George Washington University

EMSE 6574 - Programming for Analytics

Final Project Report

Instructor: Michael Rossetti
Date: December 12, 2025

Team members: Vidyullatha Kavilukodige Sathishrao (GWID: G44335744), Taekwon Choi (GWID: G27747799)

Table of Contents

1. Dataset Overview
2. Data Dictionary
3. Dataset Analysis
4. Machine Learning Methods
5. Machine Learning Results
6. Reflections

1. Dataset Overview

For this project, we analyzed one hundred thousand motor vehicle crash records from New York City. These records were obtained through the NYC Open Data API and covered incidents reported between October 2024 and December 2025. Each record included detailed information, such as the time and location of the crash, the number of people injured or killed, and the types of victims involved. This made the dataset suitable for examining patterns in both crash frequency and crash severity.

Before using the data for analysis and modeling, we performed extensive data cleaning. Some records were missing latitude or longitude values. Some contributing factors were unclear or labeled as Unspecified. We had to convert the date and time into a single datetime variable and extract more useful values, such as the hour of the day, the day of the week, the month, and whether the crash occurred on a weekend. This made the data easier to study.

After processing everything, the final machine learning dataset contained 98,335 rows. The main goal was to predict whether a crash was high severity. We defined high severity as a crash with one or more fatalities or two or more injuries. As shown in Figure 1, this target variable had about 90 % low severity and about 10% high severity. Despite the imbalance, the dataset was still large and informative enough to support strong predictive modeling.

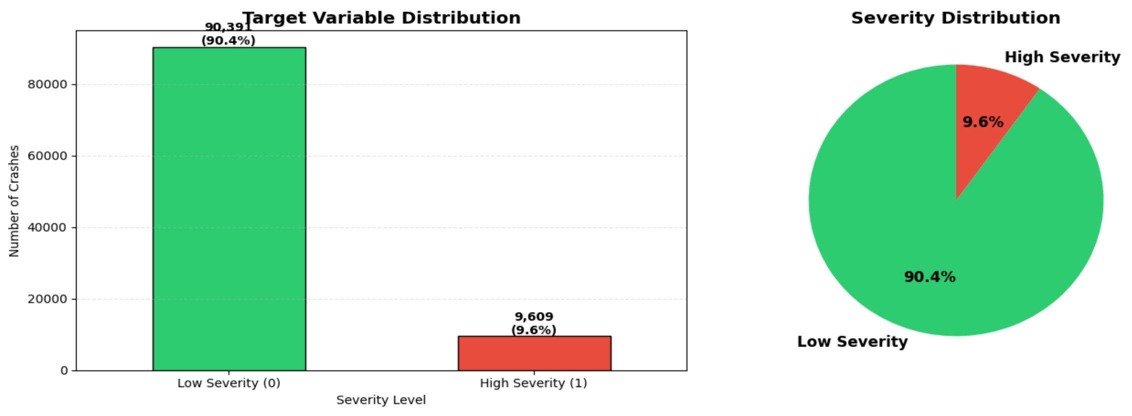


Figure 1: Target Variable and Severity Distribution

This dataset was very useful because it included a mix of raw crash information and newly engineered features. Together, these allowed us to explore crash patterns and build a model that could estimate the severity of a crash based on the available information. Since the data covered many parts of New York City and a wide time range, it felt like a realistic and complete dataset for this type of prediction task.

2. Data Dictionary

To better understand the structure of the dataset, we created a data dictionary that summarizes all the variables used in the analysis. The dataset as shown in Table 1 contained twenty-nine original variables that described each crash. These included crash_date, crash_time, borough, latitude, longitude, multiple injury and fatality counts for different victim types, contributing_factor_vehicle_1, vehicle_type_code_1, and other similar variables. These fields gave us the basic structure of each crash. We also added several new variables to capture time patterns more clearly. For example, we created crash_datetime, which combined the date and time and created hour, day_of_week, and month. We also included is_weekend to mark Saturday and Sunday and is_rush_hour to highlight typical busy traffic periods. For victims, we created simple flags such as has_pedestrian_victim, has_cyclist_victim, and has_motorist_victim. These made it easier to see whether each type of victim was involved in the crash.

Table 1: Overview of Dataset Variables and Descriptions

Variable Name	Data Type	Description
crash_date	datetime	Date when the crash occurred.
crash_time	object (HH:MM)	Time of the crash before transformation.
latitude	float	Latitude coordinate of the crash location.
longitude	float	Longitude coordinate of the crash location.
borough	category	NYC borough where the crash occurred.
number_of_persons_injured	integer	Total number of persons injured.
number_of_persons_killed	integer	Total number of persons killed.
number_of_pedestrians_injured	integer	Pedestrians injured in the crash.
number_of_pedestrians_killed	integer	Pedestrians killed in the crash.
number_of_cyclist_injured	integer	Cyclists injured in the crash.
number_of_cyclist_killed	integer	Cyclists killed in the crash.
number_of_motorist_injured	integer	Motorists injured in the crash.
number_of_motorist_killed	integer	Motorists killed in the crash.
severity_injury_flag	binary (0/1)	1 if any injury occurred, 0 otherwise.
severity_killed_flag	binary (0/1)	1 if any fatality occurred, 0 otherwise.
severity_flag	binary (0/1)	Combined indicator for injury or fatality.
high_severity_flag	binary (0/1)	Target variable; 1 for serious injury or death.
injury_count	integer	Sum of all injury-related variables.
killed_count	integer	Sum of all fatality-related variables.
contributing_factor_vehicle_1	category	Primary reported contributing factor.
contributing_factor_vehicle_2	category	Secondary contributing factor.
contributing_factor_vehicle_3	category	Third contributing factor.
contributing_factor_vehicle_4	category	Fourth contributing factor.
contributing_factor_vehicle_5	category	Fifth contributing factor.
crash_hour	integer (0–23)	Hour extracted from crash_time.

day_of_week	integer (0–6)	Day of the week (Monday = 0).
month	integer (1–12)	Month of the crash.
total_crashes	integer	Aggregated total crashes (used for borough-level EDA).
coord_missing_flag	binary (0/1)	Flags missing latitude/longitude data.

After organizing the full dataset, the next step was to choose which variables would be used for machine learning. For modeling, we selected thirteen key features in Table 2. These features were mostly the victim injury counts, the hour, the day of the week, the month, the borough, and the weekend and rush-hour indicators. We chose these features because they were the most relevant and provided enough predictive information without making the model too cluttered. Overall, the final feature set was clear and well organized, and we felt confident using it for machine learning.

Table 2: Selected Features for Machine Learning Analysis

Feature Name	Data Type	Description
injury_count	integer	Total number of injuries in the crash (sum of all injury-related fields).
killed_count	integer	Total fatalities in the crash (sum of all fatality-related fields).
severity_flag	binary (0/1)	Indicates whether the crash involved any injury or fatality.
crash_hour	integer (0–23)	Hour of day when the crash occurred.
day_of_week	integer (0–6)	Day of week of crash (Monday = 0).
month	integer (1–12)	Month of crash occurrence.
latitude	float	Latitude coordinate of the crash location.
longitude	float	Longitude coordinate of the crash location.
borough	category	Borough where the crash occurred; later one-hot encoded.
contributing_factor_vehicle_1	category	Primary contributing factor reported for the crash.
contributing_factor_vehicle_2	category	Secondary contributing factor.
contributing_factor_vehicle_3	category	Third contributing factor (if reported).
contributing_factor_vehicle_4	category	Fourth contributing factor (if reported).

Once the key features were identified, we performed exploratory data analysis to look for patterns, trends, and important relationships in the dataset. During exploratory data analysis, we wanted to understand where crashes occurred, when they occurred, and why they occurred. We also wanted to see how injuries and fatalities were distributed across different types of victims.

3. Dataset Analysis

We first looked at geographic patterns as shown in Figure 2. Brooklyn had the highest number of crashes, which made sense because it has one of the largest populations and a lot of dense traffic. Queens had the second highest count. Manhattan had fewer severe crashes, possibly because traffic moves slower there and there are more controlled intersections. One surprising result was the Unknown borough category. It had a high severity rate. This made us question whether some records were missing location information and if these missing values affected severity labeling. It is something that might need more investigation in the future.

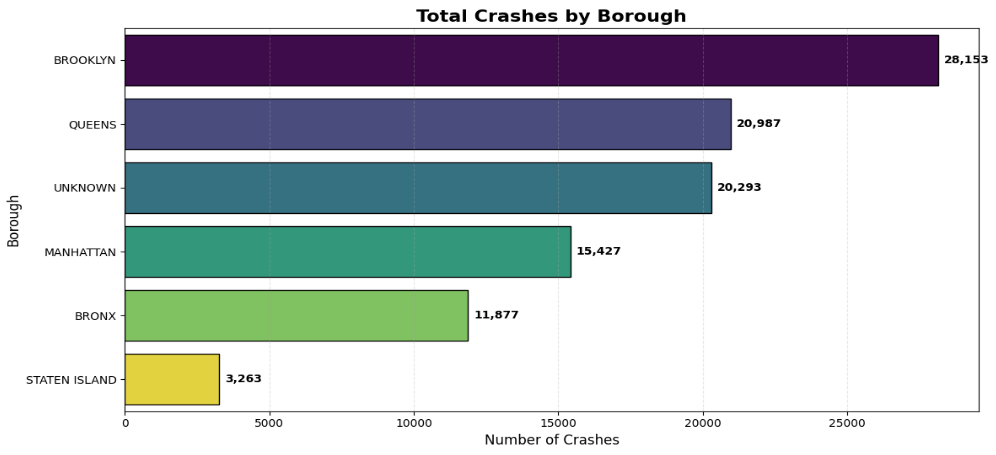


Figure 2: Total Crashes by Borough in New York

Next, we studied the time of day. As shown in Figure 3, The highest number of crashes happened around 5 in the afternoon when many people were commuting home. However, the most severe crashes happened late at night, especially around 11 at night. This pattern showed a clear difference between crash frequency and crash severity. People get into more crashes during busy hours, but those crashes are usually less severe. Severe crashes tend to happen when there are fewer cars but more risky situations, such as tired drivers or impaired driving.

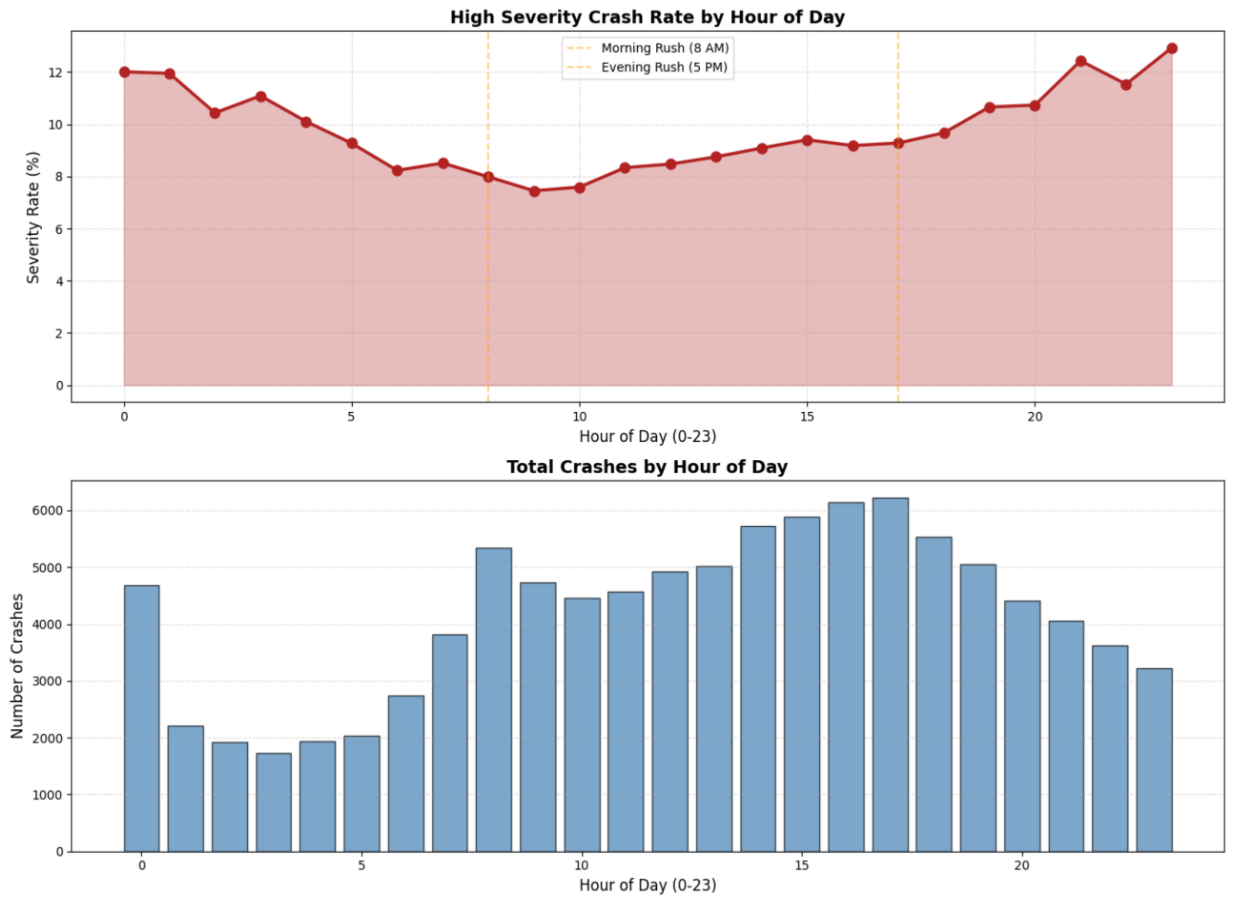


Figure 3: High Severity Crash Rate and Total Crashes by Hour of Day

Weekly and monthly patterns also revealed insights. As shown in Figure 4, crash counts were relatively stable across the week but increased slightly on Fridays, possibly due to late-night travel or social activities. Seasonal trends showed that November and December had elevated crash counts, which may be linked to holiday travel, harsher weather conditions, and increased traffic congestion.

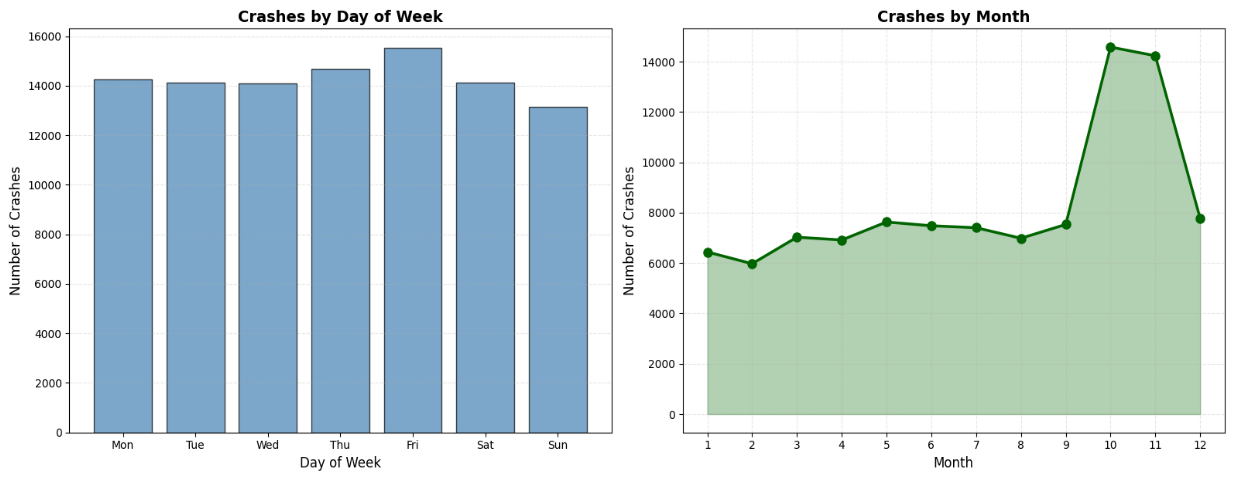


Figure 4: Number of Crashes by Day of Week and Month

Looking at contributing factors in Figure 5, Driver Inattention or Distraction and Unspecified were the most common. This does not necessarily mean they cause the most severe crashes. The factors related to high severity were Lost Consciousness, Illness, and Unsafe Speed. These factors are more urgent and dangerous, and it made sense that they led to more severe outcomes. They represent situations where the driver cannot control the vehicle or reacts too slowly.

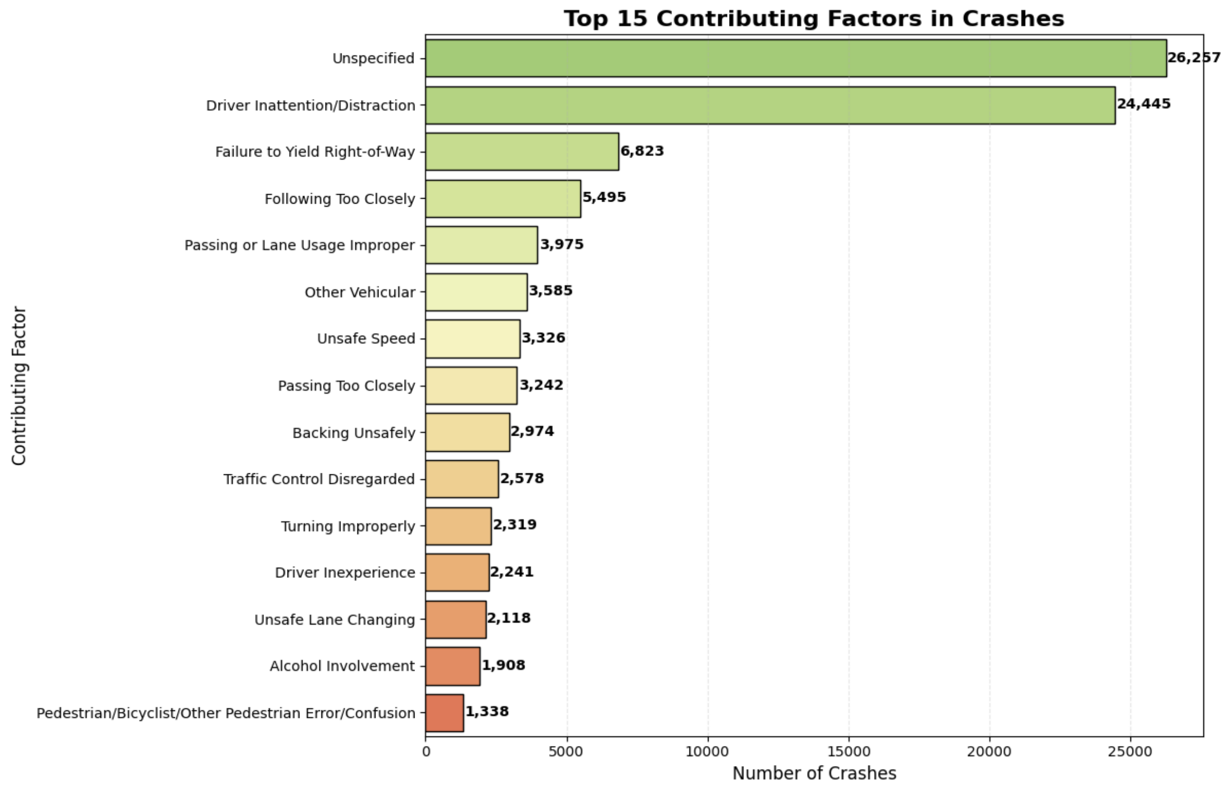


Figure 5: Top 15 Contributing Factors in Crashes

We also checked correlations as shown in Figure 6. Injury-related features were strongly correlated with the High_Severity_Flag, which was expected in Figure 7. What surprised us was how little temporal features contributed. Things like hour, day_of_week, or month did not show strong relationships with severity. This finding suggested that the severity of a crash depends more on what directly happened during the crash rather than when it happened. Overall, the exploratory analysis helped us understand the data structure and confirmed that injury counts and victim types would be important in the machine learning model.

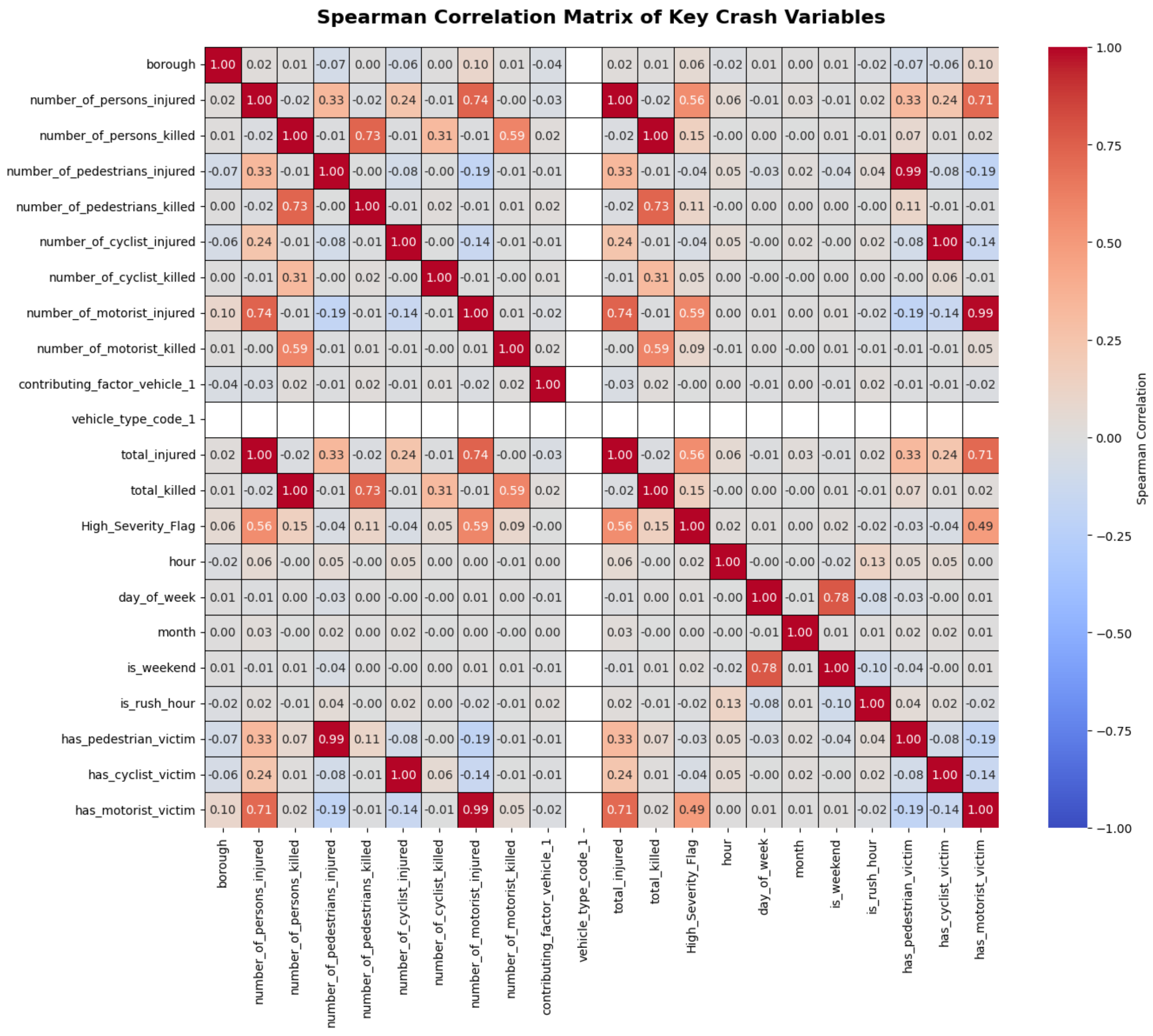
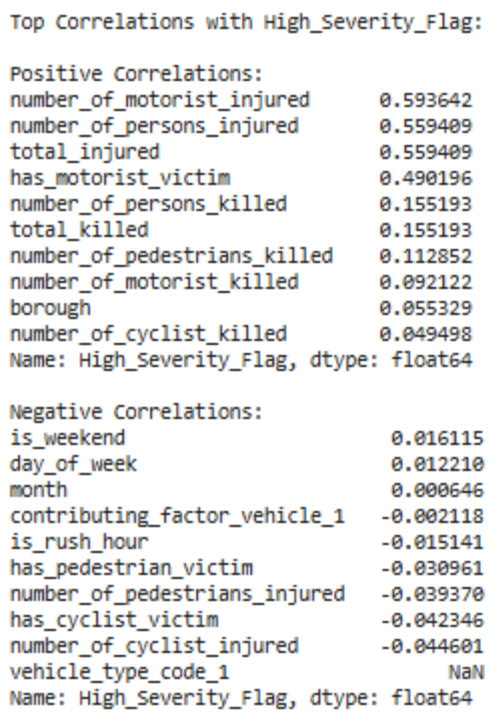


Figure 6: Correlation Matrix of Key Crash Variables



```
Top Correlations with High_Severity_Flag:

Positive Correlations:
number_of_motorist_injured      0.593642
number_of_persons_injured       0.559409
total_injured                   0.559409
has_motorist_victim             0.490196
number_of_persons_killed        0.155193
total_killed                    0.155193
number_of_pedestrians_killed    0.112852
number_of_motorist_killed       0.092122
borough                         0.055329
number_of_cyclist_killed        0.049498
Name: High_Severity_Flag, dtype: float64

Negative Correlations:
is_weekend                       0.016115
day_of_week                      0.012210
month                            0.000646
contributing_factor_vehicle_1   -0.002118
is_rush_hour                    -0.015141
has_pedestrian_victim           -0.030961
number_of_pedestrians_injured   -0.039370
has_cyclist_victim              -0.042346
number_of_cyclist_injured       -0.044601
vehicle_type_code_1                   NaN
Name: High_Severity_Flag, dtype: float64
```

Figure 7: Summary of the Correlation Matrix

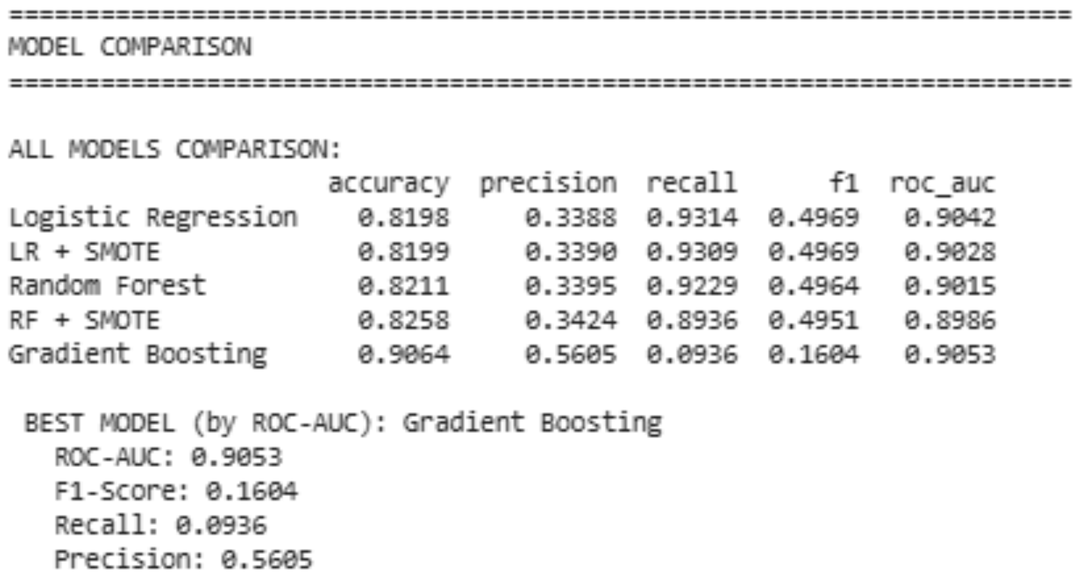
4. Machine Learning Methods

After studying these patterns, we moved on to building predictive models to see whether crash severity could be estimated using the available features. For the machine learning part of this project, we treated the prediction task as a binary classification problem. We trained several models to compare their performance and see which one was most effective. The models included Logistic Regression, Logistic Regression with SMOTE, Random Forest, Random Forest with SMOTE, and Gradient Boosting. We chose these models because they offer different strengths. Logistic Regression is simple and interpretable. Random Forest captures non-linear relationships. Gradient Boosting focuses on combining many weaker learners to create a strong model.

We used ROC AUC as our main evaluation metric because it is reliable when dealing with imbalanced data. We also looked at precision, recall, accuracy, and the F1 score. Before training the models, we handled missing values, encoded the borough variable, scaled numerical features, and used SMOTE on the training data when needed. The training dataset contained 78,668 samples, and the test dataset contained 19,667 samples. This modeling process helped us understand the strengths and weaknesses of each algorithm. More importantly, it allowed us to evaluate how well the models captured high-severity cases, which was the main goal of the project.

5. Machine Learning Results

After training the models, we evaluated their performance using several different metrics to understand their strengths and weaknesses. Based on the comparison of all five models as shown in Figure 8, the results showed clear tradeoffs between accuracy, precision, recall, and ROC AUC. Gradient Boosting achieved the highest ROC AUC score at 0.9053, but it performed extremely poorly in recall by detecting only about nine percent of all severe crashes. This made the model unreliable for the goals of this project. Logistic Regression had one of the strongest recall scores at 0.9314, meaning it successfully identified most of the severe crashes in the dataset. Its ROC AUC of 0.9042 was only slightly lower than Gradient Boosting, and it still showed strong overall performance. Even though its precision was lower, the balance between recall and ROC AUC made Logistic Regression the more dependable choice.



```
============================================================================
MODEL COMPARISON
============================================================================

ALL MODELS COMPARISON:
                     accuracy  precision  recall      f1  roc_auc
Logistic Regression    0.8198     0.3388  0.9314  0.4969   0.9042
LR + SMOTE             0.8199     0.3390  0.9309  0.4969   0.9028
Random Forest          0.8211     0.3395  0.9229  0.4964   0.9015
RF + SMOTE             0.8258     0.3424  0.8936  0.4951   0.8986
Gradient Boosting      0.9064     0.5605  0.0936  0.1604   0.9053

 BEST MODEL (by ROC-AUC): Gradient Boosting
   ROC-AUC: 0.9053
   F1-Score: 0.1604
   Recall: 0.0936
   Precision: 0.5605
```

Figure 8: Results from All Models Comparison

The ROC AUC comparison graph in Figure 9 also supported this conclusion. All five models performed at a high level and scored well above the random classifier line. This showed that each model could separate high severity and low severity crashes better than chance. The ROC curves for all models were also very close to one another, which meant that ROC AUC alone was not enough to pick the final model. Since the differences were small, the project needed a different metric to guide model selection. Because the main goal was to identify as many severe crashes as possible, recall became the most important measure. Logistic Regression provided the strongest recall while also keeping a competitive ROC AUC, so it offered the most balanced performance for this study.

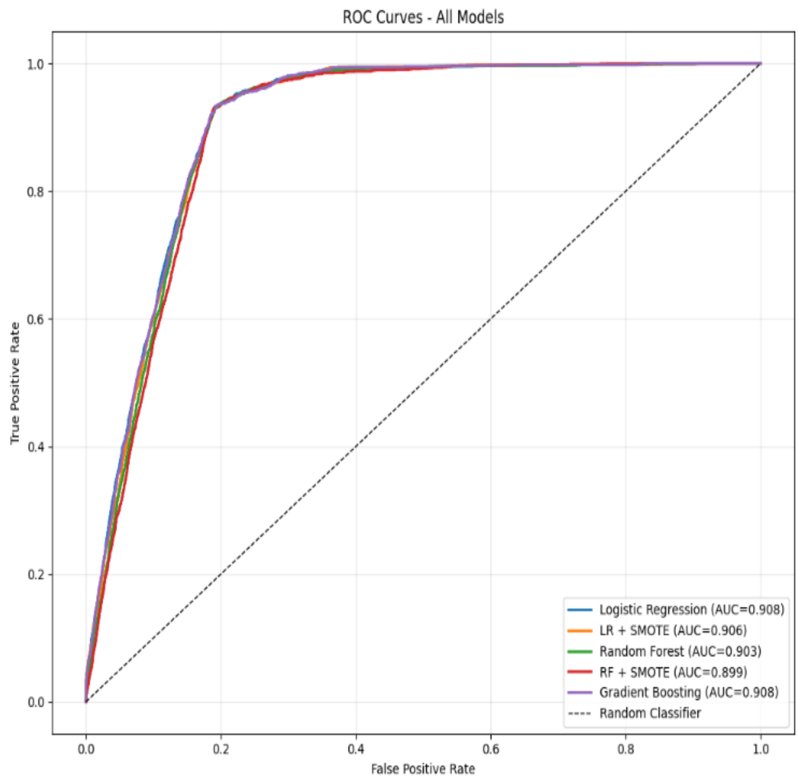


Figure 9: ROC AUC All Five Models Comparison Graph

Figure 10 shows a direct comparison of the ROC AUC scores for all five models using a bar chart. This figure makes it easy to see that every model performed well above the random classifier line, which confirms that all of them were able to separate high severity and low severity crashes better than chance. Although Gradient Boosting had the highest ROC AUC score, the difference between the models was very small. Because the scores were so close to each other, the bar chart supported the idea that ROC AUC alone was not enough to decide the best model for this project. This is why recall became the main metric for choosing the final model.

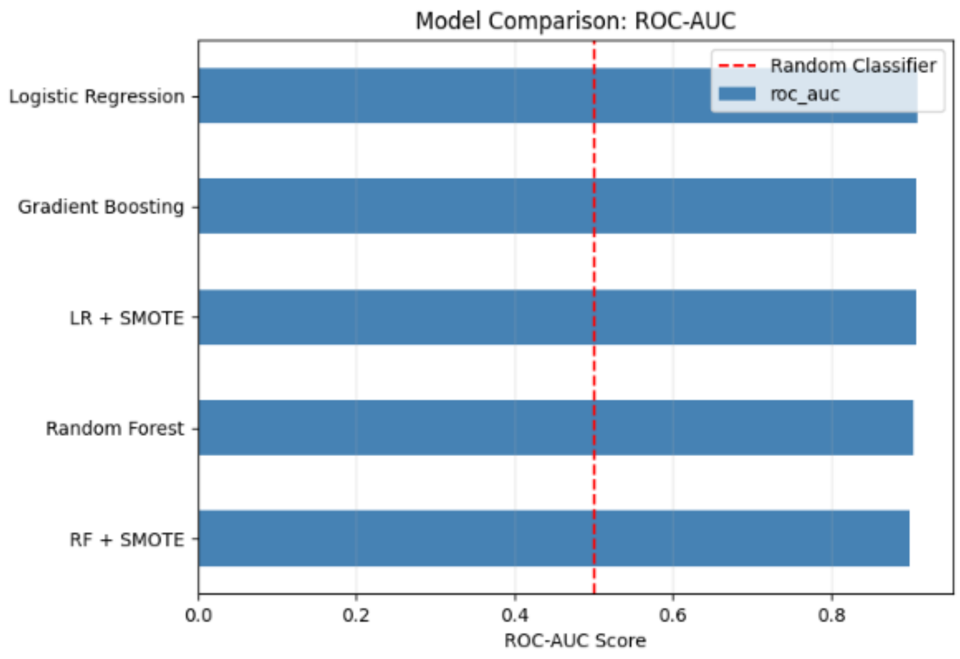


Figure 10: Comparison of the ROC-AUC score for all five models

Figure 11 presents the comparison of F1 scores across the models. Most models had similar F1 scores except for Gradient Boosting, which had a very low F1 score due to its extremely poor recall. This again confirmed that Gradient Boosting was not suitable even though it had the highest accuracy and precision. The F1 score is useful in imbalanced datasets because it combines both precision and recall into a single measure. Logistic Regression and the other models remained close in F1 score, which supported the idea that Logistic Regression was still performing well overall.

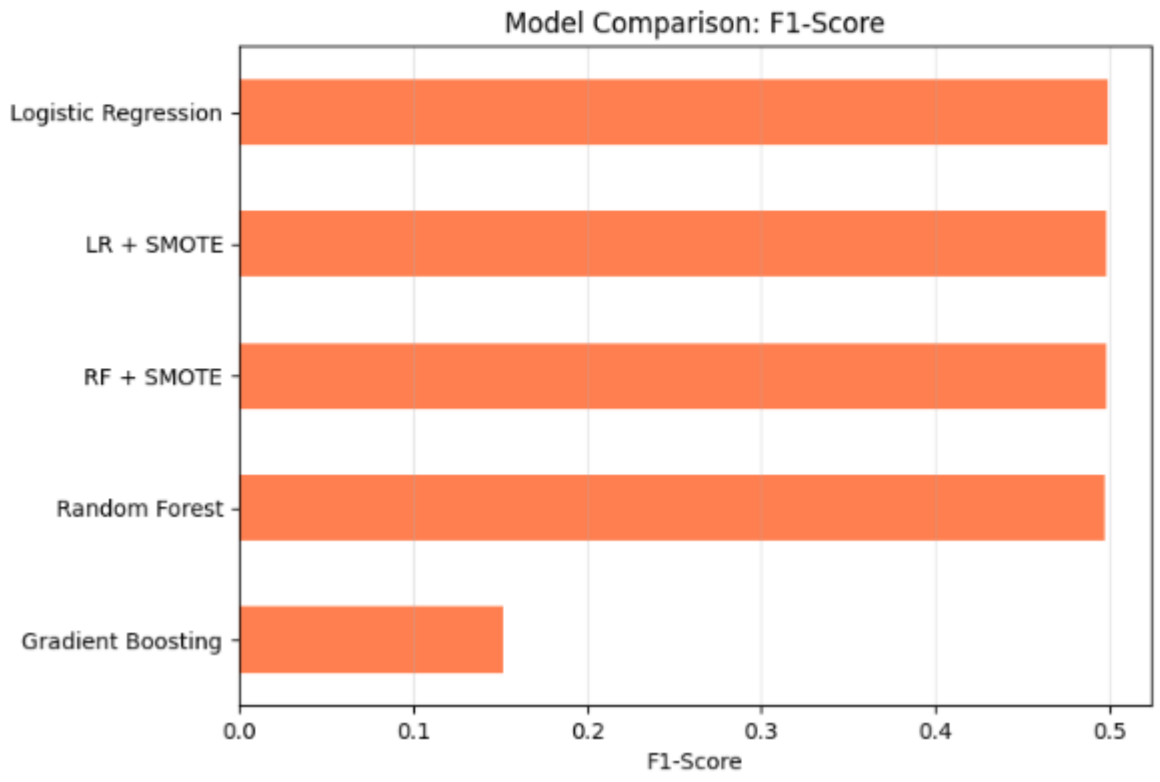


Figure 11: Model Comparison with F1-Score

Figure 12 shows the precision and recall trade off across different models. The plot makes it even clearer that Gradient Boosting had very high precision but extremely low recall. This means it predicted severe crashes only when it was very confident, but it missed almost all severe crashes. On the other hand, Logistic Regression and the SMOTE based models had strong recall, which matched the project goal. In a safety related problem, missing severe crashes is more harmful than predicting some normal crashes as severe. Because of this, a model with high recall was more valuable, even if its precision was lower.

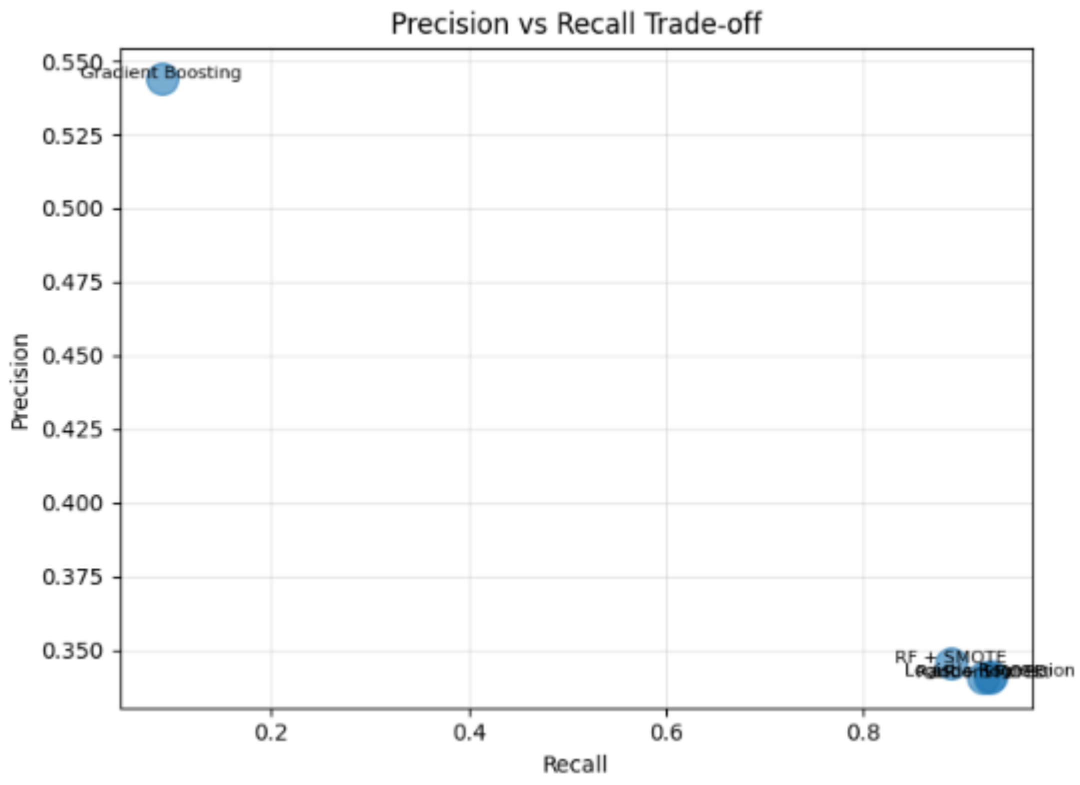


Figure 12: Precision and recall trade-off across different models

The confusion matrix for Logistic Regression in Figure 13 gave a more detailed understanding of how the model behaved. Logistic Regression correctly identified 1750 severe crashes and missed 132. It did label some low severity crashes as severe, but this was an acceptable tradeoff because the purpose of the project was to avoid missing severe crashes. The confusion matrix showed that the model was reliable in identifying dangerous events and performed well on both classes given the imbalance in the dataset.

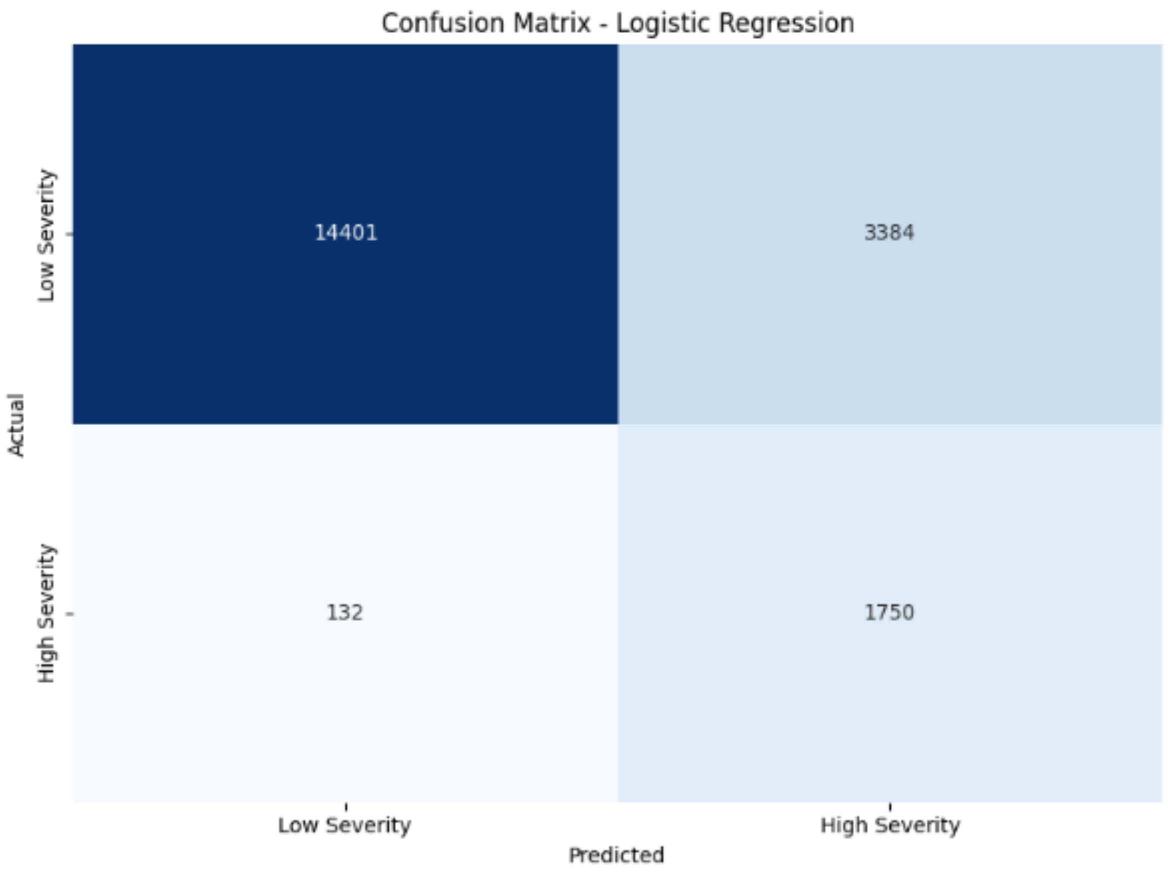


Figure 13: Confusion Matrix for Logistic Regression

Figure 14, which shows the heatmap of all metrics, provided a helpful summary of how each model performed across accuracy, precision, recall, F1 score, and ROC AUC. This figure made it easy to see that Gradient Boosting only stood out in accuracy and precision, but it failed in recall. Logistic Regression showed the strongest recall and one of the best ROC AUC scores, while still maintaining a balanced performance in the other metrics. This heatmap supported the final decision to choose Logistic Regression.

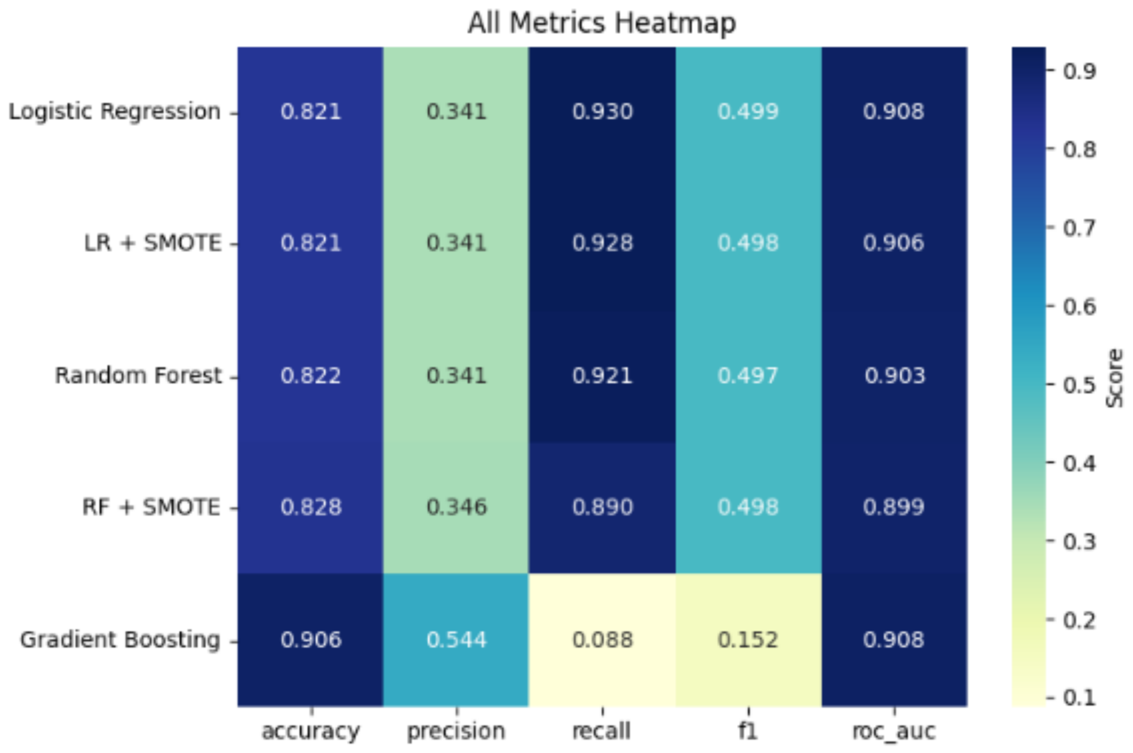


Figure 14: Heatmap of all metrics showing a summary of each model performance

6. Reflections

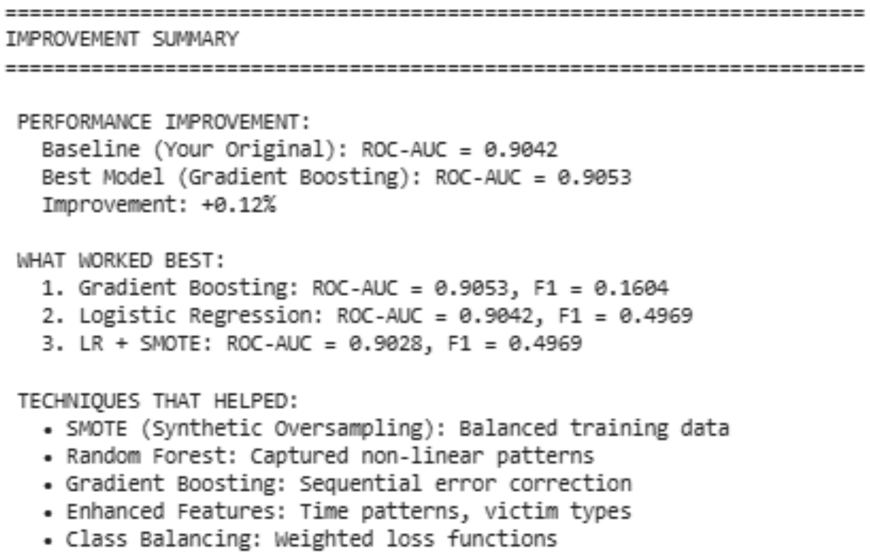
Overall, Logistic Regression stood out as the most balanced and practical model for this project. It was simple, easy to interpret, and performed reliably across the evaluation metrics that mattered most. The strong recall demonstrated that the model was effective in identifying critical crash events, and the stable ROC AUC score showed that it had good predictive separation between high and low severity crashes. This made it an appropriate choice for real world applications where detecting severe crashes early can support emergency response and reduce risks.

Although the final model performed well, there are still opportunities for improvement in future work. Some possible next steps include adjusting the prediction threshold to improve recall even further, testing class weighted versions of Gradient Boosting or XGBoost, and exploring more advanced resampling strategies. Additional feature engineering such as adding weather conditions, road types, and traffic volume data could also help improve prediction accuracy. These upgrades could help refine the model and provide stronger support for public safety decision making.

With the final model selected, we also reflected on the overall process and considered possible improvements for future work. Throughout this project, we learned how important the overall process is when working with real-world data. One of the main insights was that model performance depends heavily on the quality of data preparation. Cleaning the dataset, handling missing values, and creating meaningful features had a noticeable impact on the results. We also realized how important it is to choose evaluation metrics that match the goals of the problem. Since severe crashes were the minority class, accuracy alone was not useful, and we needed to focus more on recall and ROC AUC to understand how well each model identified high severity cases. This experience helped us appreciate that machine learning is not just about running algorithms but also about making careful decisions at every step.

Another part of the process that stood out to us was comparing different models and understanding their strengths and weaknesses. Some models looked strong at first because they had high accuracy or precision, but they performed poorly when detecting severe crashes. Logistic Regression ended up being the most balanced model because it achieved the highest recall while still keeping a competitive ROC AUC score. This showed us the importance of selecting a model based on the problem's real world needs rather than choosing the most complex or high performing model on paper.

We also noticed several techniques that played an important role in improving model performance as shown in Figure 15. Gradient Boosting produced the highest overall ROC AUC score at 0.9053, although its recall was too low for our main goal. Logistic Regression remained more reliable because it offered the best balance between recall and overall predictive ability. The combination of Logistic Regression with SMOTE also performed well and showed that balancing the training data helped the model recognize more severe crash cases. Techniques such as synthetic oversampling, non linear modeling from Random Forest, and the use of time based and victim based engineered features all contributed to the improvements we observed. Even though the overall ROC AUC improvement was modest, these steps showed us how important it is to experiment with different approaches and understand which methods actually help the model focus on the minority class.



```
======================================================================
IMPROVEMENT SUMMARY
======================================================================

 PERFORMANCE IMPROVEMENT:
   Baseline (Your Original): ROC-AUC = 0.9042
   Best Model (Gradient Boosting): ROC-AUC = 0.9053
   Improvement: +0.12%

 WHAT WORKED BEST:
   1. Gradient Boosting: ROC-AUC = 0.9053, F1 = 0.1604
   2. Logistic Regression: ROC-AUC = 0.9042, F1 = 0.4969
   3. LR + SMOTE: ROC-AUC = 0.9028, F1 = 0.4969

 TECHNIQUES THAT HELPED:
   • SMOTE (Synthetic Oversampling): Balanced training data
   • Random Forest: Captured non-linear patterns
   • Gradient Boosting: Sequential error correction
   • Enhanced Features: Time patterns, victim types
   • Class Balancing: Weighted loss functions
```

Figure 15: Improvement Summary from the Model Training

For future work, there are several directions that could help improve the analysis. One next step would be to explore more advanced feature engineering, such as adding weather information, road type, and traffic conditions, which might provide additional predictive power. Another step would be to try class weighted models or tune the prediction threshold to push recall even higher. We could also explore other resampling techniques or algorithms like XGBoost to see whether they can improve performance on the minority class. These improvements could strengthen the model’s ability to detect severe crashes and support better decision making in public safety settings.