Predicting Accident Severity:
A Data-Driven Approach

Haider Daresalamwala, Lokesh K B, Manali Shetye, Ruchika Godugu

December 2024

1 Introduction

Traffic accidents remain a pressing public safety issue, resulting in significant human and economic costs worldwide. Understanding the factors that contribute to the severity of accidents is essential for designing effective safety measures and improving road infrastructure.

In this study, we analyze traffic accident data from Montgomery County, Baltimore, to explore patterns and relationships between factors such as weather, road conditions, and time of occurrence. To enhance our analysis, predictive models are employed to estimate accident severity, providing insights that complement traditional data analysis techniques. These models help identify complex patterns and inform strategies aimed at mitigating traffic-related risks.

While the findings are rooted in local data, the methodologies and insights presented in this research are broadly applicable. By combining analysis and predictive modeling, this study contributes to ongoing efforts to improve road safety and reduce the impact of traffic accidents.

2 Problem Statement

Traffic accidents in Montgomery County, Baltimore, frequently result in injuries and fatalities, highlighting the need for better road safety measures. Despite the availability of extensive traffic data, traditional methods of analyzing accident data often fail to uncover the complex relationships between contributing factors, such as weather, road conditions, and time of day.

This gap in understanding limits the ability of policymakers and planners to implement targeted interventions. There is a critical need for a data-driven approach to accurately predict accident severity and provide actionable insights to enhance road safety.

This study addresses this gap by leveraging historical traffic data to identify patterns, build predictive models, and generate insights that can guide strategic planning and safety improvements.

3 Data and Methodologies

3.1 Dataset overview

3.1.1 Source

The dataset used in this study provides detailed information on motor vehicle operators (drivers) involved in traffic collisions on county and local roadways in Montgomery County, Baltimore. The data is collected through the Automated Crash Reporting System (ACRS) by the Maryland State Police and is contributed by agencies such as:

- Montgomery County Police
- Gaithersburg Police
- Rockville Police
- Maryland-National Capital Park Police

3.1.2 Scope of the Dataset

The dataset includes details about all reported traffic collisions occurring within Montgomery County, covering:

- Collision circumstances (e.g., location, time, weather).
- Information about the drivers involved.
- Other contributing factors to the accidents.

3.1.3 Purpose of the Dataset

This dataset forms the basis for analyzing accident trends and developing predictive models to estimate accident severity.

3.2 Exploratory Data Analysis

Our Analysis focuses on analyzing car crash data to understand key factors affecting injury severity. This report provides insights into patterns and trends in the dataset, helping us identify crucial variables for predicting injury severity.

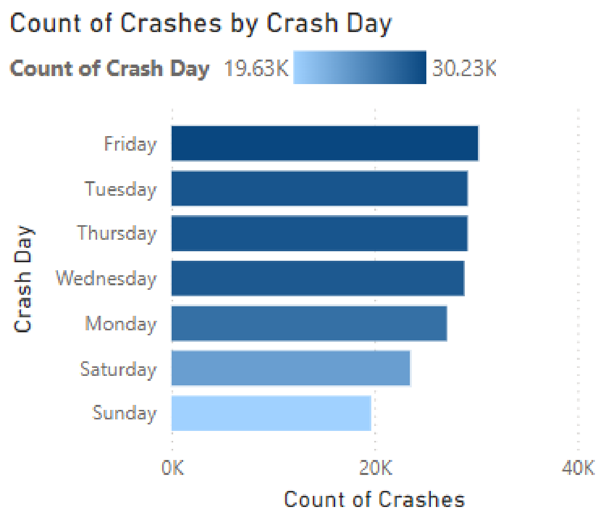


Figure 1: Count of Crashes by day

This bar chart shows the number of crashes that occurred on different days of the week. Crashes are most frequent on weekdays and least on weekends. This could indicate higher traffic volume or activity on weekdays.

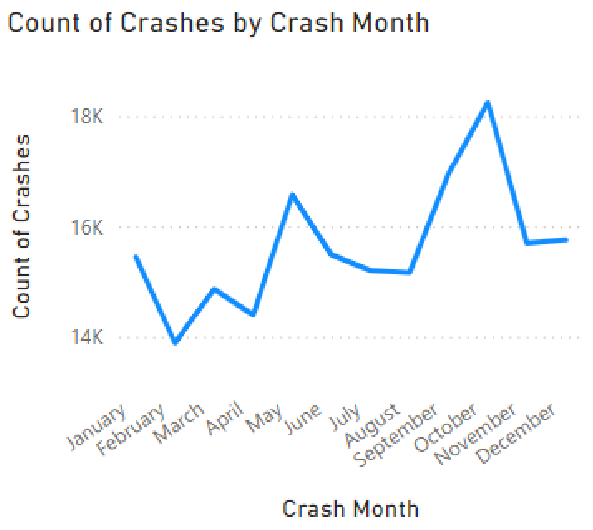


Figure 2: Count of car crashes by month

This line chart tracks the number of crashes across different months of the year. Crashes peak in October and are lowest in February. Seasonal factors like weather or holidays might contribute to this trend.

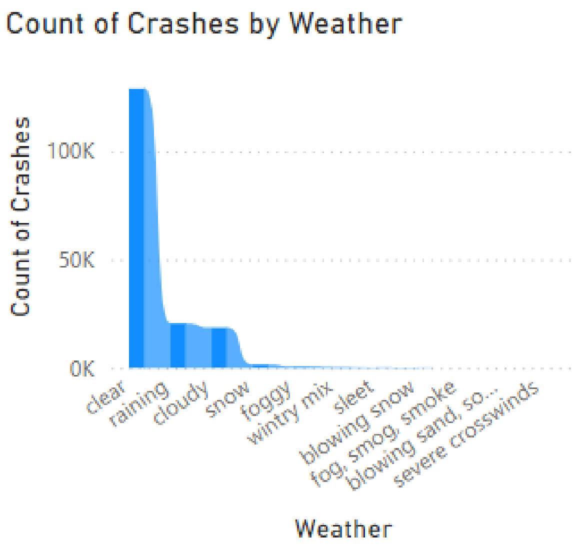


Figure 3: Crashes by weather condition

This bar chart highlights crash counts under different weather conditions. Most crashes happen in clear weather, followed by rainy conditions. This suggests that clear weather does not necessarily equate to safer driving, and other factors might be at play.

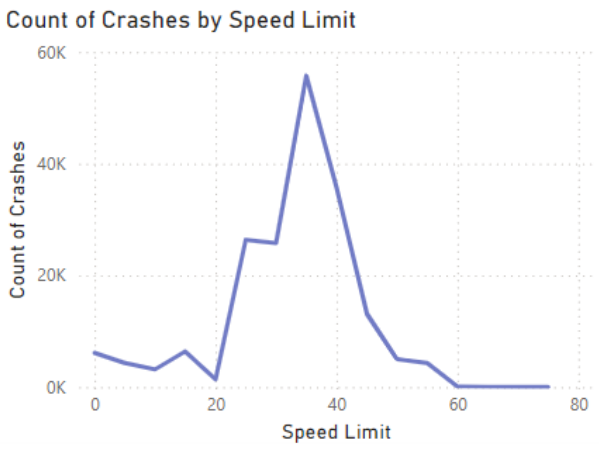


Figure 4: Crash Frequency by Speed limit

This line chart illustrates crash frequencies at different speed limits. Crashes are highest in areas with a speed limit of 40 km/h, likely due to urban traffic and intersections. Urban zones with moderate speed limits see more crashes, indicating areas where traffic safety improvements might be needed.

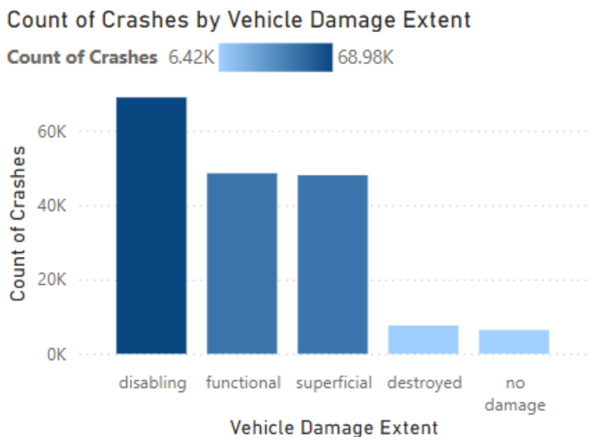
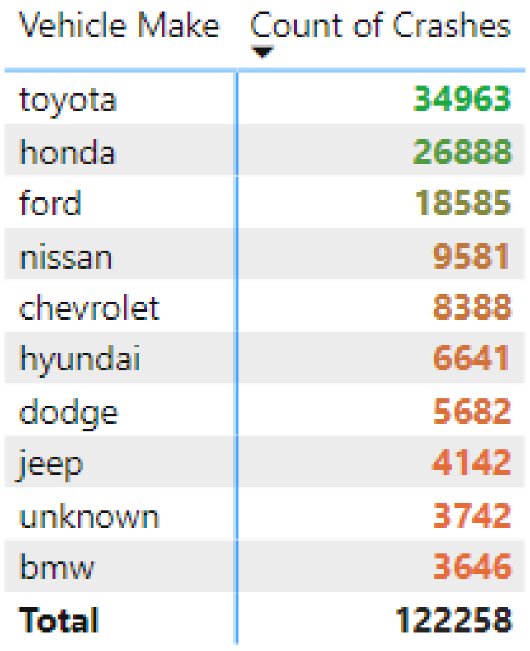


Figure 5: Enter Caption

This bar chart categorizes crashes based on the extent of vehicle damage. Disabling and functional damages dominate the crash data. A large number of crashes cause significant vehicle damage, pointing to the severity and potential impact of these incidents.



Vehicle Make	Count of Crashes
toyota	34963
honda	26888
ford	18585
nissan	9581
chevrolet	8388
hyundai	6641
dodge	5682
jeep	4142
unknown	3742
bmw	3646
Total	**122258**

Figure 6: Number of classes by vehicle make

This table ranks vehicle makes by the number of crashes. Toyota has the highest crash count, followed by Honda and Ford. Popular vehicle brands like Toyota and Honda appear more frequently in crash data, likely correlating with their market share.

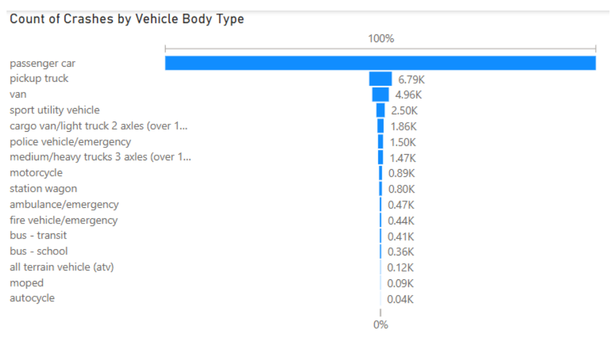


Figure 7

Passenger cars are involved in the vast majority of crashes, followed by pickup trucks and vans. Specialized vehicles like motorcycles, emergency vehicles, and buses account for a much smaller share.

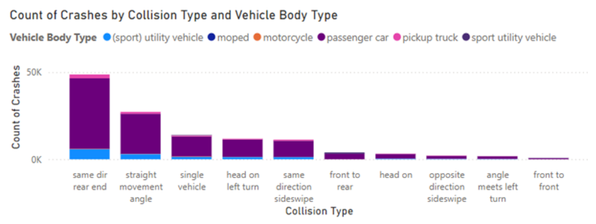


Figure 8

Rear-end collisions are the most frequent type of crash, especially involving passenger cars. Other common collision types include straight movement angles and single-vehicle crashes.

3.3 Geospatial Analysis

Geographic Distribution of Data Points in Montgomery County, Maryland:

Figure 9: MAP

The map illustrates the geographic distribution of data points within Montgomery County, Maryland. This county is situated adjacent to Washington, D.C., the nation's capital.

3.3.1 Heatmap

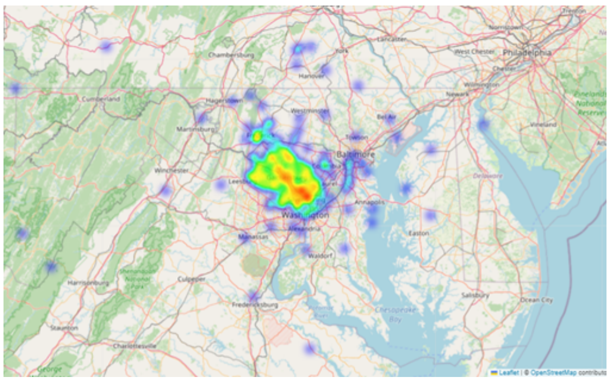


Figure 10: Heatmap of traffic accidents

The heatmap illustrates crash density, with hotspots (Red/Yellow) around urban areas suggesting a higher risk.

Potential causes for high-risk areas:
This may be due to high traffic volumes, poor road signals, or speed-related issues

Mitigation plan:
The policymakers can plan targeted interventions like better traffic controls, emergency preparedness, infrastructure improvement in critical issues

3.3.2 Clusters

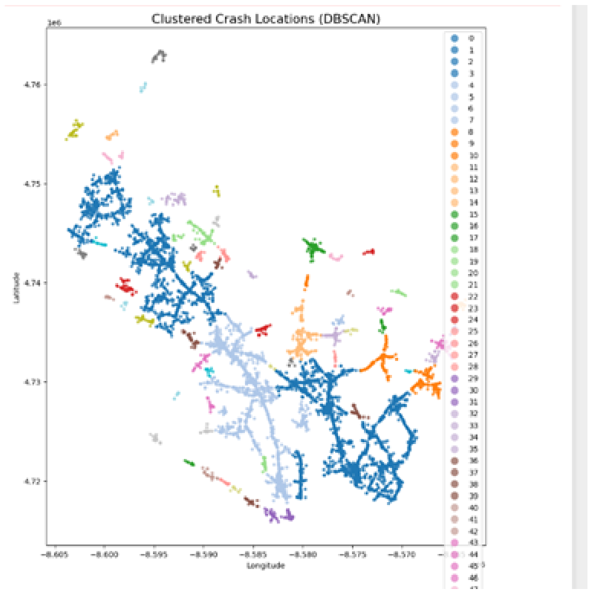


Figure 11: Clustering using DBSCAN

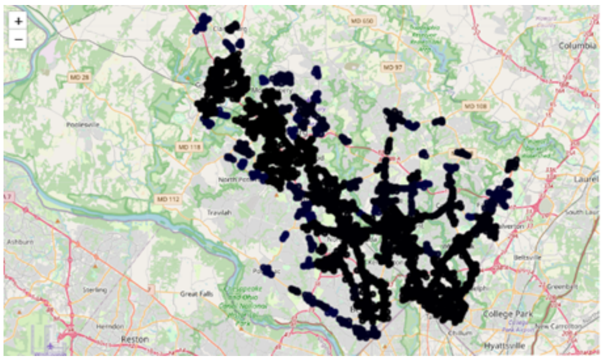


Figure 12: Cluster on Map

The map highlights crash clusters identifying high-risk zones(large clusters) & localized issues(small clusters). It excludes noise points to focus on patterns

Dense/Large clusters: Indicate areas needing improved traffic flow management, better signals, or restructured intersections.

Small clusters: Requires targeted interventions for local issues such as pedestrian crossings, speed controls, or road designs.

3.3.3 Dense cluster areas

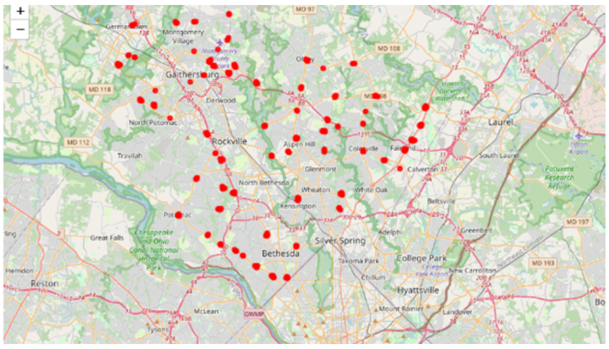


Figure 13: Dense Cluster Areas

This map shows specific dense cluster areas like Rockville, Bethesda, and Aspen Hill, which indicate high-risk zones that require targeted safety interventions.

3.4 Data Pre-processing

The dataset consists of a total of 187,311 observations and 39 features.To ensure the quality and relevance of the data for modeling, features with more than 80% missing values and those deemed not useful for predictive modeling were removed. The following features were dropped during preprocessing:

- Circumstance
- Local Case Number
- Agency Name
- Driver's License State
- Vehicle ID
- Report Number
- Person ID
- Crash Date/Time

This preprocessing step refined the dataset, retaining only the most informative and complete features for analysis and modeling.

3.5 Feature Engineering

To enhance the quality and usability of the dataset, categorical variables were standardized and grouped to ensure consistency and relevance. The following features were processed during this step:

3.5.1 Injury Severity

The ‘Injury Severity‘ feature was standardized by consolidating similar categories into two primary groups:

- **Injury:** Includes categories indicating the presence of injury:
  - Possible Injury
  - Suspected Minor Injury
  - Suspected Serious Injury
  - Fatal Injury
- **No Injury:** Includes categories indicating the absence of injury:
  - No Apparent Injury

3.5.2 Surface Conditions

The ‘Surface Conditions‘ feature was grouped into three main categories based on road surface characteristics:

- **Dry:** Includes conditions such as:
  - Dry, Mud, Dirt, Gravel, Sand
- **Wet:** Includes conditions indicating moisture or ice:
  - Wet, Ice, Snow, Slush, Standing or Moving Water, Oil
- **Other:** Includes categories with unclear or unspecified conditions:
  - Unknown, Other

3.5.3 Time Transformation

The ‘Time‘ feature contained inconsistent and descriptive formats such as ”Twelve O Clock” or ”Six O Clock,” which were standardized for consistency and easier interpretation. The transformation mapped these descriptive labels to standard ‘AM‘ and ‘PM‘ time periods. The following changes were applied:

- Descriptive times corresponding to morning hours (e.g., ”TWELVE OCLOCK,” ”SIX OCLOCK”) were mapped to **AM**.
- Descriptive times corresponding to afternoon and evening hours (e.g., ”Twelve O Clock,” ”Six O Clock”) were mapped to **PM**.

This standardization ensured uniformity in the dataset and enabled the effective analysis and modeling of time-related patterns. Below is the mapping logic used during preprocessing:

```
"TWELVE OCLOCK" -> "AM"
"SIX OCLOCK" -> "AM"
"ONE OCLOCK" -> "AM"
...
"Twelve O Clock" -> "PM"
"Six O Clock" -> "PM"
"Eight O Clock" -> "PM"
```

3.6 Label Encoding and Missing Data Imputation

To prepare the dataset for modeling, the following preprocessing steps were applied:

- **Label Encoding**: Categorical variables in the dataset were converted into numerical representations using label encoding. This ensured compatibility with machine learning models while retaining the categorical information.
- **Missing Data Imputation**: Median imputation was applied to address any remaining missing values in the dataset. By replacing missing values with the median of each respective feature, we minimized the impact of outliers while preserving the central tendency of the data.

These steps ensured the dataset was complete and appropriately formatted for machine learning, facilitating accurate and reliable predictions.

3.7 Variance Inflation Factor (VIF) Analysis

To address multicollinearity among the features, the Variance Inflation Factor (VIF) was calculated for all variables in the dataset. Variables with a VIF value greater than 12 were considered highly collinear and subsequently removed to improve model performance and stability.

The following features were removed due to high VIF values:

- **Driver Substance Abuse**: VIF = 20.51
- **Driver Distracted By**: VIF = 20.00
- **Light**: VIF = 14.70

This step ensured the remaining features had minimal multicollinearity, thereby improving the interpretability and predictive capabilities of the model.

3.8 Standard Scaling

To standardize the data, all numerical features were scaled using standard scaling. This process involved transforming the features to have a mean of 0 and a standard deviation of 1. Standard scaling ensures that all features contribute equally to the modeling process, preventing variables with larger magnitudes from dominating the results.

3.9 Handling Class Imbalance

The dataset exhibited class imbalance, which could lead to biased model predictions. To address this issue, three techniques were employed to balance the data:

- **Stratified Sampling**: Ensured that each class was proportionally represented in the training and testing datasets, preserving the original distribution of classes while balancing subsets for modeling.
- **SMOTE (Synthetic Minority Oversampling Technique)**: Generated synthetic samples for the minority class by interpolating between existing samples, thus increasing the representation of the minority class without introducing duplicates.
- **Tomek Links**: Identified and removed instances that were ambiguous or likely to be incorrectly classified by being close to the decision boundary, improving class separability and model robustness.

By combining these methods, the class imbalance was effectively reduced, ensuring that the model's performance metrics were not biased toward the majority class.

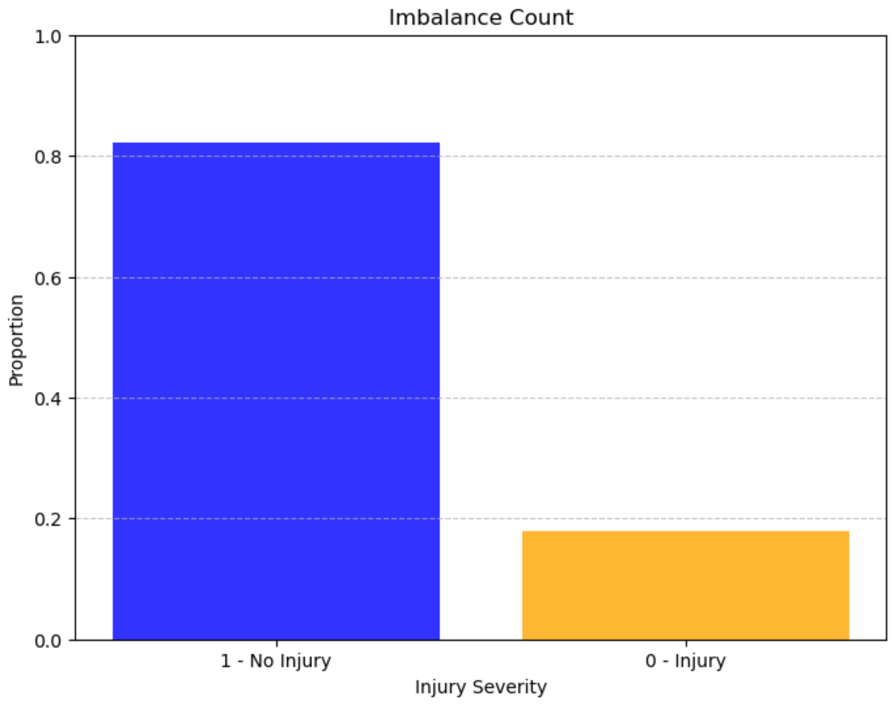


Figure 14: Class Imbalance

4 Results

4.1 Model Training and Evaluation

To build and assess predictive models, the dataset was split into training and testing sets using a 70-30 ratio. The training set was used to fit the models, while the testing set was reserved for evaluating their performance.

4.1.1 Models Trained

Four machine learning models were trained on the dataset:

- **Logistic Regression**: A baseline model to assess the dataset's linear separability.
- **Bagging Decision Trees**: An ensemble method to improve model stability and accuracy by combining multiple decision trees.
- **Gradient Boosting Classifier**: A boosting method that builds models sequentially to minimize prediction errors.
- **XGBoost**: An advanced gradient boosting algorithm optimized for efficiency and performance.

4.1.2 Evaluation Metrics

The following metrics were used to assess the performance of the trained models:

- **Accuracy**: The overall correctness of the predictions.
- **Precision**: The proportion of true positive predictions among all positive predictions.
- **Recall**: The ability of the model to identify all actual positive instances.
- **AUC (Area Under the Curve)**: Measures the model's ability to distinguish between classes.
- **Model Runtime**: The time taken to train and evaluate each model.
- **AIC (Akaike Information Criterion)**: Used to assess model quality relative to others, balancing goodness of fit and complexity.

This combination of metrics ensured a comprehensive evaluation of each model's effectiveness, efficiency, and suitability for the dataset.

4.2 Stratified Sampling Results

The table below summarizes the performance of the models trained using stratified sampling. Various metrics were evaluated to compare the effectiveness and efficiency of each model:

Table 1: Performance Metrics for Models using Stratified Sampling

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC**	**Runtime (sec)**	**AIC**
Logistic Regression	0.8702	0.8752	0.8702	0.8724	0.9298	1.6013	30.4750
Bagging Decision Tree	0.8760	0.8782	0.8760	0.8770	0.9353	8.1031	30.2386
Gradient Boosting	0.8866	0.8963	0.8866	0.8902	0.9479	8.5629	30.4108
XGBoost	0.8863	0.8947	0.8863	0.8895	0.9474	0.8328	30.4136

The results indicate that:

- **F1-Score**: The highest F1-score was achieved by the Gradient Boosting model (0.8902), closely followed by XGBoost (0.8895).
- **AIC**: The lowest AIC value was observed for the Bagging Decision Tree model (30.2386), indicating better model fit relative to its complexity.
- **Runtime**: XGBoost had the least runtime (0.8328 seconds), making it the most efficient model computationally.

4.3 SMOTE Results

Table 2: Performance Metrics for Models using SMOTE

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC**	**Runtime (sec)**	**AIC**
Logistic Regression	0.8475	0.9866	0.8256	0.8990	0.9310	3.5947	30.5196
Bagging Decision Tree	0.8737	0.9540	0.8892	0.9204	0.9338	17.2302	30.2265
Gradient Boosting	0.8735	0.9835	0.8605	0.9179	0.9442	14.4234	30.4351
XGBoost	0.8725	0.9833	0.8595	0.9172	0.9435	0.6238	30.4380

The results indicate that:

- **F1-Score**: The highest F1-score was achieved by the Bagging Decision Tree model (0.9204), showcasing its strong balance between precision and recall.
- **AIC**: The lowest AIC value was observed for the Bagging Decision Tree model (30.2265), indicating its superior balance of goodness-of-fit and complexity.
- **Runtime**: XGBoost demonstrated the least runtime (0.6238 seconds), making it the most computationally efficient model.

4.4 Tomek-Links Result

The table below summarizes the performance of the models trained using Tomek Links to handle class imbalance. Various metrics were evaluated to compare the effectiveness and efficiency of each model:

Table 3: Performance Metrics for Models using Tomek Links

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC**	**Runtime (sec)**	**AIC**
Logistic Regression	0.8658	0.9186	0.9179	0.9183	0.9270	15.9971	30.4795
Bagging Decision Tree	0.8764	0.9303	0.9183	0.9243	0.9356	7.3762	30.2390
Gradient Boosting	0.8866	0.9498	0.9101	0.9295	0.9481	8.1651	30.4093
XGBoost	0.7663	0.8908	0.8155	0.8515	0.6775	0.5719	37.6033

The results indicate that:

- **F1-Score**: The Gradient Boosting model achieved the highest F1-score (0.9295), making it the most balanced in terms of precision and recall.
- **AIC**: The Bagging Decision Tree model had the lowest AIC value (30.2390), indicating its strong balance between model fit and complexity.

- **Runtime**: XGBoost demonstrated the least runtime (0.5719 seconds), but its overall performance was lower compared to other models due to a significantly lower AUC (0.6775).

4.5 Hyperparameter Tuning

Hyperparameter tuning is the process of optimizing a model's hyperparameters to enhance performance and prevent overfitting or underfitting. This involves systematically searching for the best combination of hyperparameters to achieve the highest possible model accuracy and generalizability.

4.5.1 Techniques for Hyperparameter Tuning

Two methods were considered for hyperparameter tuning:

- **GridSearchCV**: This method exhaustively tests all possible combinations of specified hyperparameters, ensuring that the optimal combination is found. However, it can be computationally expensive for large search spaces.
- **RandomizedSearchCV**: Instead of testing all combinations, this method evaluates a random subset of the hyperparameter space. It provides faster results for large and complex models, making it a practical alternative when time or computational resources are limited.

4.5.2 Implementation

The best-performing model from each balancing technique (Stratified Sampling, SMOTE, and Tomek Links) was selected for hyperparameter tuning. The goal was to further enhance the models' performance by refining hyperparameters such as learning rate, maximum depth, and the number of estimators.

4.6 Tuned Results

4.6.1 Stratified Sampling

Table 4: Performance Metrics for Models after Hyperparameter Tuning (Stratified Sampling)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC**	**Runtime (sec)**	**AIC**
Stratify Logistic Regression	0.8763	0.8822	0.8763	0.8788	0.9349	340.6387	30.4629
Stratify Bagging Decision Tree	0.8885	0.8956	0.8885	0.8913	0.9488	3840.4495	30.3906
Stratify Gradient Boosting	0.8904	0.8982	0.8904	0.8934	0.9500	481.5378	30.3980
Stratify XGBoost	0.8897	0.8966	0.8897	0.8924	0.9500	208.6539	30.4019

The results indicate that:

- **F1-Score**: The highest F1-score was achieved by the Gradient Boosting model (0.8934), closely followed by XGBoost (0.8924).
- **AUC**: Both Gradient Boosting and XGBoost achieved the highest AUC value of 0.9500, reflecting strong classification performance.

- **Runtime**: XGBoost demonstrated the least runtime (208.6539 seconds), making it the most computationally efficient model post-tuning.

4.6.2 SMOTE

Table 5: Performance Metrics for Models after Hyperparameter Tuning

Model	Accuracy	Precision	Recall	F1-Score	AUC	Runtime (sec)	AIC
Logistic Regression	0.8448	0.9923	0.8174	0.8964	0.9330	117.3745	30.5179
Bagging Decision Tree	0.8806	0.9043	0.8806	0.8876	0.9399	2592.7816	30.3460
Gradient Boosting	0.8795	0.9150	0.8795	0.8884	0.9458	876.6295	30.4433
XGBoost	0.8806	0.9144	0.8806	0.8892	0.9459	111.4456	30.4262

The results indicate that:

- **F1-Score**: XGBoost achieved the highest F1-score (0.8892), closely followed by Gradient Boosting (0.8884).
- **AUC**: XGBoost also had the highest AUC value (0.9459), indicating its strong classification performance.
- **Runtime**: XGBoost demonstrated the least runtime (111.4456 seconds), showcasing its computational efficiency compared to other models, particularly Bagging Decision Tree.

4.6.3 Tomek-Links

Table 6: Performance Metrics for Models after Hyperparameter Tuning

Model	Accuracy	Precision	Recall	F1-Score	AUC	Runtime (sec)	AIC
Logistic Regression	0.8759	0.8854	0.8759	0.8796	0.9349	15.4533	30.4610
Bagging Decision Tree	0.8791	0.8740	0.8791	0.8760	0.9413	47.7762	30.3313
Gradient Boosting	0.8898	0.8982	0.8898	0.8930	0.9493	365.7060	30.4097
XGBoost	0.8896	0.8978	0.8896	0.8927	0.9492	196.1555	30.4139

The results indicate that:

- **F1-Score**: Gradient Boosting achieved the highest F1-score (0.8930), indicating strong balance between precision and recall.
- **AUC**: Gradient Boosting also had the highest AUC (0.9493), slightly outperforming XGBoost (0.9492).
- **Runtime**: Logistic Regression demonstrated the fastest runtime (15.4533 seconds), while XGBoost was significantly more efficient than Gradient Boosting (196.1555 vs. 365.7060 seconds).
- **AIC**: Bagging Decision Tree had the lowest AIC (30.3313), suggesting a better balance between model fit and complexity.

5 Conclusion

5.1 Key Findings

Across all the experiments and balancing techniques, the models demonstrated similar performance, with differences in metrics generally within a 5% margin. This consistency across models was an unexpected finding, indicating that the dataset's characteristics allowed multiple approaches to yield comparable results.

- **Best Model by AIC**: According to the AIC criterion, the best model was the SMOTE Bagging Decision Tree model, which achieved an AIC score of 30.2265. This indicates its superior balance between goodness of fit and model complexity.
- **Most Consistent Model**: XGBoost emerged as the most consistent performer, demonstrating competitive classification metrics, including F1-score and AUC, across all experiments. Additionally, it had significantly lower runtimes compared to Gradient Boosting and Bagging Decision Tree models, making it a strong candidate for applications requiring efficiency.

These results underscore the importance of evaluating models across multiple criteria to identify the most suitable approach for a given problem. While the SMOTE Bagging Decision Tree model excelled in AIC, XGBoost consistently balanced performance and runtime efficiency.

5.2 Modelling Challenges

5.2.1 Challenges and Limitations

The process of preparing the dataset for modeling posed several challenges, which influenced the overall results and scope of the study:

- **Data Cleaning**: Cleaning the dataset was a significant challenge. Many features required extensive preprocessing due to inconsistencies and missing values. With more resources, more advanced feature engineering techniques could have been applied, such as creating interaction terms or exploring additional external data sources to enhance predictive power. Furthermore, keeping the target variable as a multiclass problem rather than simplifying it to a binary classification task could have provided richer insights into accident severity levels.
- **Imputing Missing Values**: Imputing missing values was particularly challenging. Advanced techniques like KNN imputation produced abnormal and inconsistent values for some features, which were unsuitable for the dataset. As a result, simpler approaches like median imputation were used, which, while effective, may have resulted in the loss of subtle patterns in the data.

Despite these challenges, the preprocessing steps implemented ensured a clean and functional dataset, laying a strong foundation for model development and evaluation.

References

[1] Chawla, N. V., Bowyer, K. W., Hall, L. O., Kegelmeyer, W. P. (2002). *SMOTE: Synthetic Minority Over-sampling Technique.* Journal of Artificial Intelligence Research, 16, 321–357. doi:10.1613/jair.953

[2] Chen, T., Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System.* Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. doi:10.1145/2939672.2939785

[3] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). *Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research, 12, 2825–2830. doi:10.5555/1953048.2078195

[4] Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., ... Altman, R. B. (2001). *Missing value estimation methods for DNA microarrays.* Bioinformatics, 17(6), 520–525. doi:10.1093/bioinformatics/17.6.520

[5] Bradley, A. P. (1997). *The use of the area under the ROC curve in the evaluation of machine learning algorithms.* Pattern Recognition, 30(7), 1145–1159. doi:10.1016/S0031-3203(96)00142-2

[6] Zheng, A., Casari, A. (2018). *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists.* O'Reilly Media, Inc.

[7] Rahm, E., Do, H. H. (2000). *Data Cleaning: Problems and Current Approaches.* IEEE Data Engineering Bulletin, 23(4), 3–13.

[8] Powers, D. M. W. (2011). *Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation.* Journal of Machine Learning Technologies, 2(1), 37–63.

[9] James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). *An Introduction to Statistical Learning: With Applications in R.* Springer.