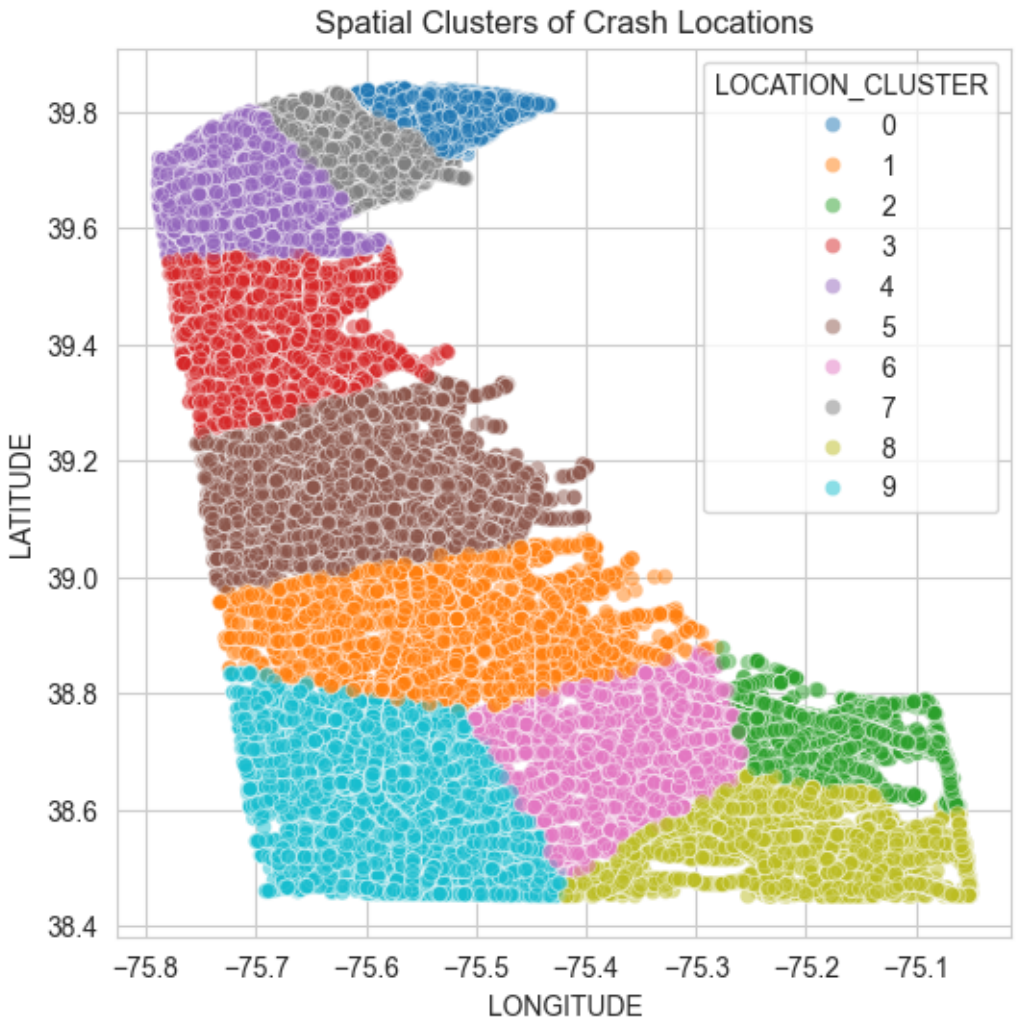
```
PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION        23
year                                                 16
month                                                12
ROAD SURFACE CODE                                    11
ROAD SURFACE DESCRIPTION                             11
WEATHER 1 CODE                                       11
WEATHER 1 DESCRIPTION                                11
MANNER OF IMPACT DESCRIPTION                         10
MANNER OF IMPACT CODE                                10
LIGHTING CONDITION CODE                               8
dtype: int64
```

[61]:
```
# drop the_geom
X = X.drop(columns=['the_geom'])
```

[62]:
```
# k means clustering to latitude/longitude
from sklearn.cluster import KMeans

# rows with valid coordinates
coord_df = clean_df[['LATITUDE', 'LONGITUDE']].dropna()

k = 10

# fit kmeans
kmeans = KMeans(n_clusters=k, random_state=42)
clusters = kmeans.fit_predict(coord_df)


clean_df['LOCATION_CLUSTER'] = -1
clean_df.loc[coord_df.index, 'LOCATION_CLUSTER'] = clusters
```

[63]:
```
# Drop the raw coordinates
X = clean_df.drop(columns=['the_geom', 'LATITUDE', 'LONGITUDE'],␣
 ↪errors='ignore')
```

[64]:
```
plt.figure(figsize=(6, 6))
sns.set_style("whitegrid")

sns.scatterplot(x='LONGITUDE', y='LATITUDE', hue='LOCATION_CLUSTER',␣
 ↪data=clean_df, palette='tab10', alpha=0.5)
plt.title("Spatial Clusters of Crash Locations")
plt.show()
```



[65]:
```
non_numeric_cols = X.select_dtypes(include=['object', 'datetime64']).columns
X = X.drop(columns=non_numeric_cols)
```

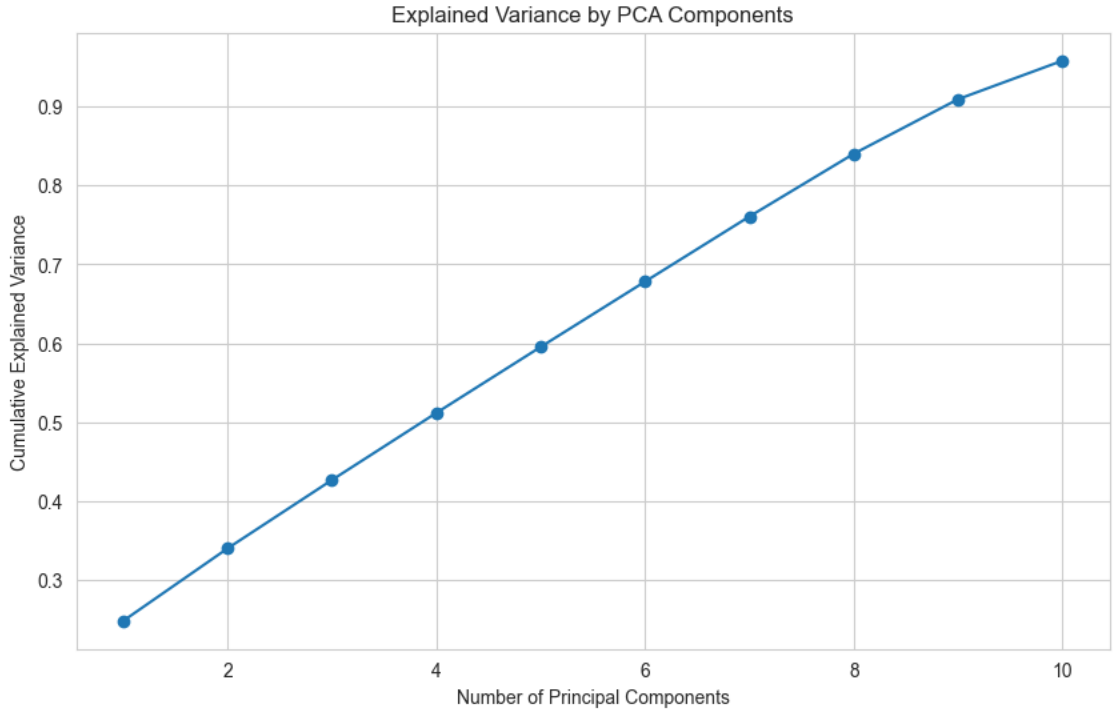
[66]:
```
from sklearn.preprocessing import StandardScaler

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

[67]:
```
# identify object columns that would fail standardization
print(X.select_dtypes(include=['object', 'datetime64']).columns)
```

```
Index([], dtype='object')
```

[68]:
```
from sklearn.decomposition import PCA

# enough components to explain 95% of the variance
pca = PCA(n_components=0.95)
X_pca = pca.fit_transform(X_scaled)
```

[69]:
```
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(pca.explained_variance_ratio_) + 1),
         pca.explained_variance_ratio_.cumsum(), marker='o')
plt.xlabel("Number of Principal Components")
plt.ylabel("Cumulative Explained Variance")
plt.title("Explained Variance by PCA Components")
plt.grid(True)
plt.show()
```

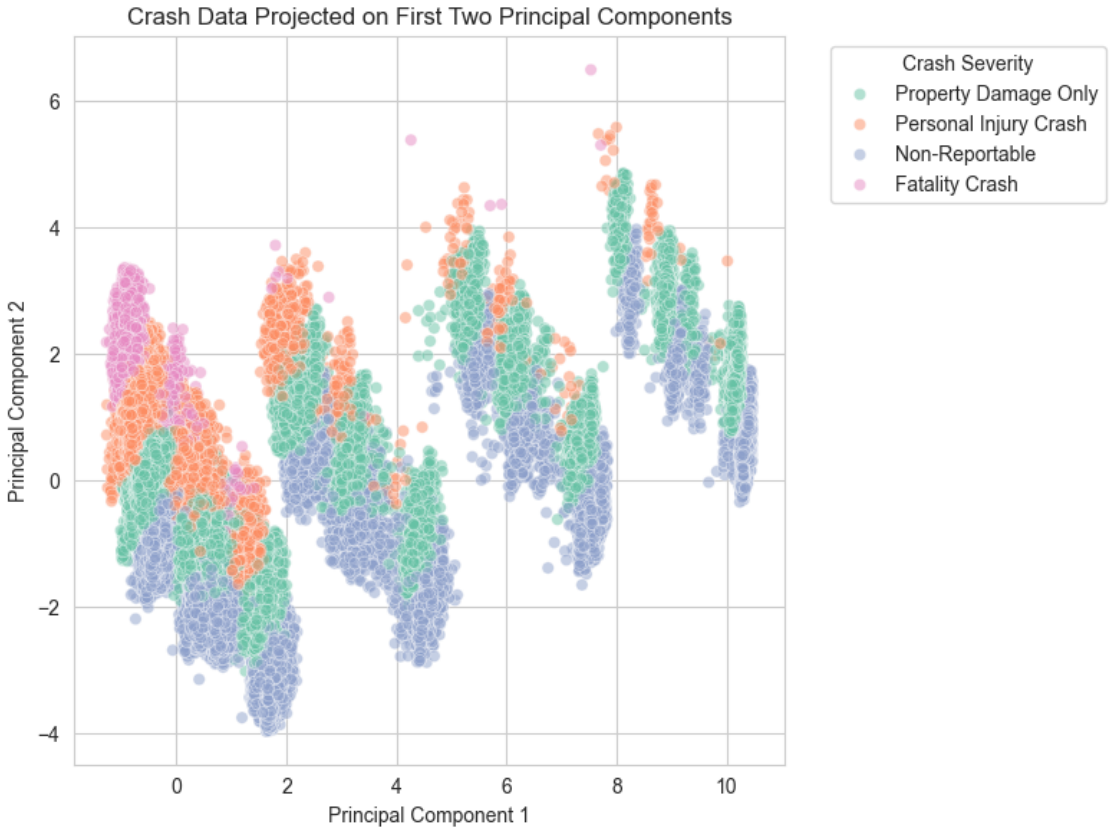


[70]:
```
# top contributing features in PC1

pca_components = pd.DataFrame(
    pca.components_,
    columns=X.columns,
    index=[f'PC{i+1}' for i in range(pca.n_components_)]
)
```

```
top_features_pc1 = pca_components.loc['PC1'].abs().sort_values(ascending=False).
  ↪head(10)
print("Top contributing features to PC1:")
print(top_features_pc1)
```

```
Top contributing features to PC1:
WEATHER 1 CODE                              0.512826
ROAD SURFACE CODE                           0.506244
LIGHTING CONDITION CODE                     0.484568
MANNER OF IMPACT CODE                       0.348253
PRIMARY CONTRIBUTING CIRCUMSTANCE CODE      0.323728
CRASH CLASSIFICATION CODE                   0.128624
hour                                        0.051776
SCHOOL BUS INVOLVED CODE                    0.014795
month                                       0.014226
LOCATION_CLUSTER                            0.012438
Name: PC1, dtype: float64
```

[71]:
```
y = clean_df['CRASH CLASSIFICATION DESCRIPTION']

# visualizing the first 2 components
plt.figure(figsize=(8,6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=y, alpha=0.5, palette='Set2')
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.title("Crash Data Projected on First Two Principal Components")
plt.legend(title="Crash Severity", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



[72]:
```python
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

infra_cols = ['WEATHER 1 CODE', 'ROAD SURFACE CODE', 'LIGHTING CONDITION CODE']

# Replace missing or "unknown" with -1
crash_df_fe[infra_cols] = crash_df_fe[infra_cols].replace({88: -1, 99: -1})

# (optional) If you had NaNs already from earlier:
crash_df_fe[infra_cols] = crash_df_fe[infra_cols].fillna(-1)

# Standardize
scaler = StandardScaler()
infra_scaled = scaler.fit_transform(crash_df_fe[infra_cols])

# Apply PCA
pca = PCA(n_components=1)
crash_df_fe['INFRASTRUCTURE_CONDITION'] = pca.fit_transform(infra_scaled)
```

[73]:
```python
print("All columns:", crash_df_fe.columns.tolist())
```

```
All columns: ['CRASH DATETIME', 'DAY OF WEEK CODE', 'CRASH CLASSIFICATION CODE',
'COLLISION ON PRIVATE PROPERTY', 'PEDESTRIAN INVOLVED', 'MANNER OF IMPACT CODE',
'ALCOHOL INVOLVED', 'DRUG INVOLVED', 'ROAD SURFACE CODE', 'LIGHTING CONDITION
CODE', 'WEATHER 1 CODE', 'SEATBELT USED', 'MOTORCYCLE INVOLVED', 'MOTORCYCLE
HELMET USED', 'BICYCLED INVOLVED', 'BICYCLE HELMET USED', 'LATITUDE',
'LONGITUDE', 'PRIMARY CONTRIBUTING CIRCUMSTANCE CODE', 'SCHOOL BUS INVOLVED
CODE', 'WORK ZONE', 'WORKERS PRESENT', 'the_geom', 'COUNTY CODE', 'COUNTY NAME',
'hour', 'dayofweek', 'month', 'year', 'INFRASTRUCTURE_CONDITION']
```

[74]:
```
target = 'CRASH CLASSIFICATION CODE'

correlations = crash_df_fe.select_dtypes(include=['number']).corr()[target].
  ↪sort_values(ascending=False)
print(correlations)
```

```
CRASH CLASSIFICATION CODE                1.000000
SEATBELT USED                            0.127548
MOTORCYCLE HELMET USED                   0.113169
MOTORCYCLE INVOLVED                      0.104714
LIGHTING CONDITION CODE                  0.102563
PEDESTRIAN INVOLVED                      0.101429
ALCOHOL INVOLVED                         0.097898
INFRASTRUCTURE_CONDITION                 0.084128
DRUG INVOLVED                            0.073590
BICYCLED INVOLVED                        0.067542
BICYCLE HELMET USED                      0.067542
WEATHER 1 CODE                           0.066160
LATITUDE                                 0.051009
ROAD SURFACE CODE                        0.050448
month                                    0.014283
WORK ZONE                                0.012329
year                                     0.012200
SCHOOL BUS INVOLVED CODE                 0.003466
DAY OF WEEK CODE                        -0.001180
hour                                    -0.031388
LONGITUDE                               -0.040982
MANNER OF IMPACT CODE                   -0.138348
PRIMARY CONTRIBUTING CIRCUMSTANCE CODE  -0.181245
COLLISION ON PRIVATE PROPERTY           -0.434470
Name: CRASH CLASSIFICATION CODE, dtype: float64
```

[75]:
```
crash_df['CRASH CLASSIFICATION CODE'].value_counts(normalize=True)
```

[75]:
```
CRASH CLASSIFICATION CODE
2    0.563919
1    0.275696
3    0.157260
4    0.003125
```

```
Name: proportion, dtype: float64
```

4 Clustering

[77]:
```
df = crash_df.copy()
```

[78]:
```
categorical_cols = [col for col in df.columns if col[-4:] != 'CODE' and col not␣
 ↪in ['CRASH DATETIME', 'LATITUDE', 'LONGITUDE', 'the_geom', 'CRASH␣
 ↪CLASSIFICATION DESCRIPTION', 'hour', 'dayofweek', 'month', 'year',␣
 ↪'LOCATION_CLUSTER']]
categorical_cols
```

[78]:
```
['DAY OF WEEK DESCRIPTION',
 'COLLISION ON PRIVATE PROPERTY',
 'PEDESTRIAN INVOLVED',
 'MANNER OF IMPACT DESCRIPTION',
 'ALCOHOL INVOLVED',
 'DRUG INVOLVED',
 'ROAD SURFACE DESCRIPTION',
 'LIGHTING CONDITION DESCRIPTION',
 'WEATHER 1 DESCRIPTION',
 'SEATBELT USED',
 'MOTORCYCLE INVOLVED',
 'MOTORCYCLE HELMET USED',
 'BICYCLED INVOLVED',
 'BICYCLE HELMET USED',
 'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION',
 'SCHOOL BUS INVOLVED DESCRIPTION',
 'WORK ZONE',
 'WORKERS PRESENT',
 'COUNTY NAME']
```

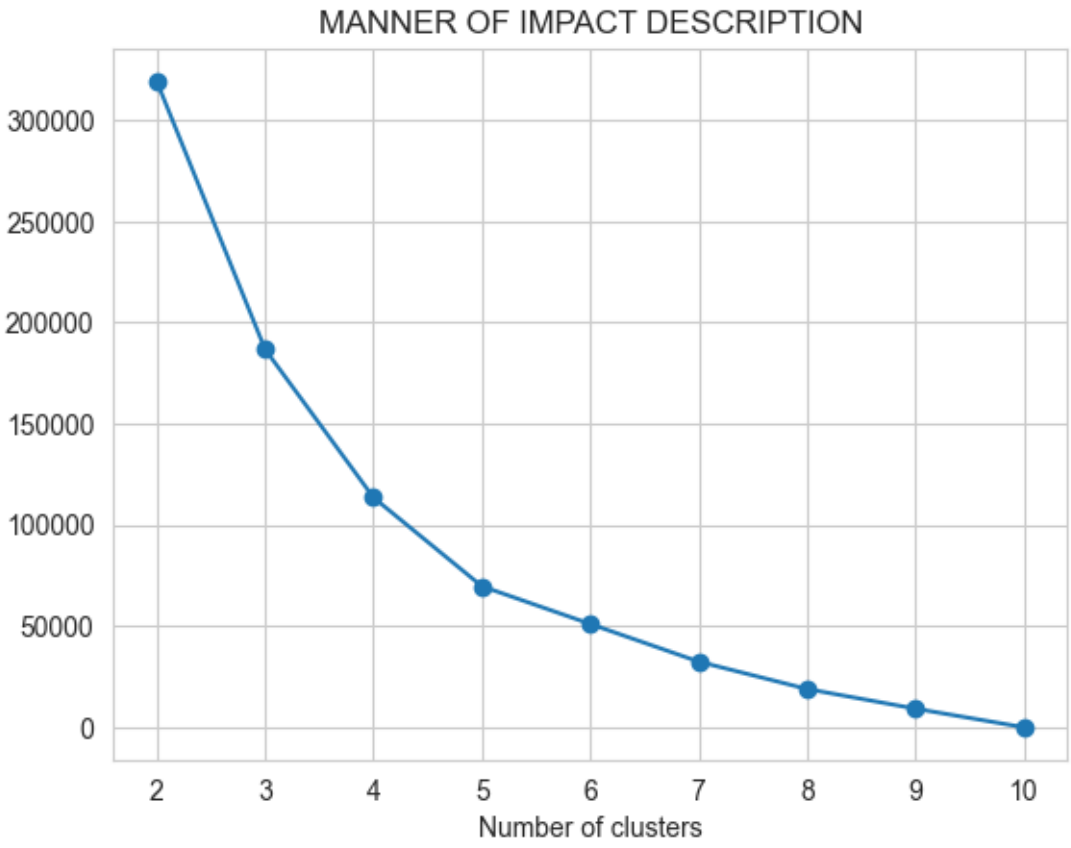
[79]:
```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.cluster import KMeans
from sklearn.compose import ColumnTransformer

high_card_cols = [col for col in categorical_cols if df[col].nunique() > 7]

ohe = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
ohe_encoded = ohe.fit_transform(df[high_card_cols])
ohe_col_names = ohe.get_feature_names_out(high_card_cols)
df_ohe = pd.DataFrame(ohe_encoded, columns=ohe_col_names, index=df.index)
```

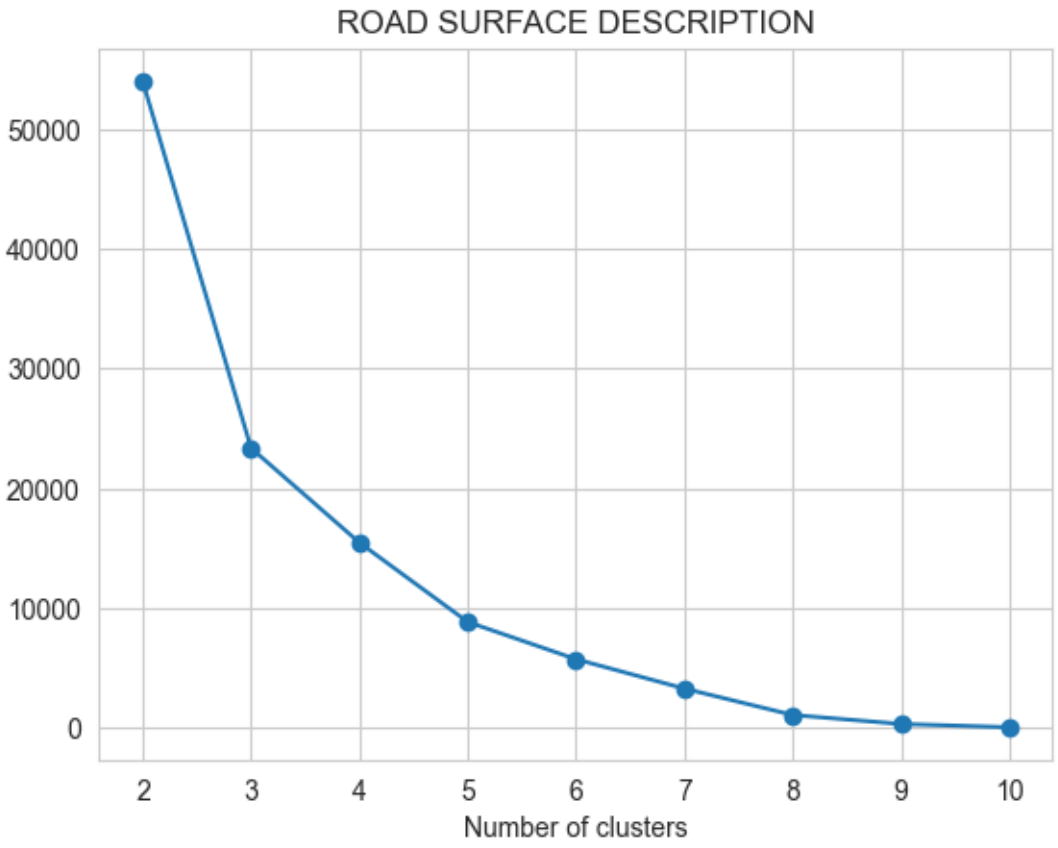
[80]:
```
high_card_cols
```

[80]:
```
['MANNER OF IMPACT DESCRIPTION',
 'ROAD SURFACE DESCRIPTION',
 'LIGHTING CONDITION DESCRIPTION',
 'WEATHER 1 DESCRIPTION',
 'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION']
```

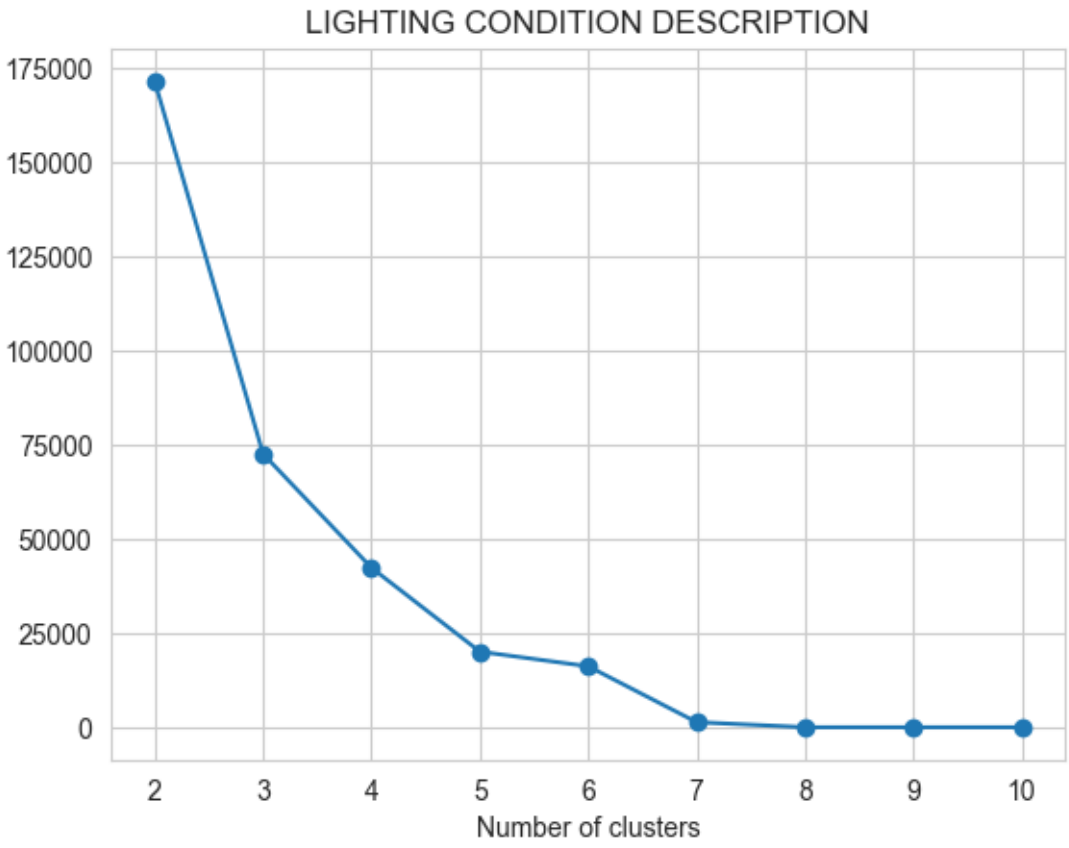
[81]:
```
def elbow_plot(df, col, max_k=10):
    values = df[col].dropna().astype(str)
    enc = OneHotEncoder(handle_unknown='ignore')
    encoded = enc.fit_transform(values.values.reshape(-1, 1))
    inertias = []
    for k in range(2, max_k+1):
        km = KMeans(n_clusters=k, random_state=42)
        km.fit(encoded)
        inertias.append(km.inertia_)
    plt.plot(range(2, max_k+1), inertias, marker='o')
    plt.title(f'{col}')
    plt.xlabel('Number of clusters')
    plt.grid(True)
    plt.show()
```

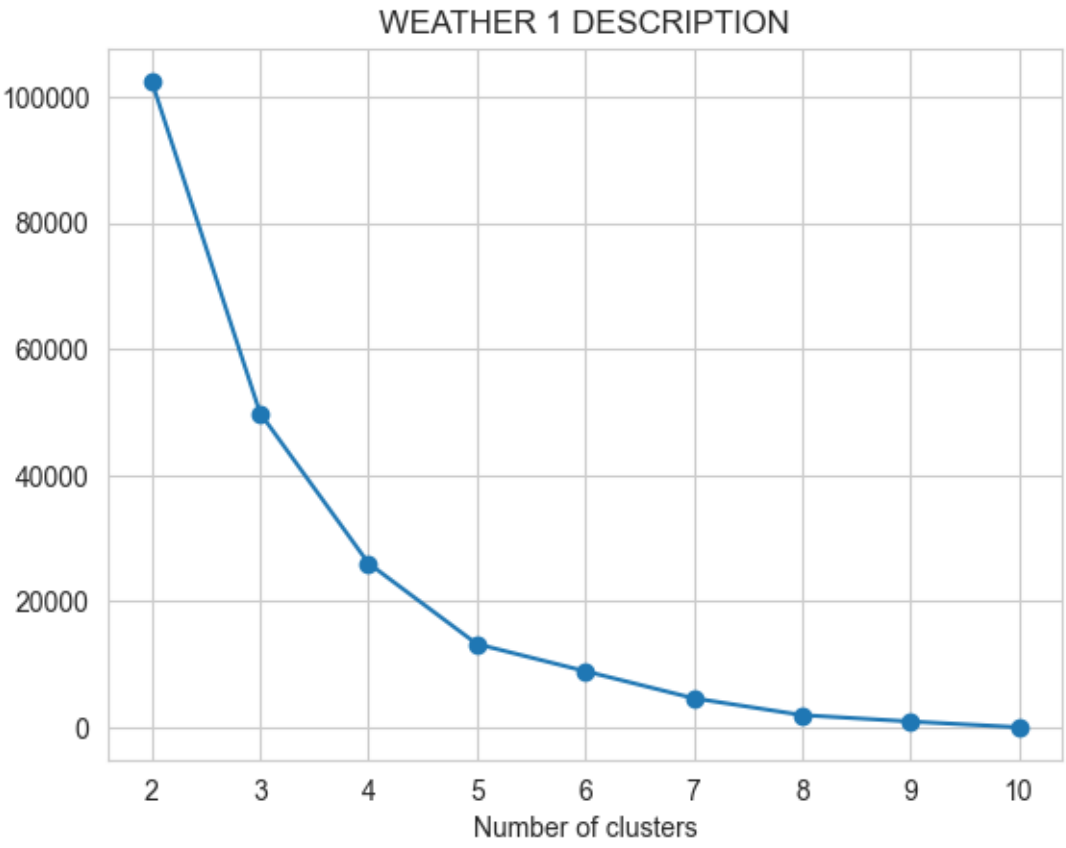
[82]:
```
for col in high_card_cols:
    elbow_plot(df, col, max_k=10)
```

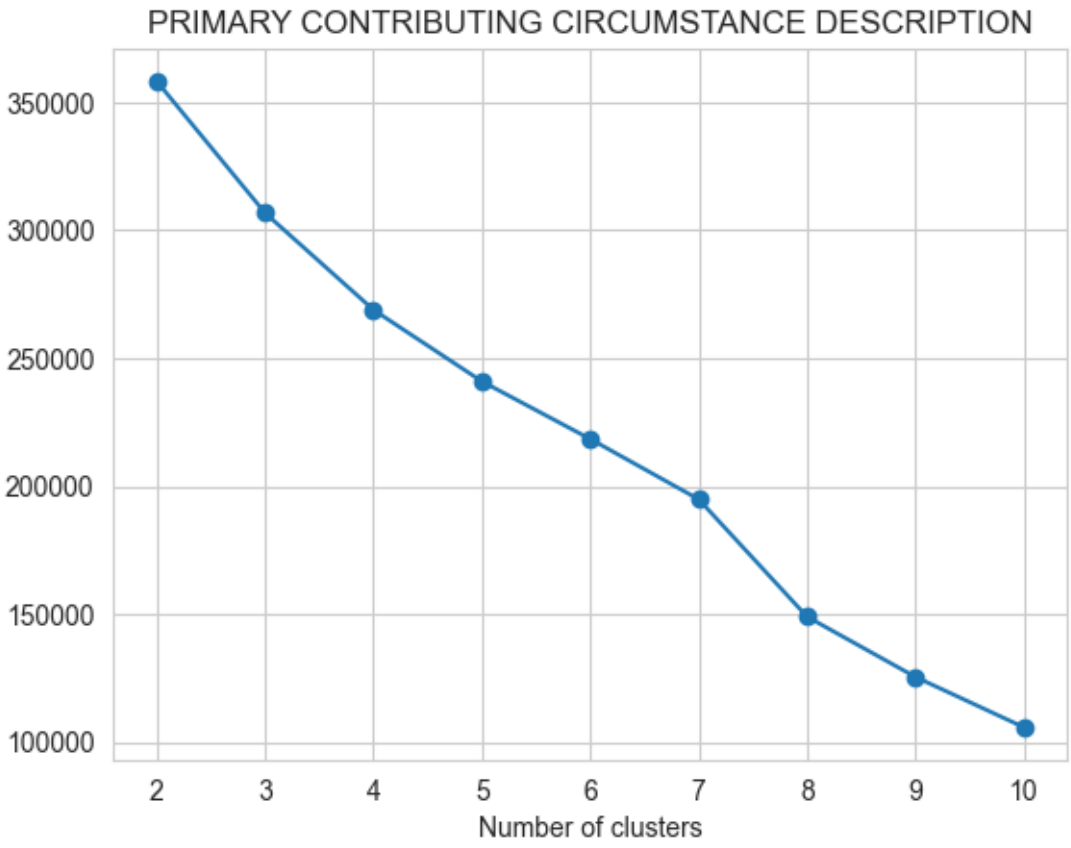

MANNER OF IMPACT DESCRIPTION
300000
250000
200000
150000
100000
50000
0
2
3
4
5
6
7
8
9
10
Number of clusters




```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/base.py:1473:
ConvergenceWarning: Number of distinct clusters (8) found smaller than
n_clusters (9). Possibly due to duplicate points in X.
  return fit_method(estimator, *args, **kwargs)
/opt/anaconda3/lib/python3.12/site-packages/sklearn/base.py:1473:
ConvergenceWarning: Number of distinct clusters (8) found smaller than
n_clusters (10). Possibly due to duplicate points in X.
  return fit_method(estimator, *args, **kwargs)
```


LIGHTING CONDITION DESCRIPTION
175000
150000
125000
100000
75000
50000
25000
0
2
3
4
5
6
7
8
9
10
Number of clusters


WEATHER 1 DESCRIPTION
100000
80000
60000
40000
20000
0
2
3
4
5
6
7
8
9
10
Number of clusters



[83]:
```
cluster_df = pd.DataFrame(index=df.index)
best_k = {
    'MANNER OF IMPACT DESCRIPTION': 5,
    'ROAD SURFACE DESCRIPTION': 4,
    'LIGHTING CONDITION DESCRIPTION': 3,
    'WEATHER 1 DESCRIPTION': 4,
    'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION': 5
}

for col, k in best_k.items():
    col_ohe = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
    col_encoded = col_ohe.fit_transform(df[[col]])
    kmeans = KMeans(n_clusters=k, random_state=3)
    clusters = kmeans.fit_predict(col_encoded)
    cluster_df[f'{col}_CLUSTER'] = clusters

df = pd.concat([df.reset_index(drop=True), cluster_df.reset_index(drop=True)],␣
 ↪axis=1)
```

[84]:
```
pd.set_option('display.max_columns', 500)
df.drop(['CRASH DATETIME', 'LATITUDE', 'LONGITUDE', 'the_geom', 'hour',␣
 ↪'dayofweek', 'month', 'year'], axis=1, inplace=True)
df
```

[84]:

```
        DAY OF WEEK CODE DAY OF WEEK DESCRIPTION  CRASH CLASSIFICATION CODE  \
0                      5                Thursday                         2
1                      1                  Sunday                         3
2                      6                  Friday                         2
3                      3                 Tuesday                         1
4                      2                  Monday                         1
…                    …                        …                       …
536560                 5                Thursday                         2
536561                 5                Thursday                         2
536562                 1                  Sunday                         2
536563                 2                  Monday                         2
536564                 5                Thursday                         1

       CRASH CLASSIFICATION DESCRIPTION COLLISION ON PRIVATE PROPERTY  \
0                  Property Damage Only                            N
1                 Personal Injury Crash                            N
2                  Property Damage Only                            N
3                        Non-Reportable                            Y
4                        Non-Reportable                            Y
…                                   …                       …
536560             Property Damage Only                            N
536561             Property Damage Only                            N
536562             Property Damage Only                            N
536563             Property Damage Only                            N
536564                   Non-Reportable                            N

       PEDESTRIAN INVOLVED  MANNER OF IMPACT CODE  \
0                        N                    1.0
1                        N                    3.0
2                        N                   99.0
3                        N                    4.0
4                        N                    1.0
…                      …                      …
536560                   N                    1.0
536561                   N                    1.0
536562                   N                    1.0
536563                   N                    4.0
536564                   N                    1.0

       MANNER OF IMPACT DESCRIPTION ALCOHOL INVOLVED DRUG INVOLVED  \
0                      Front to rear                N             N
1                              Angle                N             N
```

```
2                          Unknown                N             N
3         Sideswipe, same direction               N             N
4                     Front to rear                N             N
…                               …                  …             …
536560                Front to rear                N             N
536561                Front to rear                N             N
536562                Front to rear                N             N
536563    Sideswipe, same direction                N             N
536564                Front to rear                N             N

        ROAD SURFACE CODE ROAD SURFACE DESCRIPTION  LIGHTING CONDITION CODE  \
0                     1.0                      Dry                      1.0
1                     1.0                      Dry                      1.0
2                     1.0                      Dry                      1.0
3                     1.0                      Dry                      1.0
4                     1.0                      Dry                      1.0
…                   …                          …                        …
536560                1.0                      Dry                      1.0
536561                1.0                      Dry                      1.0
536562                1.0                      Dry                      1.0
536563                1.0                      Dry                      1.0
536564                1.0                      Dry                      1.0

       LIGHTING CONDITION DESCRIPTION  WEATHER 1 CODE WEATHER 1 DESCRIPTION  \
0                            Daylight             1.0                 Clear
1                            Daylight             2.0                Cloudy
2                            Daylight             1.0                 Clear
3                            Daylight             2.0                Cloudy
4                            Daylight             1.0                 Clear
…                               …                 …                    …
536560                       Daylight             2.0                Cloudy
536561                       Daylight             2.0                Cloudy
536562                       Daylight             1.0                 Clear
536563                       Daylight             2.0                Cloudy
536564                       Daylight             1.0                 Clear

       SEATBELT USED MOTORCYCLE INVOLVED MOTORCYCLE HELMET USED  \
0                  Y                    N                      N
1                  Y                    N                      N
2                  Y                    N                      N
3                  Y                    N                      N
4                  Y                    N                      N
…                …                      …                      …
536560             Y                    N                      N
536561             Y                    N                      N
536562             Y                    N                      N
536563             Y                    N                      N
```

```
536564             Y                    N                      N

       BICYCLED INVOLVED BICYCLE HELMET USED  \
0                      N                    N
1                      N                    N
2                      N                    N
3                      N                    N
4                      N                    N
…                    …                    …
536560                 N                    N
536561                 N                    N
536562                 N                    N
536563                 N                    N
536564                 N                    N

        PRIMARY CONTRIBUTING CIRCUMSTANCE CODE  \
0                                         8.0
1                                        12.0
2                                        99.0
3                                        11.0
4                                        88.0
…                                         …
536560                                   11.0
536561                                   11.0
536562                                   11.0
536563                                   11.0
536564                                   17.0

       PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION  \
0                                Following too close
1           Driving in a careless or reckless manner
2                                            Unknown
3         Driver inattention, distraction, or fatigue
4                                              Other
…                                                …
536560    Driver inattention, distraction, or fatigue
536561    Driver inattention, distraction, or fatigue
536562    Driver inattention, distraction, or fatigue
536563    Driver inattention, distraction, or fatigue
536564                      Animal in Roadway - Deer

        SCHOOL BUS INVOLVED CODE SCHOOL BUS INVOLVED DESCRIPTION WORK ZONE  \
0                            0.0                              No         N
1                            0.0                              No         N
2                            0.0                              No         N
3                            0.0                              No         N
4                            0.0                              No         N
```

```
…                          …                                …       …
536560                       0.0                             No         N
536561                       0.0                             No         N
536562                       0.0                             No         N
536563                       0.0                             No         N
536564                       0.0                             No         N

       WORKERS PRESENT COUNTY CODE COUNTY NAME  \
0                    N           S      Sussex
1                    N           S      Sussex
2                    N           S      Sussex
3                    N           K        Kent
4                    N           S      Sussex
…                  …           …          …
536560               N           N  New Castle
536561               N           N  New Castle
536562               N           N  New Castle
536563               N           N  New Castle
536564               N           S      Sussex

        MANNER OF IMPACT DESCRIPTION_CLUSTER  \
0                                          1
1                                          0
2                                          3
3                                          4
4                                          1
…                                        …
536560                                     1
536561                                     1
536562                                     1
536563                                     4
536564                                     1

        ROAD SURFACE DESCRIPTION_CLUSTER  \
0                                       0
1                                       0
2                                       0
3                                       0
4                                       0
…                                     …
536560                                  0
536561                                  0
536562                                  0
536563                                  0
536564                                  0

        LIGHTING CONDITION DESCRIPTION_CLUSTER  WEATHER 1 DESCRIPTION_CLUSTER  \
```

```
0                                                0                                0
1                                                0                                1
2                                                0                                0
3                                                0                                1
4                                                0                                0
…                                              …                                …
536560                                           0                                1
536561                                           0                                1
536562                                           0                                0
536563                                           0                                1
536564                                           0                                0

        PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_CLUSTER
0                                                        1
1                                                        0
2                                                        2
3                                                        0
4                                                        3
…                                                      …
536560                                                   0
536561                                                   0
536562                                                   0
536563                                                   0
536564                                                   0

[536565 rows x 34 columns]
```

4.1 Modeling

[86]:
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.feature_selection import SelectFromModel
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder
import numpy as np

X = df[categorical_cols]
y = df['CRASH CLASSIFICATION DESCRIPTION']

le = LabelEncoder()
y= le.fit_transform(y)

encoder = ColumnTransformer([
    ('onehot', OneHotEncoder(), categorical_cols)
```

```
])

xgb = XGBClassifier(random_state = 42, eval_metric='mlogloss')

pipe = Pipeline([
    ('encode', encoder),
    ('model', xgb)
])
```

[87]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state = 3)

pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)
```

[88]:
```
print("Without clusters:")
print(classification_report(y_test, y_pred, target_names=le.classes_))
```

```
Without clusters:
                      precision    recall  f1-score   support

       Fatality Crash       0.61      0.18      0.28       318
       Non-Reportable       0.80      0.47      0.59     29605
Personal Injury Crash       0.67      0.14      0.23     16801
 Property Damage Only       0.65      0.93      0.77     60589

             accuracy                           0.68    107313
            macro avg       0.68      0.43      0.47    107313
         weighted avg       0.70      0.68      0.63    107313
```

[89]:
```
categorical_cols = [col for col in df.columns if col[-4:] != 'CODE' and col not␣
 ↪in ['CRASH DATETIME', 'LATITUDE', 'LONGITUDE', 'the_geom', 'CRASH␣
 ↪CLASSIFICATION DESCRIPTION', 'hour', 'dayofweek', 'month', 'year',␣
 ↪'COUNTY_NAME']]
categorical_cols
```

[89]:
```
['DAY OF WEEK DESCRIPTION',
 'COLLISION ON PRIVATE PROPERTY',
 'PEDESTRIAN INVOLVED',
 'MANNER OF IMPACT DESCRIPTION',
 'ALCOHOL INVOLVED',
 'DRUG INVOLVED',
 'ROAD SURFACE DESCRIPTION',
 'LIGHTING CONDITION DESCRIPTION',
 'WEATHER 1 DESCRIPTION',
 'SEATBELT USED',
 'MOTORCYCLE INVOLVED',
```

```
 'MOTORCYCLE HELMET USED',
 'BICYCLED INVOLVED',
 'BICYCLE HELMET USED',
 'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION',
 'SCHOOL BUS INVOLVED DESCRIPTION',
 'WORK ZONE',
 'WORKERS PRESENT',
 'COUNTY NAME',
 'MANNER OF IMPACT DESCRIPTION_CLUSTER',
 'ROAD SURFACE DESCRIPTION_CLUSTER',
 'LIGHTING CONDITION DESCRIPTION_CLUSTER',
 'WEATHER 1 DESCRIPTION_CLUSTER',
 'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_CLUSTER']
```

[90]:
```
X = df[categorical_cols]
y = df['CRASH CLASSIFICATION DESCRIPTION']

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y= le.fit_transform(y)

encoder = ColumnTransformer([
    ('onehot', OneHotEncoder(), categorical_cols)
])

xgb = XGBClassifier(random_state = 42, eval_metric='mlogloss')

pipe = Pipeline([
    ('encode', encoder),
    ('model', xgb)
])
```

[91]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state = 3)

pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)
```

[92]:
```
print("With clusters:")
print(classification_report(y_test, y_pred, target_names=le.classes_))
```

```
With clusters:
                      precision    recall  f1-score   support

       Fatality Crash       0.60      0.19      0.28       318
       Non-Reportable       0.80      0.47      0.59     29605
Personal Injury Crash       0.67      0.14      0.23     16801
 Property Damage Only       0.65      0.94      0.77     60589
```

accuracy			0.68	107313
macro avg	0.68	0.43	0.47	107313
weighted avg	0.70	0.68	0.63	107313

[93]:
```
categorical_cols = [col for col in df.columns if col[-4:] != 'CODE' and col not␣
 ↪in ['CRASH DATETIME', 'LATITUDE', 'LONGITUDE', 'the_geom', 'CRASH␣
 ↪CLASSIFICATION DESCRIPTION', 'hour', 'dayofweek', 'month', 'year']]

X = df[categorical_cols]
y = df['CRASH CLASSIFICATION DESCRIPTION']

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=3)

categorical_transformer = OneHotEncoder()

preprocessor = ColumnTransformer(
    transformers=[
        ('cat', categorical_transformer, categorical_cols)
    ]
)

feature_selector = SelectFromModel(
    XGBClassifier(
        objective='multi:softmax',
        num_class=len(np.unique(y)),
        eval_metric='mlogloss',
        random_state=3
    ),
    threshold='median'
)

pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('feature_selection', feature_selector),
    ('classifier', XGBClassifier(
        objective='multi:softmax',
        num_class=len(np.unique(y)),
        eval_metric='mlogloss',
        random_state=3
    ))
```

```
])

pipeline.fit(X_train, y_train)

y_pred = pipeline.predict(X_test)
print("Best Selection")
print(classification_report(y_test, y_pred, target_names=le.classes_))
```

```
Best Selection
                      precision    recall  f1-score   support

       Fatality Crash       0.58      0.16      0.26       318
       Non-Reportable       0.80      0.46      0.59     29605
Personal Injury Crash       0.67      0.14      0.23     16801
 Property Damage Only       0.65      0.94      0.77     60589

             accuracy                           0.68    107313
            macro avg       0.68      0.43      0.46    107313
         weighted avg       0.70      0.68      0.63    107313
```

[94]:
```
ohe = pipeline.named_steps['preprocessor'].named_transformers_['cat']
encoded_feature_names = ohe.get_feature_names_out(categorical_cols)
encoded_feature_names[pipeline.named_steps['feature_selection'].get_support()␣
↪== True]
```

[94]:
```
array(['COLLISION ON PRIVATE PROPERTY_N', 'PEDESTRIAN INVOLVED_N',
       'MANNER OF IMPACT DESCRIPTION_Angle',
       'MANNER OF IMPACT DESCRIPTION_Front to front',
       'MANNER OF IMPACT DESCRIPTION_Front to rear',
       'MANNER OF IMPACT DESCRIPTION_Not a collision between two vehicles',
       'MANNER OF IMPACT DESCRIPTION_Other',
       'MANNER OF IMPACT DESCRIPTION_Rear to rear',
       'MANNER OF IMPACT DESCRIPTION_Rear to side',
       'MANNER OF IMPACT DESCRIPTION_Sideswipe, opposite direction',
       'MANNER OF IMPACT DESCRIPTION_Sideswipe, same direction',
       'MANNER OF IMPACT DESCRIPTION_Unknown', 'ALCOHOL INVOLVED_N',
       'DRUG INVOLVED_N', 'ROAD SURFACE DESCRIPTION_Dry',
       'ROAD SURFACE DESCRIPTION_Snow',
       'ROAD SURFACE DESCRIPTION_Unknown', 'ROAD SURFACE DESCRIPTION_Wet',
       'LIGHTING CONDITION DESCRIPTION_Dark-Lighted',
       'LIGHTING CONDITION DESCRIPTION_Dark-Not Lighted',
       'LIGHTING CONDITION DESCRIPTION_Daylight',
       'LIGHTING CONDITION DESCRIPTION_Unknown',
       'WEATHER 1 DESCRIPTION_Clear', 'SEATBELT USED_N',
       'MOTORCYCLE INVOLVED_N', 'MOTORCYCLE HELMET USED_N',
       'BICYCLED INVOLVED_N',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Animal in Roadway - Deer',
```

```
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Animal in Roadway - Other
Animal',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Disregard Traffic Signal',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Driver inattention,
distraction, or fatigue',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Driving in a careless or
reckless manner',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Driving in an aggressive
manner',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Driving under the
influence',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Failed to yield right of
way',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Following too close',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Improper backing',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Improper lane change',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Improper passing',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Made improper turn',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Mechanical defects',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Other',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Other environmental
circumstances - weather, glare',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Other improper driving',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Passed Stop Sign',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Pedestrian',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Roadway circumstances -
debris, holes, work zone,',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Speeding',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Unknown',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_Wrong side or wrong way',
       'SCHOOL BUS INVOLVED DESCRIPTION_No', 'WORK ZONE_N',
       'WORKERS PRESENT_2', 'WORKERS PRESENT_N', 'COUNTY NAME_Kent',
       'COUNTY NAME_New Castle', 'MANNER OF IMPACT DESCRIPTION_CLUSTER_1',
       'ROAD SURFACE DESCRIPTION_CLUSTER_1',
       'LIGHTING CONDITION DESCRIPTION_CLUSTER_0',
       'WEATHER 1 DESCRIPTION_CLUSTER_3',
       'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION_CLUSTER_0'],
      dtype=object)
```

5 Baseline

[96]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix

# Models
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
import xgboost as xgb

from imblearn.under_sampling import RandomUnderSampler
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE

# Set a random seed for reproducibility
RANDOM_STATE = 42
```

[97]:
```
df = pd.read_csv('clean_crash_data.csv')
```

[98]:
```
target_variable = 'CRASH CLASSIFICATION CODE'
```

[99]:
```
y = df[target_variable]
drop_columns = [target_variable, 'the_geom', 'CRASH DATETIME', 'CRASH␣
 ↪CLASSIFICATION DESCRIPTION']
X = df.drop(columns=drop_columns, axis=1)
```

[100]:
```
numerical_features = X.select_dtypes(include=np.number).columns.tolist()
categorical_features = X.select_dtypes(include=['object', 'category']).columns.
 ↪tolist()
```

[101]:
```
print(f"\nIdentified Numerical features for preprocessing:␣
 ↪{numerical_features}")
print(f"Identified Categorical features for preprocessing:␣
 ↪{categorical_features}")
```

```
Identified Numerical features for preprocessing: ['DAY OF WEEK CODE', 'MANNER OF
IMPACT CODE', 'ROAD SURFACE CODE', 'LIGHTING CONDITION CODE', 'WEATHER 1 CODE',
'LATITUDE', 'LONGITUDE', 'PRIMARY CONTRIBUTING CIRCUMSTANCE CODE', 'SCHOOL BUS
INVOLVED CODE', 'hour', 'month', 'year']
Identified Categorical features for preprocessing: ['DAY OF WEEK DESCRIPTION',
'COLLISION ON PRIVATE PROPERTY', 'PEDESTRIAN INVOLVED', 'MANNER OF IMPACT
DESCRIPTION', 'ALCOHOL INVOLVED', 'DRUG INVOLVED', 'ROAD SURFACE DESCRIPTION',
'LIGHTING CONDITION DESCRIPTION', 'WEATHER 1 DESCRIPTION', 'SEATBELT USED',
```

```
'MOTORCYCLE INVOLVED', 'MOTORCYCLE HELMET USED', 'BICYCLED INVOLVED', 'BICYCLE
HELMET USED', 'PRIMARY CONTRIBUTING CIRCUMSTANCE DESCRIPTION', 'SCHOOL BUS
INVOLVED DESCRIPTION', 'WORK ZONE', 'WORKERS PRESENT', 'COUNTY CODE', 'COUNTY
NAME', 'dayofweek']
```

[102]:
```
y_processed = y - 1
```

[103]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y_processed, test_size=0.2, random_state=RANDOM_STATE,␣
 ↪stratify=y_processed
)
```

[104]:
```
categorical_transformer = OneHotEncoder()

preprocessor = ColumnTransformer(
    transformers=[
        ('cat', categorical_transformer, categorical_features)
    ]
)

if preprocessor == 'passthrough':
    print("Preprocessor is 'passthrough'. Converting X_train/X_test to numpy␣
 ↪arrays if they are DataFrames.")
    X_train_processed = X_train.values if hasattr(X_train, 'values') else␣
 ↪X_train
    X_test_processed = X_test.values if hasattr(X_test, 'values') else X_test
else:
    X_train_processed = preprocessor.fit_transform(X_train)
    X_test_processed = preprocessor.transform(X_test)
```

[105]:
```
print(f"X_train shape: {X_train.shape}, X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}, y_test shape: {y_test.shape}")
```

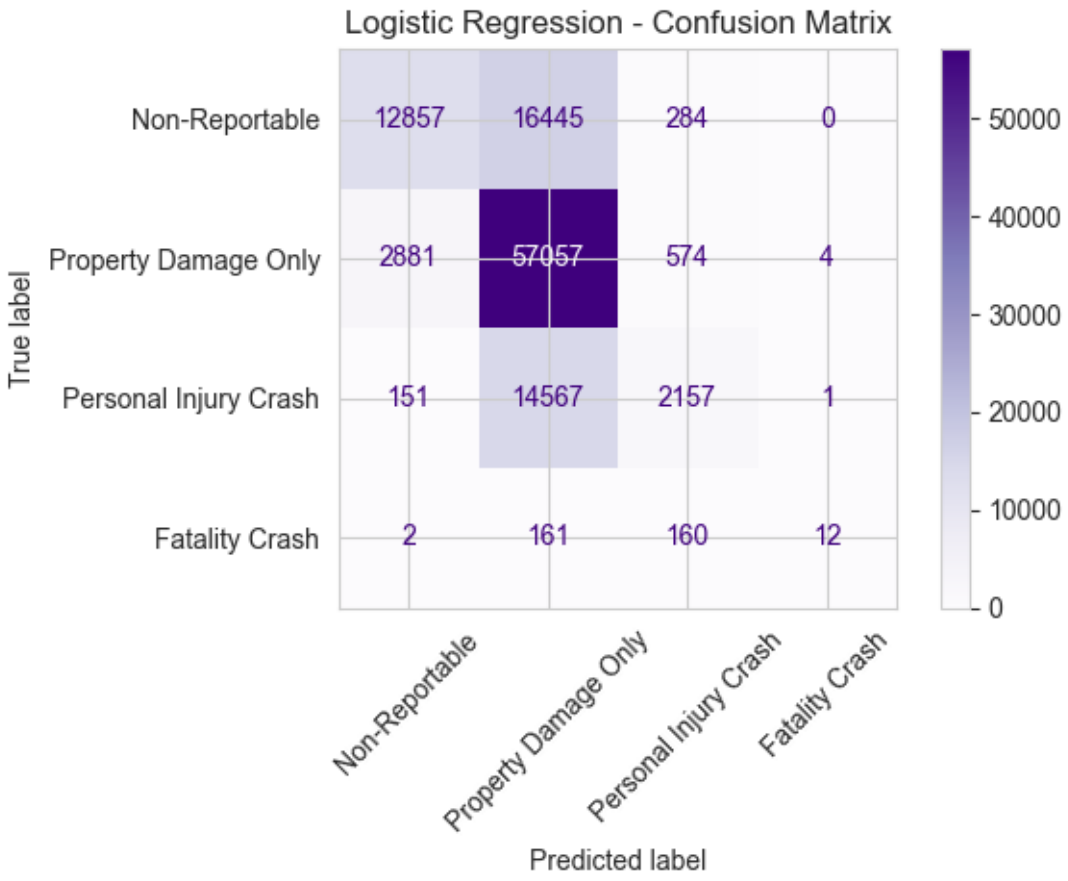
```
X_train shape: (429252, 33), X_test shape: (107313, 33)
y_train shape: (429252,), y_test shape: (107313,)
```

[106]:
```
model_performance = {}
label_mapping = {
    0: "Non-Reportable",
    1: "Property Damage Only",
    2: "Personal Injury Crash",
    3: "Fatality Crash"
}
```

5.1 Logistic Regression (base)

[108]:
```
print("\n--- Logistic Regression ---")
log_reg = LogisticRegression(random_state=RANDOM_STATE, max_iter=1000,␣
 ↪solver='liblinear')
log_reg.fit(X_train_processed, y_train)
y_pred_log_reg = log_reg.predict(X_test_processed)
accuracy_log_reg = accuracy_score(y_test, y_pred_log_reg)
report_log_reg = classification_report(y_test, y_pred_log_reg,␣
 ↪target_names=[label_mapping[i] for i in sorted(label_mapping)],␣
 ↪zero_division=0)
model_performance['Logistic Regression'] = accuracy_log_reg
print(f"Accuracy: {accuracy_log_reg:.4f}")
print("Classification Report:\n", report_log_reg)

#Plot
cm_log_reg = confusion_matrix(y_test, y_pred_log_reg)
disp_log_reg = ConfusionMatrixDisplay(confusion_matrix=cm_log_reg,␣
 ↪display_labels=[label_mapping[i] for i in sorted(label_mapping)])
disp_log_reg.plot(cmap=plt.cm.Purples, xticks_rotation=45)
plt.title("Logistic Regression - Confusion Matrix")
plt.tight_layout()
plt.show()
```

```
--- Logistic Regression ---
Accuracy: 0.6717
Classification Report:
                        precision    recall  f1-score   support

       Non-Reportable       0.81      0.43      0.57     29586
 Property Damage Only       0.65      0.94      0.77     60516
Personal Injury Crash       0.68      0.13      0.22     16876
       Fatality Crash       0.71      0.04      0.07       335

             accuracy                           0.67    107313
            macro avg       0.71      0.39      0.40    107313
         weighted avg       0.70      0.67      0.62    107313
```

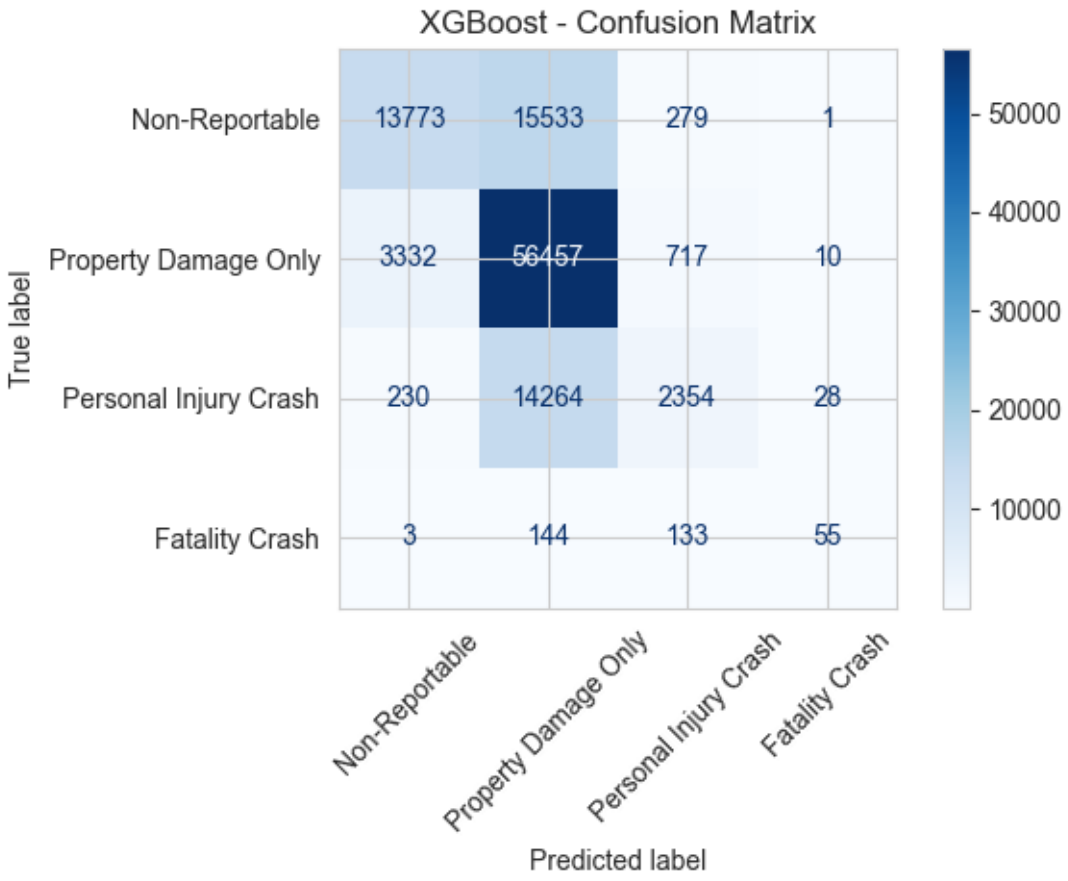


5.2 XGBoost (base)

[110]:
```
print("\n--- XGBoost ---")
# y_train should be 0-indexed for XGBoost, which y_processed should be.
xgb_clf = xgb.XGBClassifier(random_state=RANDOM_STATE, use_label_encoder=False,␣
 ↪eval_metric='mlogloss')
xgb_clf.fit(X_train_processed, y_train)
y_pred_xgb = xgb_clf.predict(X_test_processed)
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
report_xgb = classification_report(y_test, y_pred_xgb,␣
 ↪target_names=[label_mapping[i] for i in sorted(label_mapping)],␣
 ↪zero_division=0)
model_performance['XGBoost'] = accuracy_xgb
print(f"Accuracy: {accuracy_xgb:.4f}")
print("Classification Report:\n", report_xgb)

#Plot
cm_xgb = confusion_matrix(y_test, y_pred_xgb)
```

```
disp_xgb = ConfusionMatrixDisplay(confusion_matrix=cm_xgb,␣
 ↪display_labels=[label_mapping[i] for i in sorted(label_mapping)])
disp_xgb.plot(cmap=plt.cm.Blues, xticks_rotation=45)
plt.title("XGBoost - Confusion Matrix")
plt.tight_layout()
plt.show()
```

```
--- XGBoost ---
/opt/anaconda3/lib/python3.12/site-packages/xgboost/core.py:158: UserWarning:
[19:34:35] WARNING: /Users/runner/work/xgboost/xgboost/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
Accuracy: 0.6769
Classification Report:
                        precision    recall  f1-score   support

       Non-Reportable       0.79      0.47      0.59     29586
 Property Damage Only       0.65      0.93      0.77     60516
Personal Injury Crash       0.68      0.14      0.23     16876
       Fatality Crash       0.59      0.16      0.26       335

             accuracy                           0.68    107313
            macro avg       0.68      0.43      0.46    107313
         weighted avg       0.70      0.68      0.63    107313
```

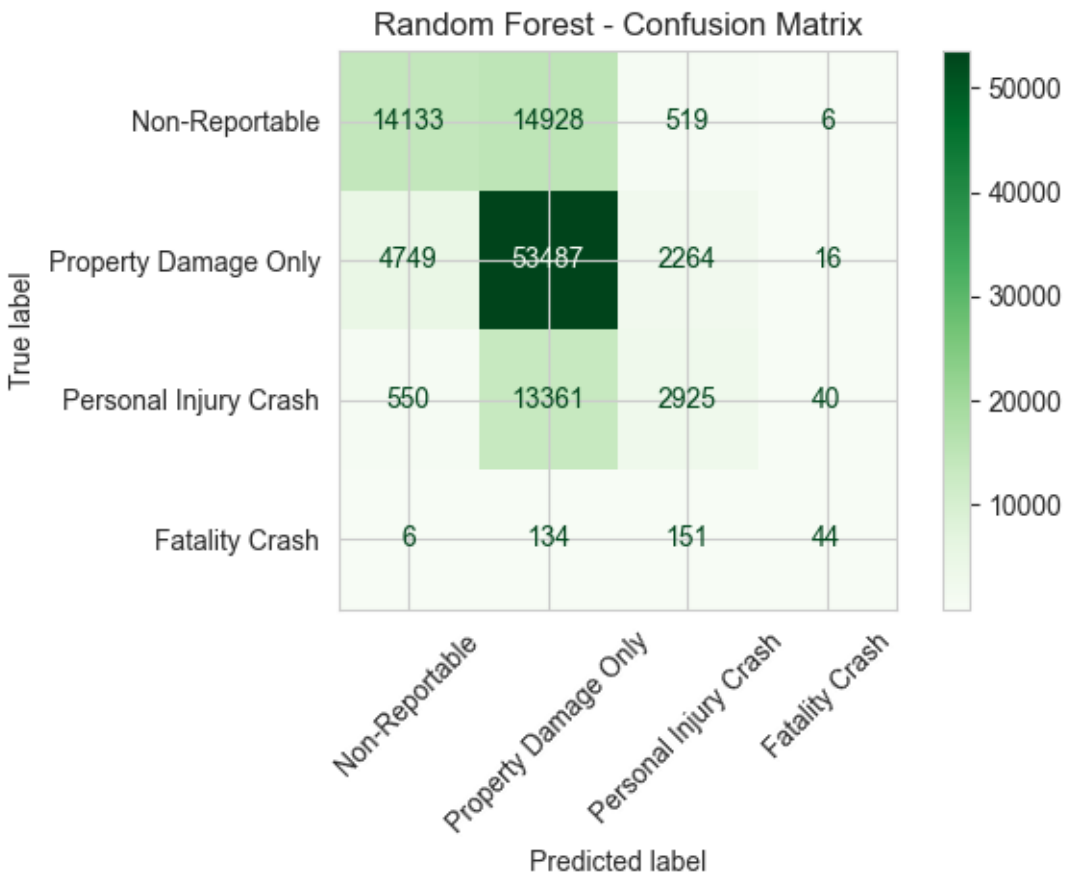


5.3 Random Forest (base)

[115]:
```
print("\n--- Random Forest ---")
rf_clf = RandomForestClassifier(random_state=RANDOM_STATE,␣
 ↪n_estimators=100,n_jobs=-1)
rf_clf.fit(X_train_processed, y_train)
y_pred_rf = rf_clf.predict(X_test_processed)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
report_rf = classification_report(y_test, y_pred_rf,␣
 ↪target_names=[label_mapping[i] for i in sorted(label_mapping)],␣
 ↪zero_division=0)
model_performance['Random Forest'] = accuracy_rf
print(f"Accuracy: {accuracy_rf:.4f}")
print("Classification Report:\n", report_rf)

#Plot
cm_rf = confusion_matrix(y_test, y_pred_rf)
```

```
disp_rf = ConfusionMatrixDisplay(confusion_matrix=cm_rf,␣
 ↪display_labels=[label_mapping[i] for i in sorted(label_mapping)])
disp_rf.plot(cmap=plt.cm.Greens, xticks_rotation=45)
plt.title("Random Forest - Confusion Matrix")
plt.tight_layout()
plt.show()
```

```
--- Random Forest ---
Accuracy: 0.6578
Classification Report:
                        precision    recall  f1-score   support

       Non-Reportable       0.73      0.48      0.58     29586
 Property Damage Only       0.65      0.88      0.75     60516
Personal Injury Crash       0.50      0.17      0.26     16876
       Fatality Crash       0.42      0.13      0.20       335

             accuracy                           0.66    107313
            macro avg       0.57      0.42      0.45    107313
         weighted avg       0.65      0.66      0.62    107313
```



5.4 KNN (base)

[119]:
```
'''
print("\n--- K-Nearest Neighbors (KNN) ---")
knn_clf = KNeighborsClassifier(n_neighbors=5,n_jobs=-1)
knn_clf.fit(X_train_processed, y_train)
y_pred_knn = knn_clf.predict(X_test_processed)
accuracy_knn = accuracy_score(y_test, y_pred_knn)
report_knn = classification_report(y_test, y_pred_knn,␣
↪target_names=[label_mapping[i] for i in sorted(label_mapping)],␣
↪zero_division=0)
model_performance['KNN'] = accuracy_knn
print(f"Accuracy: {accuracy_knn:.4f}")
print("Classification Report:\n", report_knn)

#Plot
cm_knn = confusion_matrix(y_test, y_pred_knn)
disp_knn = ConfusionMatrixDisplay(confusion_matrix=cm_knn,␣
↪display_labels=[label_mapping[i] for i in sorted(label_mapping)])
disp_knn.plot(cmap=plt.cm.PuBuGn, xticks_rotation=45)
plt.title("KNN - Confusion Matrix")
plt.tight_layout()
plt.show()
'''
```

[119]:
```
'\nprint("\n--- K-Nearest Neighbors (KNN) ---")\nknn_clf =
KNeighborsClassifier(n_neighbors=5,n_jobs=-1)\nknn_clf.fit(X_train_processed,
y_train)\ny_pred_knn = knn_clf.predict(X_test_processed)\naccuracy_knn =
accuracy_score(y_test, y_pred_knn)\nreport_knn = classification_report(y_test,
y_pred_knn, target_names=[label_mapping[i] for i in sorted(label_mapping)],
zero_division=0)\nmodel_performance[\'KNN\'] = accuracy_knn\nprint(f"Accuracy:
{accuracy_knn:.4f}")\nprint("Classification Report:\n",
report_knn)\n\n#Plot\ncm_knn = confusion_matrix(y_test, y_pred_knn)\ndisp_knn =
ConfusionMatrixDisplay(confusion_matrix=cm_knn, display_labels=[label_mapping[i]
for i in sorted(label_mapping)])\ndisp_knn.plot(cmap=plt.cm.PuBuGn,
xticks_rotation=45)\nplt.title("KNN - Confusion
Matrix")\nplt.tight_layout()\nplt.show()\n'
```

6 Class Weight

6.1 Logistic Regression (ClassWeight)

[121]:
```
# Balanced version
print("\n--- Logistic Regression (class weight) ---")
log_reg_bal = LogisticRegression(random_state=RANDOM_STATE, max_iter=1000,␣
 ↪solver='liblinear', class_weight='balanced')
log_reg_bal.fit(X_train_processed, y_train)
y_pred_log_reg = log_reg.predict(X_test_processed)
accuracy_log_reg_bal = accuracy_score(y_test, y_pred_log_reg)
report_log_reg_bal = classification_report(y_test, y_pred_log_reg,␣
 ↪target_names=[label_mapping[i] for i in sorted(label_mapping)],␣
 ↪zero_division=0)
model_performance['Logistic Regression'] = accuracy_log_reg
print(f"Accuracy: {accuracy_log_reg:.4f}")
print("Classification Report:\n", report_log_reg)

print(f"Accuracy: {accuracy_log_reg_bal:.4f}")
print("Classification Report:\n", report_log_reg_bal)
```

```
--- Logistic Regression (class weight) ---
Accuracy: 0.6717
Classification Report:
                        precision    recall  f1-score   support

       Non-Reportable       0.81      0.43      0.57     29586
 Property Damage Only       0.65      0.94      0.77     60516
Personal Injury Crash       0.68      0.13      0.22     16876
       Fatality Crash       0.71      0.04      0.07       335

             accuracy                           0.67    107313
            macro avg       0.71      0.39      0.40    107313
         weighted avg       0.70      0.67      0.62    107313

Accuracy: 0.6717
Classification Report:
                        precision    recall  f1-score   support

       Non-Reportable       0.81      0.43      0.57     29586
 Property Damage Only       0.65      0.94      0.77     60516
Personal Injury Crash       0.68      0.13      0.22     16876
       Fatality Crash       0.71      0.04      0.07       335

             accuracy                           0.67    107313
            macro avg       0.71      0.39      0.40    107313
         weighted avg       0.70      0.67      0.62    107313
```

6.2 XGBoost (ClassWeight)

[123]:
```
# Balanced version
print("\n--- XGBoost (class weight) ---")
from sklearn.utils.class_weight import compute_class_weight

# Compute sample weights
classes = np.unique(y_train)
weights = compute_class_weight(class_weight='balanced', classes=classes,␣
 ↪y=y_train)
class_weight_dict = dict(zip(classes, weights))
sample_weights = [class_weight_dict[label] for label in y_train]

xgb_clf_bal = xgb.XGBClassifier(
    random_state=RANDOM_STATE,
    use_label_encoder=False,
    eval_metric='mlogloss',
    objective='multi:softprob',
    num_class=len(classes)
)

xgb_clf_bal.fit(X_train_processed, y_train, sample_weight=sample_weights)
y_pred_xgb_bal = xgb_clf_bal.predict(X_test_processed)

accuracy_xgb_bal = accuracy_score(y_test, y_pred_xgb_bal)
report_xgb_bal = classification_report(y_test, y_pred_xgb_bal,␣
 ↪target_names=[label_mapping[i] for i in sorted(label_mapping)],␣
 ↪zero_division=0)

model_performance['XGBoost (class weight)'] = accuracy_xgb_bal

print(f"Accuracy: {accuracy_xgb_bal:.4f}")
print("Classification Report:\n", report_xgb_bal)
```

```
--- XGBoost (class weight) ---

/opt/anaconda3/lib/python3.12/site-packages/xgboost/core.py:158: UserWarning:
[19:55:11] WARNING: /Users/runner/work/xgboost/xgboost/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)

Accuracy: 0.5403
Classification Report:
                        precision    recall  f1-score   support
```

```
                      Non-Reportable       0.64      0.59      0.61     29586
                Property Damage Only       0.70      0.52      0.60     60516
               Personal Injury Crash       0.29      0.52      0.38     16876
                      Fatality Crash       0.04      0.66      0.08       335

                            accuracy                           0.54    107313
                           macro avg       0.42      0.57      0.42    107313
                        weighted avg       0.62      0.54      0.57    107313
```

6.3 Random Forest (ClassWeight)

[129]:
```
print("\n--- Random Forest (class weight) ---")
rf_bal_clf = RandomForestClassifier(random_state=RANDOM_STATE,␣
 ↪n_estimators=100, class_weight='balanced', n_jobs=-1)
rf_bal_clf.fit(X_train_processed, y_train)
y_pred_rf_bal = rf_bal_clf.predict(X_test_processed)

accuracy_rf_bal = accuracy_score(y_test, y_pred_rf_bal)
report_rf_bal = classification_report(y_test, y_pred_rf_bal,␣
 ↪target_names=[label_mapping[i] for i in sorted(label_mapping)],␣
 ↪zero_division=0)
model_performance['Random Forest (class weight)'] = accuracy_rf_bal

print(f"Accuracy: {accuracy_rf_bal:.4f}")
print("Classification Report:\n", report_rf_bal)
```

```
--- Random Forest (class weight) ---
Accuracy: 0.5604
Classification Report:
                      precision    recall  f1-score   support

      Non-Reportable       0.63      0.57      0.59     29586
Property Damage Only       0.68      0.58      0.63     60516
Personal Injury Crash       0.30      0.48      0.37     16876
      Fatality Crash       0.04      0.25      0.07       335

            accuracy                           0.56    107313
           macro avg       0.41      0.47      0.42    107313
        weighted avg       0.60      0.56      0.58    107313
```

7 Undersampling

[137]:
```
rus = RandomUnderSampler(random_state=RANDOM_STATE)
X_train_resampled, y_train_resampled = rus.fit_resample(X_train, y_train)

categorical_transformer = OneHotEncoder(handle_unknown='ignore')

preprocessor = ColumnTransformer(
    transformers=[
        ('cat', categorical_transformer, categorical_features)
    ]
)
if preprocessor == 'passthrough':
    print("Preprocessor is 'passthrough'. Converting X_train_resampled/X_test␣
 ↪to numpy arrays if they are DataFrames.")
    X_train_processed = X_train.values if hasattr(X_train_resampled, 'values')␣
 ↪else X_train_resampled
    X_test_processed = X_test.values if hasattr(X_test, 'values') else X_test
else:
    X_train_processed = preprocessor.fit_transform(X_train_resampled)
    X_test_processed = preprocessor.transform(X_test)
```

[141]:
```
print(f"X_train_resampled shape: {X_train_resampled.shape}, X_test shape:␣
 ↪{X_test.shape}")
print(f"y_train_resampled shape: {y_train_resampled.shape}, y_test shape:␣
 ↪{y_test.shape}")
```

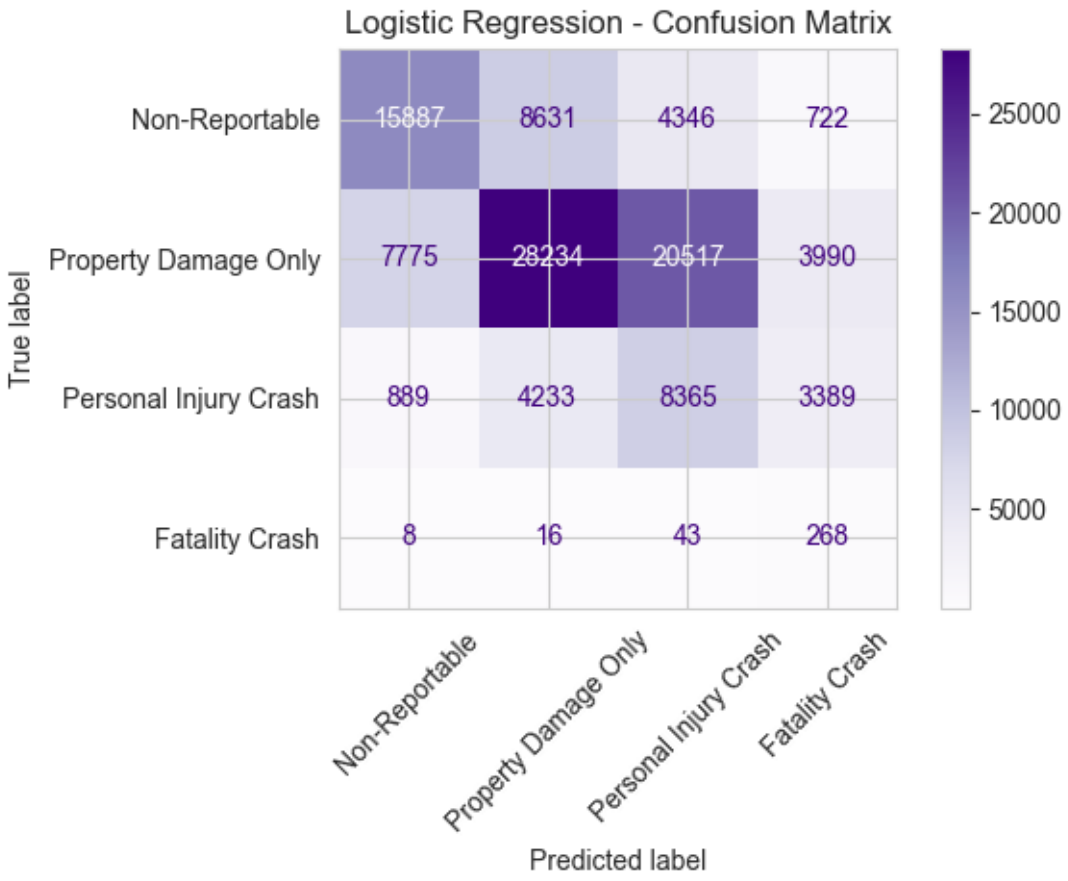
```
X_train_resampled shape: (5368, 33), X_test shape: (107313, 33)
y_train_resampled shape: (5368,), y_test shape: (107313,)
```

7.1 Logistic Regression (undersampling)

[143]:
```
print("\n--- Logistic Regression (undersampling)---")
log_reg = LogisticRegression(random_state=RANDOM_STATE, max_iter=1000,␣
 ↪solver='liblinear', n_jobs=-1)
log_reg.fit(X_train_processed, y_train_resampled)
y_pred_log_reg = log_reg.predict(X_test_processed)
accuracy_log_reg = accuracy_score(y_test, y_pred_log_reg)
report_log_reg = classification_report(y_test, y_pred_log_reg,␣
 ↪target_names=[label_mapping[i] for i in sorted(label_mapping)],␣
 ↪zero_division=0)
model_performance['Logistic Regression (undersampling)'] = accuracy_log_reg
print(f"Accuracy: {accuracy_log_reg:.4f}")
print("Classification Report:\n", report_log_reg)

cm_log_reg = confusion_matrix(y_test, y_pred_log_reg)
```

```
disp_log_reg = ConfusionMatrixDisplay(confusion_matrix=cm_log_reg,␣
 ↪display_labels=[label_mapping[i] for i in sorted(label_mapping)])
disp_log_reg.plot(cmap=plt.cm.Purples, xticks_rotation=45)
plt.title("Logistic Regression - Confusion Matrix")
plt.tight_layout()
plt.show()
```

```
--- Logistic Regression (undersampling)---
Accuracy: 0.4916
Classification Report:
                        precision    recall  f1-score   support

       Non-Reportable       0.65      0.54      0.59     29586
 Property Damage Only       0.69      0.47      0.56     60516
Personal Injury Crash       0.25      0.50      0.33     16876
       Fatality Crash       0.03      0.80      0.06       335

             accuracy                           0.49    107313
            macro avg       0.40      0.57      0.38    107313
         weighted avg       0.61      0.49      0.53    107313

/opt/anaconda3/lib/python3.12/site-
packages/sklearn/linear_model/_logistic.py:1271: UserWarning: 'n_jobs' > 1 does
not have any effect when 'solver' is set to 'liblinear'. Got 'n_jobs' = 8.
  warnings.warn(
```

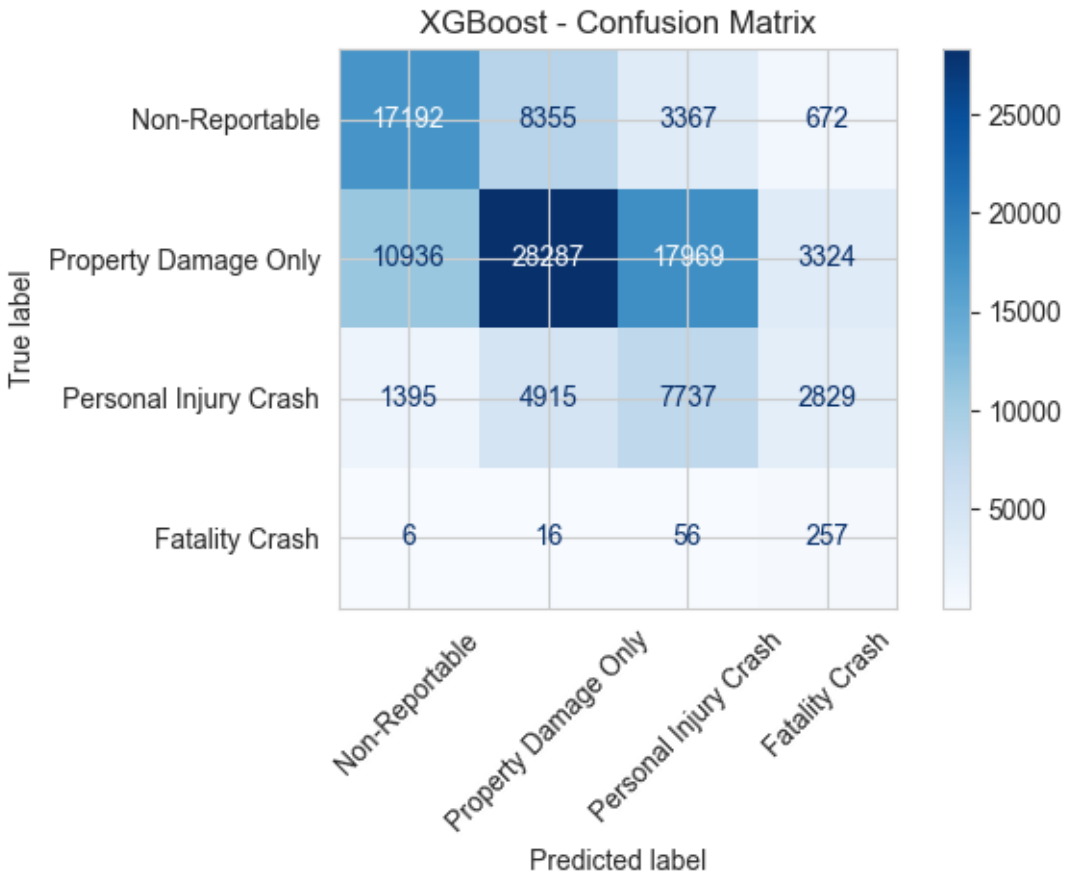


7.2 XGBoost (undersampling)

[145]:
```
print("\n--- XGBoost (undersampling) ---")
# y_train_resampled should be 0-indexed for XGBoost, which y_processed should␣
 ↪be.
xgb_clf = xgb.XGBClassifier(random_state=RANDOM_STATE, use_label_encoder=False,␣
 ↪eval_metric='mlogloss')
xgb_clf.fit(X_train_processed, y_train_resampled)
y_pred_xgb = xgb_clf.predict(X_test_processed)
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
report_xgb = classification_report(y_test, y_pred_xgb,␣
 ↪target_names=[label_mapping[i] for i in sorted(label_mapping)],␣
 ↪zero_division=0)
model_performance['XGBoost (undersampling)'] = accuracy_xgb
print(f"Accuracy: {accuracy_xgb:.4f}")
print("Classification Report:\n", report_xgb)

cm_xgb = confusion_matrix(y_test, y_pred_xgb)
```

```
disp_xgb = ConfusionMatrixDisplay(confusion_matrix=cm_xgb,␣
 ↪display_labels=[label_mapping[i] for i in sorted(label_mapping)])
disp_xgb.plot(cmap=plt.cm.Blues, xticks_rotation=45)
plt.title("XGBoost - Confusion Matrix")
plt.tight_layout()
plt.show()
```

```
--- XGBoost (undersampling) ---
/opt/anaconda3/lib/python3.12/site-packages/xgboost/core.py:158: UserWarning:
[20:05:15] WARNING: /Users/runner/work/xgboost/xgboost/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
Accuracy: 0.4983
Classification Report:
                        precision    recall  f1-score   support

       Non-Reportable       0.58      0.58      0.58     29586
 Property Damage Only       0.68      0.47      0.55     60516
Personal Injury Crash       0.27      0.46      0.34     16876
       Fatality Crash       0.04      0.77      0.07       335

             accuracy                           0.50    107313
            macro avg       0.39      0.57      0.39    107313
         weighted avg       0.59      0.50      0.53    107313
```

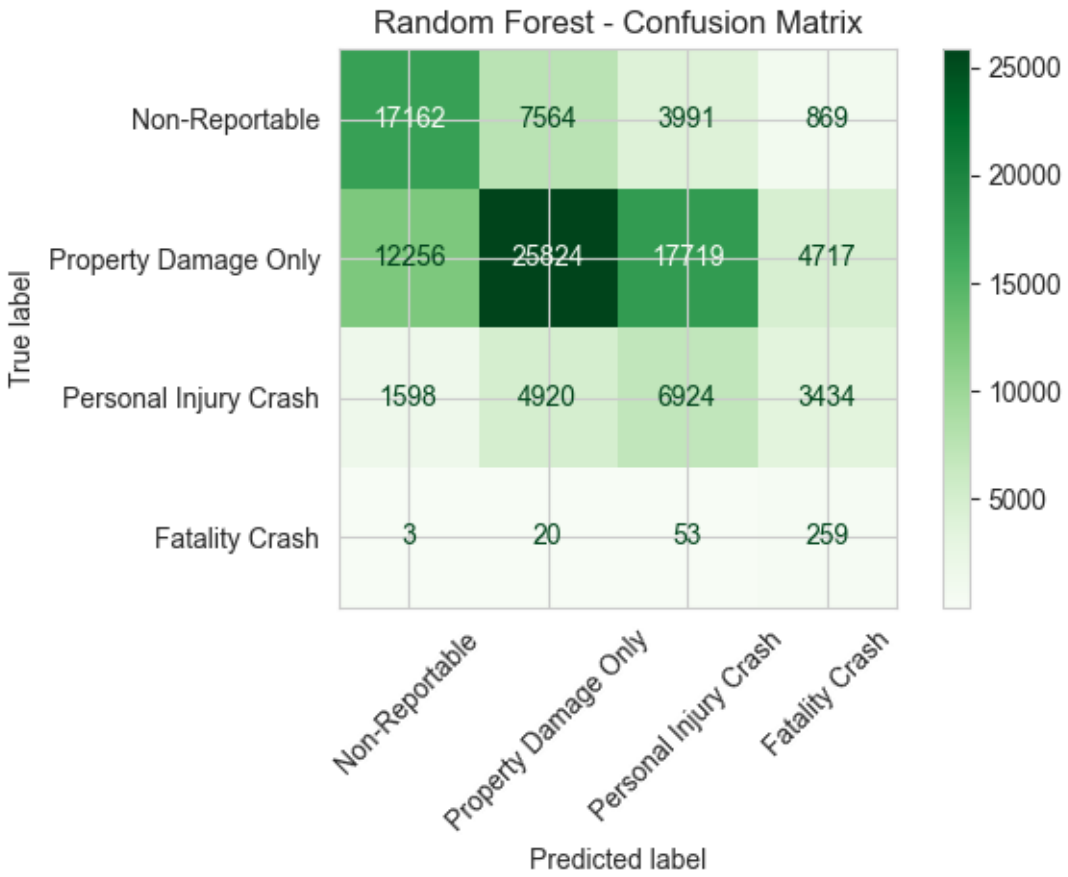


7.3 Random Forest (undersampling)

[147]:
```
print("\n--- Random Forest (undersampling) ---")
rf_clf = RandomForestClassifier(random_state=RANDOM_STATE, n_estimators=100,␣
 ↪n_jobs=-1)
rf_clf.fit(X_train_processed, y_train_resampled)
y_pred_rf = rf_clf.predict(X_test_processed)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
report_rf = classification_report(y_test, y_pred_rf,␣
 ↪target_names=[label_mapping[i] for i in sorted(label_mapping)],␣
 ↪zero_division=0)
model_performance['Random Forest (undersampling)'] = accuracy_rf
print(f"Accuracy: {accuracy_rf:.4f}")
print("Classification Report:\n", report_rf)

cm_rf = confusion_matrix(y_test, y_pred_rf)
disp_rf = ConfusionMatrixDisplay(confusion_matrix=cm_rf,␣
 ↪display_labels=[label_mapping[i] for i in sorted(label_mapping)])
```

```
disp_rf.plot(cmap=plt.cm.Greens, xticks_rotation=45)
plt.title("Random Forest - Confusion Matrix")
plt.tight_layout()
plt.show()
```

```
--- Random Forest (undersampling) ---
Accuracy: 0.4675
Classification Report:
                        precision    recall  f1-score   support

       Non-Reportable       0.55      0.58      0.57     29586
 Property Damage Only       0.67      0.43      0.52     60516
Personal Injury Crash       0.24      0.41      0.30     16876
       Fatality Crash       0.03      0.77      0.05       335

             accuracy                           0.47    107313
            macro avg       0.37      0.55      0.36    107313
         weighted avg       0.57      0.47      0.50    107313
```



7.4 KNN (undersampling)

[149]:
```
'''
print("\n--- K-Nearest Neighbors (KNN) (undersampling) ---")
knn_clf = KNeighborsClassifier(n_neighbors=5)
knn_clf.fit(X_train_processed, y_train_resampled)
y_pred_knn = knn_clf.predict(X_test_processed)
accuracy_knn = accuracy_score(y_test, y_pred_knn)
report_knn = classification_report(y_test, y_pred_knn,␣
 ↪target_names=[label_mapping[i] for i in sorted(label_mapping)],␣
 ↪zero_division=0)
model_performance['KNN (undersampling)'] = accuracy_knn
print(f"Accuracy: {accuracy_knn:.4f}")
print("Classification Report:\n", report_knn)

cm_knn = confusion_matrix(y_test, y_pred_knn)
disp_knn = ConfusionMatrixDisplay(confusion_matrix=cm_knn,␣
 ↪display_labels=[label_mapping[i] for i in sorted(label_mapping)])
disp_knn.plot(cmap=plt.cm.PuBuGn, xticks_rotation=45)
plt.title("KNN - Confusion Matrix")
plt.tight_layout()
plt.show()
'''
```

[149]:
```
'\nprint("\n--- K-Nearest Neighbors (KNN) (undersampling) ---")\nknn_clf =
KNeighborsClassifier(n_neighbors=5)\nknn_clf.fit(X_train_processed,
y_train_resampled)\ny_pred_knn = knn_clf.predict(X_test_processed)\naccuracy_knn
= accuracy_score(y_test, y_pred_knn)\nreport_knn = classification_report(y_test,
y_pred_knn, target_names=[label_mapping[i] for i in sorted(label_mapping)],
zero_division=0)\nmodel_performance[\'KNN (undersampling)\'] =
accuracy_knn\nprint(f"Accuracy: {accuracy_knn:.4f}")\nprint("Classification
Report:\n", report_knn)\n\ncm_knn = confusion_matrix(y_test,
y_pred_knn)\ndisp_knn = ConfusionMatrixDisplay(confusion_matrix=cm_knn,
display_labels=[label_mapping[i] for i in
sorted(label_mapping)])\ndisp_knn.plot(cmap=plt.cm.PuBuGn,
xticks_rotation=45)\nplt.title("KNN - Confusion
Matrix")\nplt.tight_layout()\nplt.show()\n'
```

8 Oversampling (SMOTE)

[164]:
```
categorical_transformer = OneHotEncoder(handle_unknown='ignore')

preprocessor = ColumnTransformer(
    transformers=[
```

```
        ('cat', categorical_transformer, categorical_features)
    ]
)

if preprocessor == 'passthrough':
    print("Preprocessor is 'passthrough'. Converting X_train_resampled/X_test␣
 ↪to numpy arrays if they are DataFrames.")
    X_train_processed = X_train.values if hasattr(X_train, 'values') else␣
 ↪X_train_SMOTE
    X_test_processed = X_test.values if hasattr(X_test, 'values') else X_test
else:
    X_train_processed = preprocessor.fit_transform(X_train)
    X_test_processed = preprocessor.transform(X_test)

sm = SMOTE(random_state= RANDOM_STATE)
X_train_SMOTE, y_train_SMOTE = sm.fit_resample(X_train_processed, y_train)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[164], line 27
     24 from imblearn.over_sampling import SMOTEN
     26 smoten = SMOTEN(random_state=RANDOM_STATE)
---> 27 X_train_SMOTEN, y_train_SMOTEN = smoten.fit_resample(X_train_encoded_df␣
 ↪y_train)

File /opt/anaconda3/lib/python3.12/site-packages/imblearn/base.py:208, in␣
 ↪BaseSampler.fit_resample(self, X, y)
    187 """Resample the dataset.
    188
    189 Parameters
   (…)
    205     The corresponding label of `X_resampled`.
    206 """
    207 self._validate_params()
--> 208 return super().fit_resample(X, y)

File /opt/anaconda3/lib/python3.12/site-packages/imblearn/base.py:112, in␣
 ↪SamplerMixin.fit_resample(self, X, y)
    106 X, y, binarize_y = self._check_X_y(X, y)
    108 self.sampling_strategy_ = check_sampling_strategy(
    109     self.sampling_strategy, y, self._sampling_type
    110 )
--> 112 output = self._fit_resample(X, y)
    114 y_ = (
    115     label_binarize(output[1], classes=np.unique(y)) if binarize_y else␣
 ↪output[1]
    116 )
```

```
    118 X_, y_ = arrays_transformer.transform(output[0], y_)

File /opt/anaconda3/lib/python3.12/site-packages/imblearn/over_sampling/_smote/
 ↪base.py:1031, in SMOTEN._fit_resample(self, X, y)
   1028 target_class_indices = np.flatnonzero(y == class_sample)
   1029 X_class = _safe_indexing(X_encoded, target_class_indices)
-> 1031 X_class_dist = vdm.pairwise(X_class)
   1032 self.nn_k_.fit(X_class_dist)
   1033 # the kneigbors search will include the sample itself which is
   1034 # expected from the original algorithm

File /opt/anaconda3/lib/python3.12/site-packages/imblearn/metrics/pairwise.py:
 ↪227, in ValueDifferenceMetric.pairwise(self, X, Y)
    224     else:
    225         proba_feature_Y = proba_feature_X
    226     distance += (
--> 227         distance_matrix(proba_feature_X, proba_feature_Y, p=self.k) **␣
 ↪self.r
    228     )
    229 return distance

File /opt/anaconda3/lib/python3.12/site-packages/scipy/spatial/_kdtree.py:919,␣
 ↪in distance_matrix(x, y, p, threshold)
    917 else:
    918     for j in range(n):
--> 919         result[:,j] = minkowski_distance(x,y[j],p)
    920 return result

KeyboardInterrupt:
```

[ ]:
```
print(f"X_train_resampled shape: {X_train_SMOTE.shape}, X_test shape: {X_test.
 ↪shape}")
print(f"y_train_resampled shape: {y_train_SMOTE.shape}, y_test shape: {y_test.
 ↪shape}")
```

8.1 Logistic Regression (SMOTE)

[ ]:
```
print("\n--- Logistic Regression (SMOTE)---")
log_reg = LogisticRegression(random_state=RANDOM_STATE, max_iter=1000,␣
 ↪solver='liblinear')
log_reg.fit(X_train_processed, y_train_SMOTE)
y_pred_log_reg = log_reg.predict(X_test_processed)
accuracy_log_reg = accuracy_score(y_test, y_pred_log_reg)
report_log_reg = classification_report(y_test, y_pred_log_reg,␣
 ↪target_names=[label_mapping[i] for i in sorted(label_mapping)],␣
 ↪zero_division=0)
```

```
model_performance['Logistic Regression (SMOTE)'] = accuracy_log_reg
print(f"Accuracy: {accuracy_log_reg:.4f}")
print("Classification Report:\n", report_log_reg)

cm_log_reg = confusion_matrix(y_test, y_pred_log_reg)
disp_log_reg = ConfusionMatrixDisplay(confusion_matrix=cm_log_reg,␣
 ↪display_labels=[label_mapping[i] for i in sorted(label_mapping)])
disp_log_reg.plot(cmap=plt.cm.Purples, xticks_rotation=45)
plt.title("Logistic Regression - Confusion Matrix")
plt.tight_layout()
plt.show()
```

8.2 XGBoost (SMOTE)

[ ]:
```
print("\n--- XGBoost (SMOTE)---")
# y_train_resampled should be 0-indexed for XGBoost, which y_processed should␣
 ↪be.
xgb_clf = xgb.XGBClassifier(random_state=RANDOM_STATE, use_label_encoder=False,␣
 ↪eval_metric='mlogloss')
xgb_clf.fit(X_train_processed, y_train_SMOTE)
y_pred_xgb = xgb_clf.predict(X_test_processed)
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
report_xgb = classification_report(y_test, y_pred_xgb,␣
 ↪target_names=[label_mapping[i] for i in sorted(label_mapping)],␣
 ↪zero_division=0)
model_performance['XGBoost (SMOTE)'] = accuracy_xgb
print(f"Accuracy: {accuracy_xgb:.4f}")
print("Classification Report:\n", report_xgb)

cm_xgb = confusion_matrix(y_test, y_pred_xgb)
disp_xgb = ConfusionMatrixDisplay(confusion_matrix=cm_xgb,␣
 ↪display_labels=[label_mapping[i] for i in sorted(label_mapping)])
disp_xgb.plot(cmap=plt.cm.Blues, xticks_rotation=45)
plt.title("XGBoost - Confusion Matrix")
plt.tight_layout()
plt.show()
```

8.3 Random Forest (SMOTE)

[ ]:
```
print("\n--- Random Forest (SMOTE)---")
rf_clf = RandomForestClassifier(random_state=RANDOM_STATE, n_estimators=100,␣
 ↪n_jobs=-1)
rf_clf.fit(X_train_processed, y_train_SMOTE)
y_pred_rf = rf_clf.predict(X_test_processed)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
```

```
report_rf = classification_report(y_test, y_pred_rf,␣
 ↪target_names=[label_mapping[i] for i in sorted(label_mapping)],␣
 ↪zero_division=0)
model_performance['Random Forest (SMOTE)'] = accuracy_rf
print(f"Accuracy: {accuracy_rf:.4f}")
print("Classification Report:\n", report_rf)

cm_rf = confusion_matrix(y_test, y_pred_rf)
disp_rf = ConfusionMatrixDisplay(confusion_matrix=cm_rf,␣
 ↪display_labels=[label_mapping[i] for i in sorted(label_mapping)])
disp_rf.plot(cmap=plt.cm.Greens, xticks_rotation=45)
plt.title("Random Forest - Confusion Matrix")
plt.tight_layout()
plt.show()
```

8.4 KNN (SMOTE)

[ ]:
```
'''
print("\n--- K-Nearest Neighbors (KNN) (SMOTE)---")
knn_clf = KNeighborsClassifier(n_neighbors=5)
knn_clf.fit(X_train_processed, y_train_SMOTE)
y_pred_knn = knn_clf.predict(X_test_processed)
accuracy_knn = accuracy_score(y_test, y_pred_knn)
report_knn = classification_report(y_test, y_pred_knn,␣
 ↪target_names=[label_mapping[i] for i in sorted(label_mapping)],␣
 ↪zero_division=0)
model_performance['KNN (SMOTE)'] = accuracy_knn
print(f"Accuracy: {accuracy_knn:.4f}")
print("Classification Report:\n", report_knn)

cm_knn = confusion_matrix(y_test, y_pred_knn)
disp_knn = ConfusionMatrixDisplay(confusion_matrix=cm_knn,␣
 ↪display_labels=[label_mapping[i] for i in sorted(label_mapping)])
disp_knn.plot(cmap=plt.cm.PuBuGn, xticks_rotation=45)
plt.title("KNN - Confusion Matrix")
plt.tight_layout()
plt.show()
'''
```

9 CV for Sampling

[ ]:
```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer, recall_score, f1_score
from imblearn.pipeline import Pipeline as ImbPipeline
from xgboost import XGBClassifier
import warnings
```

```
warnings.filterwarnings("ignore", message="Found unknown categories.*",␣
 ↪category=UserWarning)
RANDOM_STATE = 42

cat_cols = X_train.select_dtypes(include='object').columns
num_cols = X_train.select_dtypes(include='number').columns

preprocessor = ColumnTransformer(transformers=[
    ('num', StandardScaler(), num_cols),
    ('cat', OneHotEncoder(drop='first', handle_unknown='ignore'), cat_cols)
])

clf = RandomForestClassifier(random_state=RANDOM_STATE)

recall_macro = make_scorer(recall_score, average='macro')
f1_macro = make_scorer(f1_score, average='macro')
accuracy = make_scorer(accuracy_score)

preprocessor.fit(X_train)

X_small = X_train.sample(n=15000, random_state=RANDOM_STATE)
y_small = y_train.loc[X_small.index]
```

[ ]:
```
pipe_base = Pipeline([
    ('preprocess', preprocessor),
    ('clf', clf)
])
pipe_weighted = Pipeline([
    ('preprocess', preprocessor),
    ('clf', RandomForestClassifier(n_estimators=30, class_weight='balanced',␣
 ↪random_state=RANDOM_STATE))
])
pipe_smote = ImbPipeline([
    ('preprocess', preprocessor),
    ('smote', SMOTE(random_state=RANDOM_STATE)),
    ('clf', XGBClassifier(use_label_encoder=False, eval_metric='mlogloss',␣
 ↪random_state=RANDOM_STATE))
])
pipe_undersample = ImbPipeline([
    ('preprocess', preprocessor),
    ('undersample', RandomUnderSampler(random_state=42)),
    ('clf', clf)
])
```

[ ]:
```
# substitute scoring= as needed
print("Baseline:")
```

```
print(cross_val_score(pipe_base, X_small, y_small, scoring=f1_macro, cv=3).
 ↪mean())

print("Class Weights:")
print(cross_val_score(pipe_weighted, X_small, y_small, scoring=f1_macro, cv=3).
 ↪mean())

print("SMOTE:")
print(cross_val_score(pipe_smote, X_small, y_small, scoring=f1_macro, cv=3,␣
 ↪n_jobs=-1))

print("Undersampling:")
print(cross_val_score(pipe_undersample, X_small, y_small, scoring=f1_macro,␣
 ↪cv=3).mean())
```