The Influence of Weather and Road Conditions on Road Accident Severity

Reported by: Jenlarp Jenlarpwattanakul

School of Electrical Engineering and Computer Science
The University of Queensland

Submitted for the degree of Master of Data Science Capstone 2

November 6, 2024

Contents

Executive Summary

Road accidents are a serious and increasing public safety problem in Queensland. These accidents often lead to tragic outcomes, like loss of life, severe injuries, and high costs for property damage and medical care. For the last twenty years, many road safety programs have tried to reduce the number of crashes. However, recent data shows that accidents have gone up slightly in the past five years. Most research on this topic has focused on driver behavior and traffic conditions (Çeven and Albayrak, 2024; Birfir et al., 2023). But many studies have not paid enough attention to the important role of weather and road conditions in affecting how severe accidents can be. Today's traffic systems are more complicated than ever, so it's crucial to gain a better understanding of these environmental factors. However, some research has revealed that the weather and road condition like sunny weather leading to more negative impact than thunderstorms (Hermans et al., 2007), road condition like slippery causing severe accident (Malin et al., 2019). Additionally, pavement condition also caused different severity(S. Chen et al., 2017). By doing so, we can build better predictive models and develop safety measures that address these specific risks more effectively.

This report aims to improve predictions of crash severity by examining how environmental factors affect accident outcomes. The dataset used in this project consists of 380,000 records from the Queensland road crash database, covering the years 2001 to 2023 (Transport and Main Roads, 2024). The 2 main objectives are to apply machine learning models to deliver practical insights and to investigate the impact of weather and road conditions on crash severity.

To address the issue, I utilized five severity categories; fatal, hospitalisation, medical treatment, minor injury, and property damage only with key weather-related variables, such as atmospheric, lighting, and road surface conditions. Additionally, since the dataset has some noise and outliers, the process to handle and prepare data consists of data cleaning, outlier removal, and encoding. I developed and evaluated several models—Random Forest, XGBoost, LightGBM, and a Multilayer Perceptron (MLP)—employing resampling techniques like SMOTE, undersampling, and hybrid sampling, SMOTEENN and SMOTEtomek to address class imbalances (X. Wang et al., 2023; Bach et al., 2019; Sasada et al., 2020), respectively.

Overall, the Random Forest model showed average performance and tended to favor the more common classes. XGBoost and LightGBM do a better job at making balanced predictions, with LightGBM performing the best. The MLP model performed well at understanding complex patterns but had trouble dealing with class imbalance. Even with some improvements, all models struggled to accurately predict the less common but very important classes, like fatal and minor injury crashes. The report will cover the steps of preparing the data, building the models, and evaluating them using accuracy, precision, recall, and f1-score metrics to make sure performance is judged fairly. The results showed that weather and road conditions greatly affect accident severity, and LightGBM and MLP models promising for future improvements.

I offer three key recommendations. First, adding detailed environmental data—such as road surface temperature, humidity, windspeed, and rainfall intensity—could make predictions more accurate and relevant to real-world conditions (Malin et al., 2019). Second, employing advanced class imbalance techniques, ADASYN and BorderlineSMOTE (TurinTech, 2022) could improve model performance. Third, utilizing SHAP analysis to explain predictions and identify key features could enhance model interpretability and prioritize interventions (Lundberg and Lee, 2017). Accurate crash severity prediction is vital for public safety. By enriching the dataset with more environmental details and using advanced analytical methods, this research could drive better road safety planning and resource allocation, significantly contributing to reducing road accidents in Queensland.

1 Introduction

Road accidents represent a significant threat to public safety, with consequences ranging from loss of life and injuries to property damage. Leveraging comprehensive datasets spanning from 2001 to 2023 (Transport and Main Roads, 2024). There has been a concerning trend of slight increases in crash incidents in Queensland over the past five years. While existing research often delves into driver characteristics and external factors such as traffic density, the impact of weather and road conditions remains an overlooked yet critical aspect (Çeven and Albayrak, 2024; Birfir et al., 2023). The dataset consists of 380316 records with 53 columns, one of the columns, fatal, hospitalisation, medical treatment, minor injury, and property damage only is considered as a target, imbalanced classes, for prediction. Imbalanced classes are accounted for 1.53% fatal, 13.70% minor injury, 22.99% property damage only, 30.51% hospitalisation, and 31.27% medical treatment. Furthermore, this dataset covers attributes such as atmospheric conditions, lighting conditions, and road conditions. These attributes consist of which can be the main features for predicting severity classes.

It is well known that weather has a significant impact on the number and seriousness of traffic accidents. For example, studies indicate that while sunny weather can have the opposite impact, chilly temperatures may lower accident rates. According to (Hermans et al., 2007), thunderstorms are also linked to incidents that cause minor injuries. Another study evaluated the proportional risk of accidents in relation to several weather conditions, determining that slushy roads, low visibility, freezing rain, and slippery roads were the most dangerous. The largest accident likelihood was found to be on slippery roads, which increased the risk by over 50% (Malin et al., 2019). These results highlight how weather significantly affects the intensity of traffic on the roads.

Regarding road conditions, research has examined the safety implications of pavement conditions on rural roads. According to this study, different crash severity levels were correlated with varied road surface conditions. In particular, there were fewer collisions on segments with broader lanes than on those with narrower lanes. Additionally, it has been demonstrated that a one-foot increase in lane width can lower the risk of fatal collisions on road segments with pavements classified as excellent, good, good-fair, or fair quality (S. Chen et al., 2017). Thus, weather and road conditions have been designated as the main focus of this study due to their significant impact on traffic safety.

The project's methodology involves thorough exploratory data analysis, data preparation including cleaning, outlier removal, data encoder, followed by the development of robust machine learning models (Random Forest, XGBoost, L-GBM, MLP). By systematically exploring the interplay between weather, road conditions, and accident severity, this project aims to address this gap by assessing the severity of road accidents across various classes, including fatal, hospitalization, medical treatment, minor injury, and property damage, attributing them to weather and road conditions. Additionally, it aims to provide valuable insights for enhancing road safety measures and reducing the toll of road accidents on public health and infrastructure.

1.1 Data Privacy and Ethical Concerns

The dataset used in this project follows strict privacy and ethical guidelines set by the Queensland Government's Open Data Policy to ensure that individual privacy is protected and the data is used responsibly. The dataset contains information about road accidents, including crash locations, conditions, and severity, but it is carefully anonymized to prevent the identification of individuals involved in these incidents (Transport and Main Roads, 2024). Sensitive information was not made publicly accessible. Sensitive data must be anonymised or withheld in accordance with the Information Privacy Act of 2009. Because of this, individuals nor communities were harmed by the project's dataset. By adhering to these rules, I maintained the highest standards of ethics and privacy while ensuring that the data continues to be a useful instrument for social benefit (Queensland Government, 2024)

.

2 Related Works

For this section, I will introduce some related works that utilized crash dataset and performed crash severity predictions through multiclassification. I'll start by going over the available machine learning models, sampling techniques, and features in these projects.

One of the study delved into predicting crash severities using a range of machine learning models, including The random forest (RF), extreme gradient boosting (XGBoost), light gradient-boosting machine (L-GBM), decision jungle (DJ), cat-boost (Categorical Boosting), and adaBoost (Adaptive boosting). The accuracy of these models was 81.45%, 79.87%, 76.94%, 74.12%, 69.68%, and 65.61%, in that order (S. Ahmed, Hossain, et al., 2023). Additionally, oversampling was the only solution used to address unbalanced classes. The specific parameters used in this study were then as follows: year, day, kind of intersection, gender, weather, alcohol use, and so forth. Following by, the detail of features used in this study included year, day, junction type, gender, alcohol consumption, weather conditions, and etc. Nonetheless, Deep learning models and other sampling techniques like undersampling and hybrid techniques were not explored in this study, which could lead to better performance. The attributes explained for the final result were not fully explored, especially the impact of weather and road conditions on accident severity (see in Figure 1).

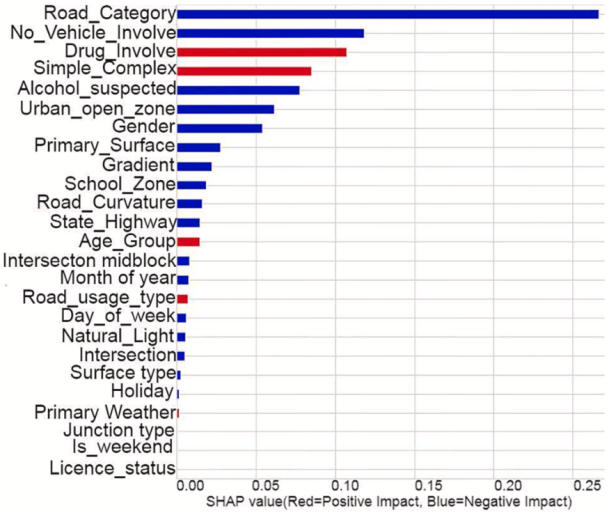


Figure 1: Feature importance using SHAP for RF from the first related study (S. Ahmed, Hossain, et al., 2023).

Similarly, another study investigated various machine learning models, such as Random Forests, Decision Trees, and Logistic Regression without applying any sampling methods. The study forecasted three severity classifications using variables like weather, vehicle status, road surface conditions, and illumination. With corresponding values of 86.23%, 85.74%, and 86.86%, the models achieved remarkable accuracy (Vanitha and Swedha, 2023). Although they showed high accuracy, I discovered that the model performance was overfitting in correctly predicting only a single class, which could result in unreliable models Even though this study may not mention about imbalanced class issues, the result of metrics could not be comprehensively predicted. This could indicate that model struggled in training on imbalanced classes. Therefore, the study was unable to adequately examine the model performance (see in Figure 2).

(Accuracy, 86.23)

	precision	recall	f1-score	support
1	0.000000	0.000000	0.000000	4111
2	0.000000	0.000000	0.000000	38151
3	0.862320	0.999996	0.926069	264697
micro avg	0.862317	0.862317	0.862317	306959
macro avg	0.287440	0.333332	0.308690	306959
Weighted avg	0.743596	0.862317	0.798568	306959

Predicted	1	3	All
Actual			
1	0	4111	4111
2	0	38151	38151
3	1	264696	264696
All	1	306958	306959

Figure 2: Logistic regression result from second related work (Vanitha and Swedha, 2023)

Consequently, In this project, I have explored undersampling and hybrid sampling techniques including undersampling, SMOTEENN, and SMOTETomek which were used to balance classes and also built neural network which was multilayer perceptron (MLP) in comparison to ensemble methods like random forest, extreme gradient boosting, and light gradient boosing machine learning. Moreover, I also considered various attributes such as road conditions and weather conditions available in the dataset combined with other features like traffic control and crash characteristics to obtain more comprehensive results and deeper insights and focused on balancing metrics rather than achieving high accuracy in terms of overall performance.

3 Objectives

This project aims to enhance model performance in multiclassification through machine learning techniques and to demonstrate the influence of features, particularly weather and road conditions, on prediction accuracy. To achieve these objectives, the following actions will be undertaken:

- Identification of the most impactful features affecting accident severity, with a focus on weather variables and road condition variables.
- Development of a high-performance model by using ensemble models and neural network model.

My primary focusing on feature selection will be on variables related to weather conditions, road conditions, and factors influencing accident severity. In the dataset, which consists of 5 severity classes and 52 features with varying value ranges. This entails considering features such as atmospheric conditions, lighting conditions, and road surface conditions as they directly impact the likelihood and severity of accidents. Additionally, features like a car speed variable will be included due to its direct correlation with each severity class. I also develop the model performance utilizing neural network model and ensemble models, which can provide the importance of each feature information.

Furthermore, addressing the issue of imbalanced classes is crucial for model performance in multiclassification. I will apply class combination and sampling methods to mitigate imbalanced classes affecting the performance. This can ensure that the model can equally learn and predict the occurrence of each severity category.

Nevertheless, the model can show an overall high accuracy, it may not indicate that the model performs well for all classes. For this reason, assessing its effectiveness in predicting accident severity across different classes. Therefore, I will examine these metrics for each severity class to gain an understanding of the model's strengths and weaknesses by comprehensively evaluating the model's performance.

4 Data Description

This study utilizes crash location data from Queensland roads, extracted from the Queensland road crash database. The data in this database are collected over the 12 months preceding each crash, as investigations into these incidents typically take one year to complete. The dataset contains 380,316 rows and 53 columns, detailing traffic crashes, including the time, date, location, and specific conditions such as weather, road surface, and traffic controls. Additionally, it provides information on the severity of casualties and the characteristics of crahses or pedestrians involved, which can be used to predict crash severity under various conditions. Selecting the relevant features that influence the severity of car accidents is a critical aspect of this analysis (see in Table 1). The target variable for this study is crash severity, classified into five categories: fatal, hospitalisation, medical treatment, minor injury, and property damage only(see in Table 2).

Table 1: Road accident influencing factors.

Category	**Factors**
Crash context	Crash nature, Crash type
Road and environmental conditions	Crash road surface condition, Crash atmospheric condition, Crash lighting condition, Crash road horiz align, Crash road vert align
Traffic control and regulation	Crash traffic control, Crash speed limit
Administrative and geographical details	Crash controlling authority, Crash roadway feature
Temporal information	Crash month

Table 2: Definitions of crash severity classes.

Class	**Definition**
Fatal	A person who dies within 30 days from injuries sustained in a road traffic crash.
Hospitalisation	A person transported to hospital, from injuries sustained in a road traffic crash, who does not die within 30 days of the crash.
Medical treatment	A person requiring medical treatment administered by a medical officer (e.g., a doctor), but not transported to hospital, from injuries sustained in a road traffic crash.
Minor injury	A person with minor injuries sustained in a road traffic crash but not requiring medical treatment.
Property damage only	No injuries; damage is limited to property only.

4.1 Geospatial Analysis of Crash Severities in Queensland

According to the map(see in Figure 1), the map displays crash data across the entire state of Queensland, highlighting a dense concentration of incidents particularly from the southeastern to the northeastern regions. The areas with the darkest tones indicate the highest frequency of crashes, suggesting these routes are high-risk zones. This visualization helps identify specific sections of roads that may require focused safety measures (see in Figure 3).

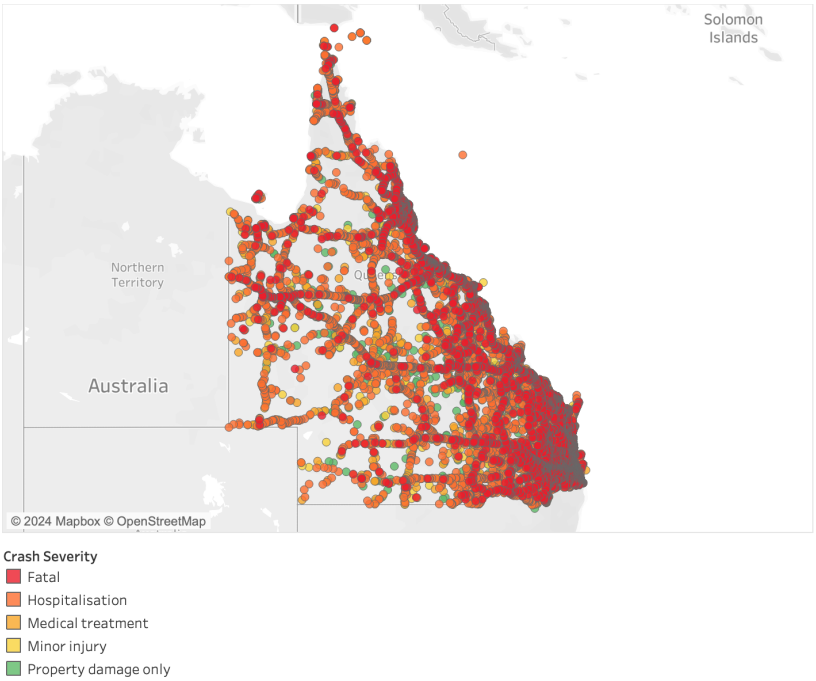


Figure 3: Crash location from Queensland roads

4.2 Distribution of Crash Severities in Queensland

This bar chart represents the distribution of crash severity for traffic incidents in Queensland, categorized into five types: Fatal, Hospitalisation, Medical treatment, Minor injury, and Property damage only (see in Figure 4). The chart illustrates that most crashes resulted in medical treatment or hospitalization, highlighting the significant impact of road traffic incidents on public health and the medical system. Minor injuries and property only damages also constitute a considerable proportion, underscoring the broad spectrum of crash severities. Fatalities, while the least frequent, represent the most critical concern due to their irreversible consequences. This visualization is crucial for understanding the severity landscape of road incidents in Queensland, aiding in targeted interventions and resource allocation for improved road safety (see in Figure 4).
The counts of incidents per category are as follows:

Fatal: 5,828 crashes, shown in red, representing the most severe outcome with fatalities.

Hospitalisation: 115,869 crashes, in orange, indicating incidents where victims required hospital care.

Medical treatment: 118,765 crashes, in yellow, involving injuries that necessitated medical attention beyond first aid.

Minor injury: 52,046 crashes, in lighter yellow, where victims sustained minor injuries.

Property damage only: 87,210 crashes, in green, where incidents resulted in property damage without personal injuries.

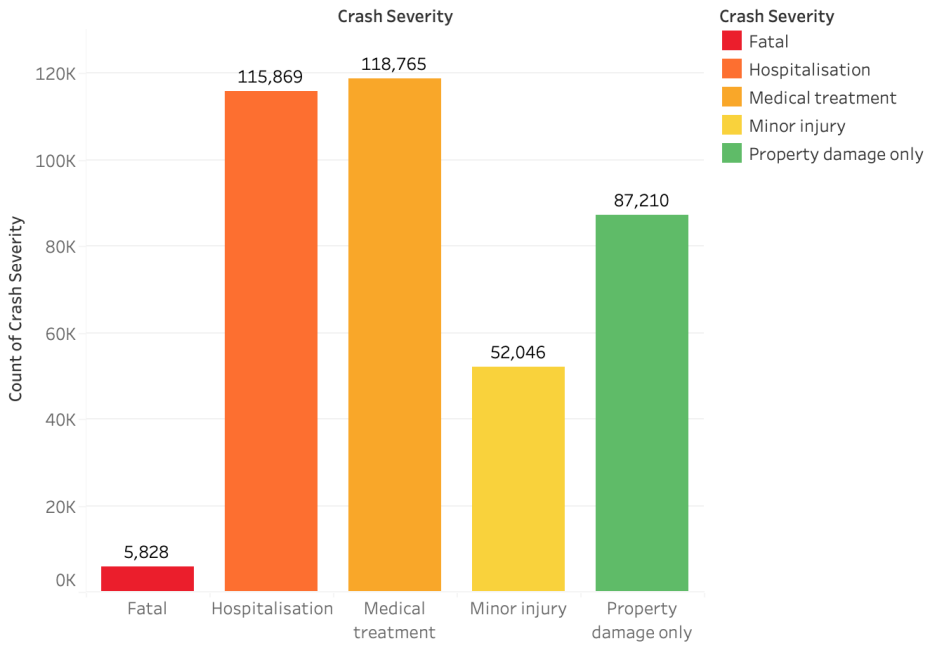


Figure 4: Distribution of Crash Severity in Queensland. This bar chart shows the number of incidents categorized by severity, highlighting the frequency of hospitalizations and medical treatments compared to other outcomes.

4.3 The Trends in Crash Severities in Queensland

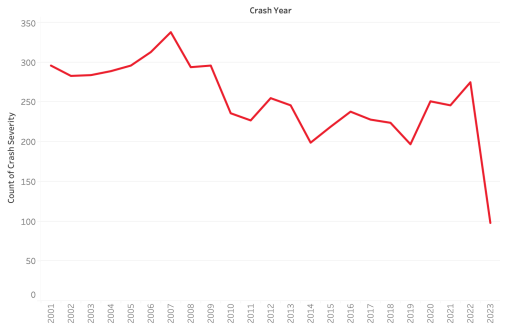
Fatal crashes: The graph depicting fatal crashes shows a peak around 2004, followed by a general decline until around 2013. A slight recovery is observed until 2019. The sharp decrease in 2023 is attributed to incomplete data collection, as data for only about 5 months of the year were recorded (see in Figure 5).

Hospitalisation: The hospitalisation trend displays a gradual increase over the years, with a pronounced stability between 2005 and 2011 and a peak around 2021. The overall increasing trend suggests an enduring severity in crashes that necessitate hospital care.

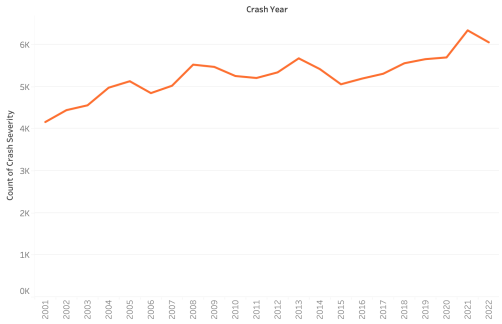
Medical treatment: Crashes requiring medical treatment show a relatively stable trend with minor fluctuations up until 2010, followed by a downward trend with some variability until 2022. The decline in 2010 could reflect improvements in road safety measures or medical response efficiency.

Minor injury: The graph for minor injury crashes shows fluctuations over the years, with a notable dip around 2011 and a recovery peaking in 2022. This category seems to show the least consistent trend, suggesting varied factors influencing the frequency of minor injuries from crashes.

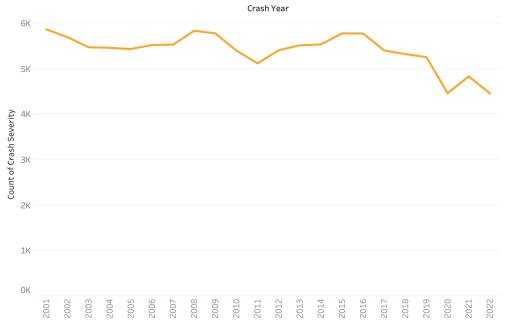
Property damage only: The trend for property damage only crashes initially shows a steady increase until about 2004, followed by a slight decline and then a period of stability. It is observed towards the latest years. The general stability in this category indicates consistent reporting and possibly consistent occurrence rates of such incidents over the years.



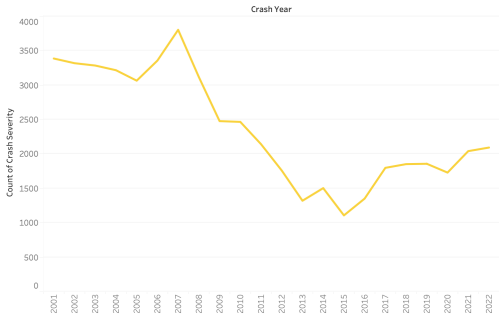
(a) Trend of fatal crashes



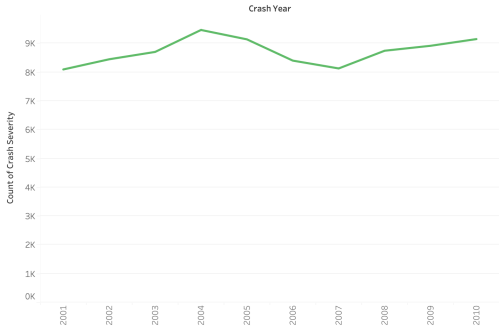
(b) Trend of hospitalisation



(c) Trend of medical treatment crashes

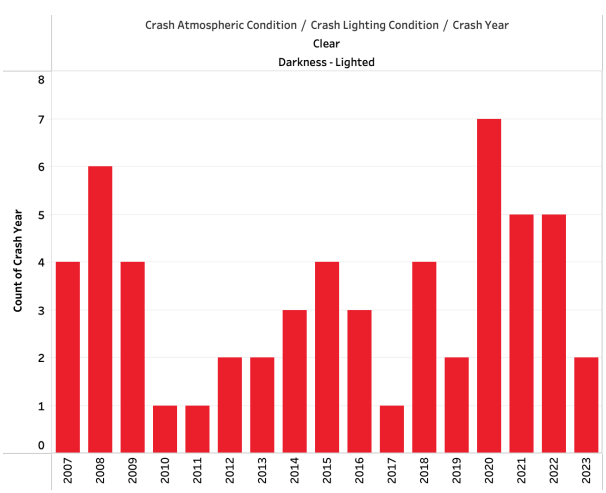


(d) Trend of minor injury crashes

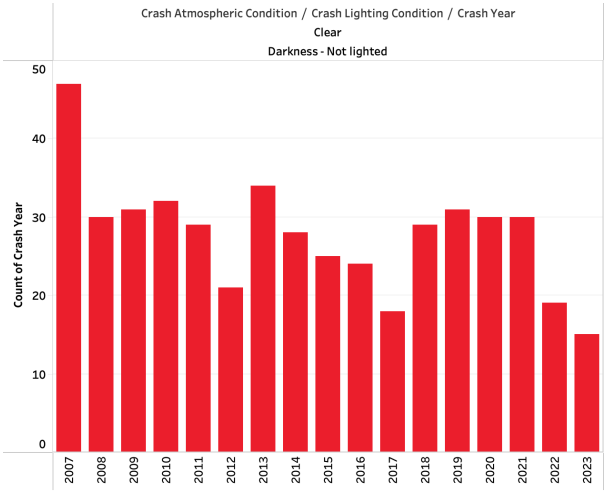


(e) Trend of property damage crashes

Figure 5: Trends in crash severities in Queensland: (a) Fatal crashes, (b) Hospitalisation, (c) Medical treatment crashes, (d) Minor injury crashes, and (e) Property damage crashes.

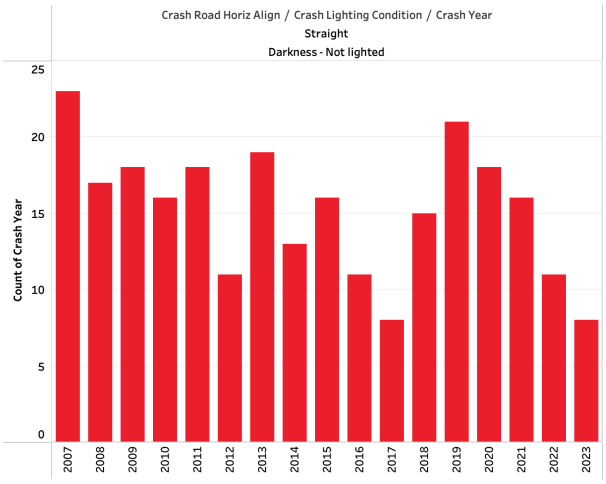


(a) The fatal crash distribution under lighted conditions

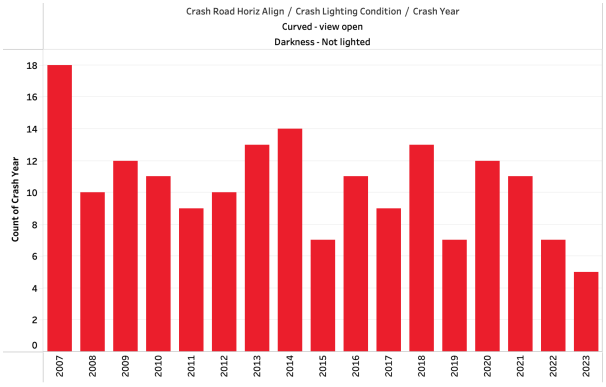


(b) The fatal crash distribution under non-lighted conditions

Figure 6: Crash distribution with clear atmosphere under different lighting conditions from 2007 to 2023: (a) Lighted conditions, (b) Non-lighted conditions.



(a) The fatal crash distribution with straight road condition under non-lighted conditions.



(b) The fatal crash distribution with view-open road condition under non-lighted conditions.

Figure 7: Crash distribution under different lighting conditions and horizontal road alignments from 2007 to 2023: (a) straight road conditions, (b) view-open road conditions.

From the observed crash distribution over the years, I prioritized this decline over trends in minor injury crashes. Each bar chart was plotted based on the same weather and highest speed limit features, clear weather condition and 100 to 100 km/h as they resulted in highest counts of fatal crashes. To further understand this reduction, I analyzed fatal crashes in relation to specific features such as atmospheric conditions, road alignment, and lighting conditions. It shows that non-lighted conditions plays an vital role in these crash outcomes. Furthermore, with non-lighted conditions, curved roads with open views, and on straight horizontal road alignments also leads to fatal crashes (see in Figure 6 and 7).

In terms of analysis, I might suggest that the improvement in road safety infrastructure after 2007 may have played a role in the decline in fatal collisions in locations with a higher concentration of darkness. The speed limit has also been strictly enforced, which has reduced the number of serious collisions. However, the persistent number of fatal crashes on both straight and curved roadways indicates that factors like visibility and road design are crucial.

5 Data Preparation

The dataset was prepared through four key steps: outlier removal, rare category removal, data grouping, and encoding. These methods are designed to enhance model performance and ensure that the model could handle the data without issues.

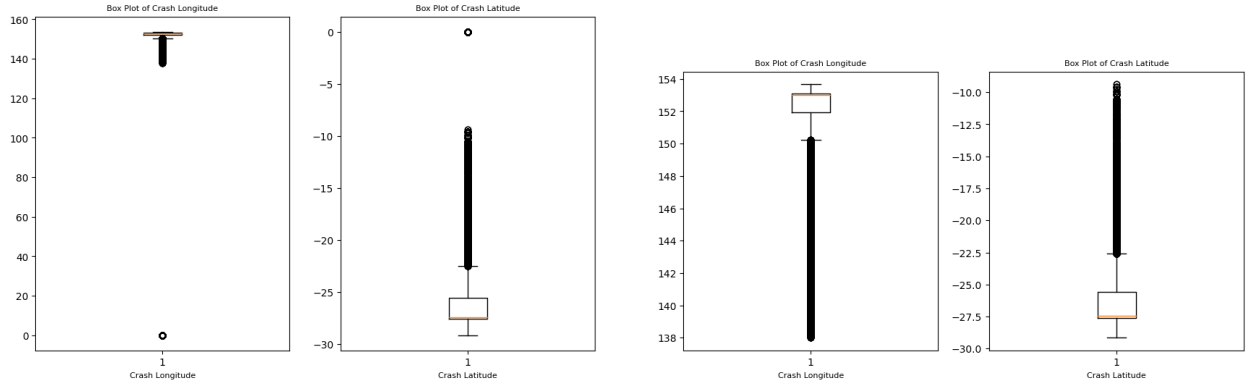
5.1 Outlier Removal using Boxplots

In this step, removing outliers I utilized box plot to detect outliers, particularly for datasets with geospatial information such as latitude and longitude values, not shown in the map. A boxplot consisted of minimum, first quartile (Q1), median, third quartile (Q3), and maximum. Any data points beyond the whiskers of the boxplot (which usually extend to 1.5 times the interquartile range (IQR) from Q1 and Q3) are flagged as potential outliers. This helped me identify values that could result from locations that do not exist on the map (Mazarei et al., 2024).

In this analysis, I set quantile filtering to detect the outlier removal process, setting the quantile threshold to 0.1. This can ensure that the values filterd was within the central 90% of the data distribution. It worked by removing both extreme upper and lower values. For example (see in Figure 8):

Lower bound: Any value below the 10th percentile (quantile = 0.1) is considered an outlier.

Upper bound: Any value above the 90th percentile (quantile = 0.9) is treated as an outlier.



(a) Before removing outliers

(b) After removing outliers

Figure 8: Comparison of the data distribution before and after removing outliers. (a) shows the boxplot with outliers. (b) shows the boxplot without outliers

5.2 Rare Category Removal

Some features in the dataset, such as Crash_Road_Surface_Condition, Crash_Atmospheric_Condition, and Crash_Lighting_Condition, contain categories labeled as "Unknown." Similarly, the Crash_Speed_Limit feature includes "None" values. These entries were removed because they are considered noise, potentially skewing the analysis due to their ambiguous nature (refer to Table 3 for details).

Table 3: Summary of data exclusions in crash dataset features

Dataset feature	**Number of exclusions**
Crash Road Surface Condition	313
Crash Atmospheric Condition	217
Crash Lighting Condition	377
Crash Speed Limit	3

5.3 Grouping Data by Time Periods

To simplify the analysis and identify seasonal trends more effectively, this study groups the Crash_Month data into quarters. Each quarter aggregated three consecutive months: Quarter 1 (January to March), Quarter 2 (April to June), Quarter 3 (July to September), and Quarter 4 (October to December).

Table 4: Quarterly distribution of crash severity types

Crash Month	**Fatal**	**Hospitalisation**	**Medical treatment**	**Minor injury**	**Property damage only**
Q1	1.48	29.83	31.86	13.99	22.83
Q2	1.54	30.63	31.29	13.53	23.01
Q3	1.56	30.78	30.94	13.67	23.04
Q4	1.56	30.94	31.11	13.53	22.87

5.4 Encoder

The dataset primarily consisted of categorical variables, it was essential to transform these into numeric formats to facilitate machine learning modeling. This transformation was accomplished using two encoding methods: one-hot encoding and label encoding. These encoding strategies converted categorical data into a format that was interpretable by machine learning algorithms, thus making the data suitable for both training and evaluation phases.

5.4.1 One-Hot Encoding

One-hot encoding was applied to nominal categories where no ordinal relationship exists, creating a new binary column, 0 and 1, for each feature(see examples in Table 6 and 7).

Table 5: Data representation before one-hot encoding

Crash_Road_Vert_Align
Grade
Crest
Grade
Level
Level

Table 6: Data representation after one-hot encoding

Crash_Road			
Vert_Align_Crest	**Vert_Align_Dip**	**Vert_Align_Grade**	**Vert_Align_Level**
0.0	0.0	1.0	0.0
1.0	0.0	0.0	0.0
0.0	0.0	1.0	0.0
0.0	0.0	0.0	1.0
0.0	0.0	0.0	1.0

5.4.2 Label Encoding

Label encoding, on the other hand, was used for ordinal data in the class column, Crash_Severity, assigning each category a unique integer, 0,1,2, and 3, based on the order. (see examples in Table 8).

Table 7: Data representation before and after label encoding

Crash_Severity	**Encoded**
Fatal	0
Hospitalisation	1
Medical treatment	2
Minor injury	3
Property damage only	4

5.5 Class Combination

This subsection considered the combination of Class 1 (Hospitalization) and Class 2 (Medical Treatment) due to their similarities of definition. The new class was defined as "Serious injury". The provided plots display the correlation using all features across the four classes in the dataset, comparing Class 1 and Class 2 against the other classes(see Figure 9).

(a) Class 1 and Other 4 Classes Comparison

(b) Class 2 and Other 4 Classes Comparison

Figure 9: Comparison of Class 1 and Class 2 against other 4 classes. (a) shows the feature correlation with the main class 1 compared to other 4 classes. (b) shows the feature correlation with the main class 2 compared to other 4 classes.

5.5.1 Comparison

Severity 0(Fatal): The correlation with Severity 1 (Hospitalization) shows a distinct peak and trough towards the end of the feature set, suggesting specific features that strongly differentiate Fatal outcomes from Hospitalization. In contrast, Severity 2 (Medical Treatment) displays a more subdued correlation pattern with Fatal outcomes, indicating less feature overlap.

Severity 3(Minor injury) and Severity 4 (Property damage): Both Severity 1 and Severity 2 show low correlation with minor injury and property damage, underscoring divergent characteristics that separate these outcomes from more medically intensive responses.

Between Severity 1 and Severity 2: The correlation between Hospitalization and Medical Treatment demonstrates opposing directions in several features, especially noticeable at the beginning and ending of the feature set. This suggests that while there are shared attributes, significant distinctions also exist, making them candidates for combining to capture a broader pattern of medical response.

Combining Severity 1 and Severity 2 named as "Serious injury"(see in Figure 10), based on the correlation plots(see Figure 9), demonstrates opposing directions in the same features, particularly evident at the beginning and ending parts of the graph. This decision is grounded in the hypothesis that the model could potentially recognize a broader pattern that encompasses both sets of behaviors. By effectively learning to identify a unified characteristic derived from these opposing features, the model may improve in recognizing and generalizing from the data, thus capturing a more comprehensive view of the underlying trends.

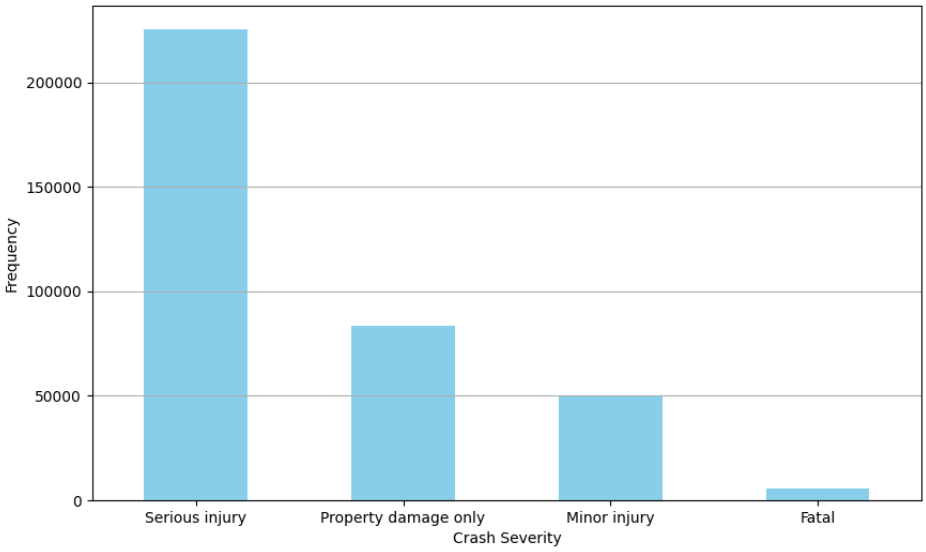


Figure 10: Frequency of crash severity

6 Methodology

6.1 Random Forest

Random forest is one of ensemble method used for combining classifier consisted of a large number of decision trees. For the classification problem, results are determined by voting in the decision trees. The algorithm is less sensitive to the hyperparameter configurations, so it can be utilized for model with minor adjustment and has effective performance in classification problem. Random forest independently generates each decision tree by samples called as boostrap samples(see the scheme in Figure 11). The classification error of the decision tree was computed by different trees and correlation between the trees. Furthermore, the increase of classification accuracy is determined by the vote of multiple decision trees (Zhou et al., 2023).

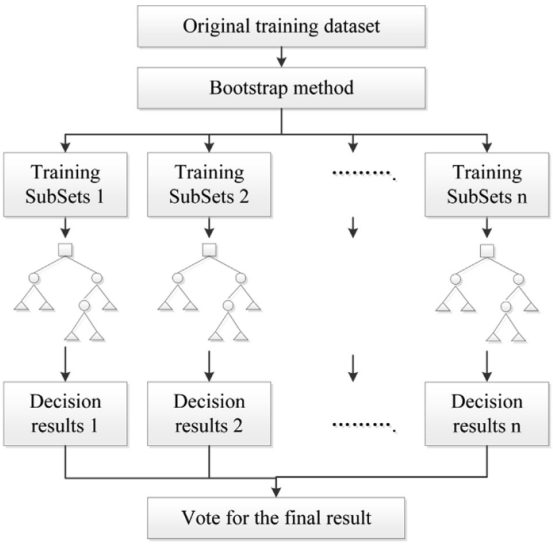


Figure 11: RF scheme (Zhou et al., 2023)

6.1.1 Concepts of Random Forest

Random forest is based on ensemble learning, creating multiple decision trees to form a comprehensive model. The model sends a collective prediction (Breiman, 2001) and trains each decision tree on a different bootstrapped portion of the data. Random forest model achieves superior accuracy and generalisability compared to a single decision tree (Liaw and Wiener, 2001) by averaging or selecting the majority vote from its decision trees.

Ensemble model works by combining several models to handle a specific problem, aims to balance the weaknesses of individual models through collective strength. This principle was detailed by (Breiman, 1996), which Random Forest utilizes by combining predictions from numerous trees (Breiman, 1996).

The technique of bagging, or bootstrap aggregation, involves training each tree on a randomly chosen subset of the training data, drawn with replacement. This can ensure that the uniqueness and variance reduction of the ensemble (Breiman, 1996). These subsets, generated by selecting N samples from the original dataset, introduce beneficial randomness that curtails the overfitting tendency in the model (Louppe, 2015).

6.1.2 Mathematical Formulation

At the core of a Random Forest is the decision tree, developed by iteratively dividing the feature space to reduce a specified loss function—such as gini impurity or entropy in classification tasks. (Quinlan, 1986). This splitting continues until the tree effectively categorizes or predicts the outcome, serving as the fundamental element of the broader Random Forest model.
The gini impurity is a measure used to evaluate the quality of a split in classification tasks. It measures the probability that a randomly chosen sample from the node would be incorrectly classified. For a node t with K classes, gini impurity $G(t)$ is defined as:

$$G(t) = 1 - \sum_{i=1}^{K} p_k^2$$

where pk is the proportion of samples belonging to class k at the node (James et al., 2013).
Entropy is another measure used to determine the impurity of a node. It quantifies the uncertainty or

randomness in the node's class distribution. For a node t, entropy $H(t)$ is calculated as:

$$H(t) = -\sum_{k=1}^{K} p_k log(p_k)$$

where p_k is the probability of a sample being in class k (James et al., 2013).

6.1.3 Random Feature Selection

For each node of every decision tree, random forest generates only a random subset of features, considered to make decision for the best split. For example, if there are p total features, a smaller subset m is chosen at each split, typically $m = \sqrt{p}$ for classification. This selection can not only decrease the correlation between trees, but also improve the model's performance (Louppe, 2015)

6.1.4 Aggregation of Predictions

Once all the trees in the random forest are built, the model aggregates their predictions. For instance, the classification tasks, this is performed by majority voting, where the class that considers the most votes across the trees becomes the final predicted class (Breiman, 2001). Formally, if there are n trees and each tree T_i provides a class prediction $\hat{y}_i$, the final prediction $\hat{y}$ is given by:

$$\hat{y} = mode(\hat{y}_1, \hat{y}_2, ..., \hat{y}_n)$$

6.1.5 Hyperparameter Tuning

Random Forest has several important hyperparameters that can be adjusted to improve its performance:

Number of trees: Increasing the number of trees in a model can improve accuracy but also requires more computational resources. Although adding more trees generally enhances the model's ability to generalize, the gains in performance tend to diminish after reaching a certain point (Louppe, 2015).

Max depth: Controlling the maximum depth of the trees helps prevent overfitting. Deeper trees can capture more complex patterns but may lead to overfitting if the depth is too large (James et al., 2013).

Min samples split and min samples leaf: These parameters determine the smallest number of samples needed to split an internal node and to form a leaf node. However, increasing these values can help prevent overfitting by promoting the creation of simpler trees (Breiman, 2001).

6.1.6 Extreme Gradient Boosting (XGBoost)

XGBoost is a strong classification algorithm that is famous for its performance as a tree-based gradient boosting ensemble model. The XGBoost model constructs multiple decision trees, where each tree is a weak learner (see scheme in Figure 12). However, through transformation techniques, each weak learner can be transformed to a strong learner. Although a weak classifier may not perform with high accuracy, the XGBoost performs very well by predicting a particular subset of the data (Zhu et al., 2022). The algorithm adopts some parameters that are tunabl to improve the accuracy. For example, the algorithm adopts regularisation parameters in a way that will reduce the risk of overfitting and improve its generalisation to unseen data (Tarwidi et al., 2023).

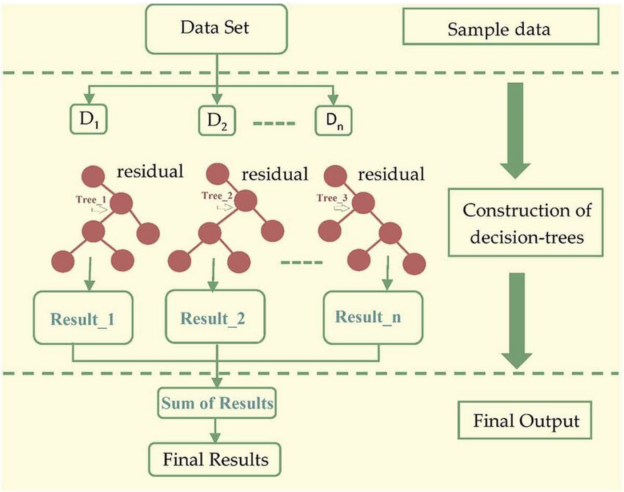


Figure 12: General architecture of XGBoost algorithm (S. Ahmed, Raza, et al., 2023)

6.1.7 Concepts of XGBoost

XGBoost is a type of gradient boosting used to create ensemble models, which are collections of simpler models that work together to acquire better predictions. In gradient boosting, each new decision tree is built to fix mistakes from the previous trees. XGBoost stands out because it uses more advanced calculations (specifically, the second-order derivatives of the loss function) for more precise updates. It also effectively handles missing data and includes features to reduce overfitting, like regularization (T. Chen and Guestrin, 2016).

Gradient boosting works by adding new models sequentially to correct previous errors. Each new model aims to the remaining inconsistency of comparing predicted and actual outcomes. Using gradient descent is used for reducing the error, adjusting the model to minimize a loss function. (Friedman, 2000). Gradient boosting has expanded into various forms, with XGBoost being one of the most popular versions.

6.1.8 Enhancements

XGBoost's power comes from several key enhancements that improve both speed and efficiency. It includes a new method for learning trees that can handle sparse data, along with a smart approach to approximate tree learning that deals with instance weights *citechentianqi*. XGBoost also speeds up model training through parallel and distributed computing, enabling faster exploration of models. It is designed to work even with massive datasets, allowing model to process hundreds of millions of records on a standard desktop using out-of-core computation *citechentianqi*. These techniques combine to create an efficient system that can handle very large datasets while using fewer resources(T. Chen and Guestrin, 2016).

In addition, XGBoost introduces regularization to prevent overfitting and uses pruning to simplify tree structures. The system is also designed to be "sparsity-aware," which means it efficiently handles missing or zero values in the data, a common issue in real-world datasets *citechentianqi*.

6.1.9 Mathematical Formulation of XGBoost

XGBoost employments decision trees base, but the key point of differences is that it sequentially builds and updates these trees employing a gradient boosting framework.
Regularized learning, for a given data set with n examples and m features, $D = \{(x_i, y_i)\}$, where $|D| = n$, $x_i \in \mathbb{R}^m$, and $y_i \in \mathbb{R}$, a tree ensemble model uses K additive functions to predict the output.

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}$$

where $F = \{f(x) = w_q(x)\}(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)$ The space of regression trees (also referred to as CART) is defined such that q represents the structure of each tree, mapping an example to its corresponding leaf index. T denotes the total number of leaves in the tree. Each function f_k is associated with an independent tree structure q and leaf weights w. Unlike decision trees, each regression tree assigns a continuous score to each leaf, where w; represents the score on the i-th leaf. For a given example, the decision rules in the trees (determined by q) classify the example into the appropriate leaves, and the final prediction is calculated by summing the scores in the corresponding leaves (determined by w. To learn the set of functions that the model uses, we minimize the following regularized objective.

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2$$

Here, l is a differentiable convex loss function that measures the difference between the predicted j_i and the target $yi_)$. The second term, Ω, penalizes model complexity (i.e., the regression trees). This regularization smooths the learned weights to prevent overfitting.
The choice of loss function depends on the type of problem (classification or regression). XGBoost supports various loss functions that are appropriate for different types of tasks
For regression tasks, the squared error loss function is commonly used. It is defined as

$$l(y_i, \hat{y}_i) = (y_i - \hat{y}_i^2)$$

In binary classification tasks, the logistic loss is generally used, defined as

$$l(y_i, \hat{y}_i) = -[y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)]$$

In this case, $y_i \in \{0, 1\}$ is the true binary label and $\hat{y}_i$ is the predicted probability belonging to class 1.
For multiclass classification, XGBoost model utilizes the softmax function combined with multiclass log loss. It is defined as

$$l(y_i, \hat{y}_i) = -\sum_{k=1}^{K} y_{i,k} log(\hat{y}_i, k)$$

where K is the total number of classes, $y_{i,k}$ is the function that is 1, instance i belongs to class k, and 0 otherwise, $\hat{y}_{i,k}$ is the predicted probability that instance i belongs to class k The softmax function normalizes

the predicted values into probabilities, and the multiclass log loss penalizes incorrect predictions.

Gradient boosting consists of function as parameters and cannot be optimized applying the methods called Euclidean space. The model trains the data by adding f_t to minimize the objective below

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(y-1)}) + f_t(x_i)) + \Omega(f_t)$$

Furthermore, second-order approximation can be utilized for a quick optimizing the objective

$$\tilde{\mathcal{L}}^{(\sqcup)} = \sum_{i=1}^{n}[g_i(f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \Omega(f_t)$$

where $g_i = \partial_{\hat{y}^(t-1)} l(y_i, \hat{y}^{t-1})$ and $h_i = \partial^2_{y^{(t-1)}} l(y_i, \hat{y}(t-1))$ are the first and second order gradient statistics on the loss function. Then, we can define $I_j = \{i|q(x_i) = j\}$ which is instance of leaf j.

$$\tilde{\mathcal{L}}^{(t)} = \sum_{j=1}^{T}[(\sum_{i\in I} g_i)w_j + \frac{1}{2}(\sum_{i\in I} h_i + \lambda) + \gamma T]$$

and, compute the optimal weightw_j^* of leaf j by

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda}$$

$$\tilde{\mathcal{L}}^{(t)}(g) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i=I} g_i)^2}{\sum_{i\in I} h_i + \lambda} + \gamma T$$

This is the corresponding optimal value. It can be used for measuring the quality of a tree structure$q$, like the impurity. It is generally infeasible to enumerate all possible tree structures q. Instead, a greedy algorithm is employed, which begins with a single leaf and iteratively adds branches to the tree. Let I_L and I_R represent the instance sets of the left and right nodes after a split, with $I = I_L \cup I_R$, The reduction in loss after the split is then given by

$$L_{split} = \frac{1}{2}[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}] - \gamma$$

This formula is commonly used in practice to evaluate potential split candidates during the tree-building process.

6.1.10 Hyperparameter Tuning

Some of the most important hyperparameters include:

Learning Rate η : The learning rate or step size is to add weights by η at each boosting iteration. It allows training model to decrease the influence of each tree and leaves space for future trees. This can leads to the reduction of overfitting if learning rate is set smaller, but it requires more boosting rounds to converge.(Kavlakoglu and Russi, 2024).

Max Depth : It controls the model complexity to define how deep each tree can grow in the training stage. This refers to the deep of the root node to the leaf nodes. The increase of max depth can make the model more complex and lead to overfitting. (Kavlakoglu and Russi, 2024).

The number of estimator : It specifies the number of trees where each boosting round adds a new tree to the ensemble. This parameter is directly relevant to the model complexity and training time, also affecting the model generalization to unseen data. Patterns in the data can be captured by increasing n _estimators Moreover, adding trees should balance with learning rate, for example, high n_estimators and low learning rate.(Kavlakoglu and Russi, 2024).

Gamma : It controls the minimum amount of loss reduction which make a further split on a leaf node. XGBoost may stop learning early if this parameter is set at low value, leading to not finding the best solution. In the other way, high value allows the model trained longer which can lead to better solution, or probably risk of overfitting.(Kavlakoglu and Russi, 2024).

6.2 Light Gradient Boosting (L-GBM)

Light gradient boosting machine, commonly referred to as LightGBM orL-GBM, is an advanced and rapid version of the gradient boosting decision tree (GBDT) algorithm. It was created to tackle the computational hurdles associated with extensive datasets that have numerous features. LightGBM employs several cutting-edge methods, including gradient-based one-side sampling (GOSS) and exclusive feature bundling (EFB), which enhance both processing speed and memory efficiency while ensuring high levels of accuracy.(Ke et al., 2017) (see structure in Figure 13).

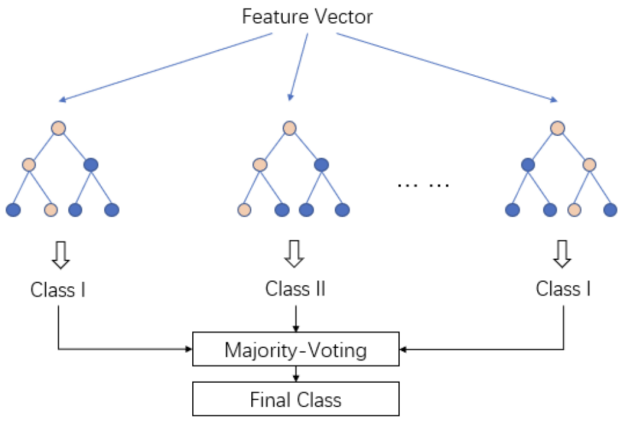


Figure 13: The structure of LightGBM model which is implementation of gradient boosting (Liu et al., 2022)

6.2.1 Key Innovations of LightGBM

What makes LightGBM different from other gradient boosting algorithms is data sampling and feature bundling. The purpose of this is to decrease the dataset size including the number of features. This leads to accelerate the training process.

Since all data instances need to be computed for information gain, this consumes the computation time based on the size of the dataset, especially large dataset. LightGBM has the solution to deal with this computation, retaining instances with large gradients called as gradient-based one-side sampling(GOSS (Ke et al., 2017).

Mathematically, GOSS works by adjusting the variance gain for the sampled data. The variance gain for a split is typically computed, let O be the training dataset on a fixed node of the decision tree. The variance gain of splitting feature j at point d for this node is defined as:

$$V_{j|O}(d) = \frac{1}{n_O}((\frac{\sum_{\{x_i \in O: x_{ij} \leq d\}} g_i}{n^j_{l|O}(d)})^2 + (\frac{\sum_{\{x_i \in O: x_{ij} > d\}} g_i}{n^j_{r|O}(d)})^2)$$

$n^j_{l|O}(d)$ and $n^j_{r|O}(d)$ are the number of instances to the left and right of the split, respectively (Ke et al., 2017). where $n_O = \sum I[x_i \in O], n^j_{l|O} = \sum I[x_i \in O : x_{ij} \leq d]$ and $n^j_{r|O}(d) = \sum I[x_i \in O : x_{ij} > d]$.

However, from the variance gain formulation above, instances are trained depending on absolute values of the gradients with descending order and next, it retains the $top - a \times 100\%$ instances applying the larger gradients to receive an instance subset A. Also, the set A^c with $1-a\times100\%$ instances with smaller gradients, followed by random sampling a subset$B$ with size $b \times |A^c|$. Then, the variance gain can be estimated as:

$$\tilde{V}_j(d) = \frac{1}{n}((\frac{\sum_{x_i \in A_l} g_i + \frac{1-a}{b} \sum_{x_i \in B_l} g_i}{n_l^j(d)})^2 + (\frac{\sum_{x_i \in A_r} g_i + \frac{1-a}{b} \sum_{x_i \in B_r} g_i}{n_r^j(d)})^2)$$

where $A_l = x_i \in A : x_{ij} \leq d, A_r = \{x_i \in A = x_{ij} \in A : x_{ij} > d\}, B_l = \{x_i \in B : x_{ij} \leq d\}, B_r = \{x_i \in B : x_{ij} > d\}$, and the normalization of gradients over B for A^c using the coefficient $\frac{1-a}{b}$(Ke et al., 2017).

Algorithm 2: Gradient-based One-Side Sampling

Input: I: training data, d: iterations
Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: $loss$: loss function, L: weak learner
models ← { }, fact ← $\frac{1-a}{b}$
topN ← a × len(I) , randN ← b × len(I)
for $i = 1$ ***to*** d **do**
  preds ← models.predict(I)
  g ← $loss$(I, preds), w ← {1,1,...}
  sorted ← GetSortedIndices(abs(g))
  topSet ← sorted[1:topN]
  randSet ← RandomPick(sorted[topN:len(I)], randN)
  usedSet ← topSet + randSet
  w[randSet] × = fact ▷ Assign weight $fact$ to the small gradient data.
  newModel ← L(I[usedSet], − g[usedSet], w[usedSet])
  models.append(newModel)

Figure 14: Gradient-based one-side sampling

LightGBM also addresses the problem of high-dimensional feature spaces through exclusive feature bundling (EFB). Many real-world datasets contain sparse features that rarely take nonzero values simultaneously. EFB takes advantage of this sparsity by bundling mutually exclusive features into a single feature, reducing the number of effective features and thereby lowering the computational cost (Ke et al., 2017).

To train gradient boosting decisition tree without the loss of accuracy, exclusive feature bundle changes the complexity of histogram building from $O(\#data \times \#features) to O\#data \times \#bundles)$ while $\#bundle << \#feature$. For this reason, training of GBDT can be speeded up with less impact on accuacy (Ke et al., 2017).

There are 2 key algorithms as shown Figure 18.

Input: F: features, K: max conflict count
Construct graph G
searchOrder ← G.sortByDegree()
bundles ← { }, bundlesConflict ← { }
for i ***in*** *searchOrder* **do**
  needNew ← True
  for $j = 1$ ***to*** *len(bundles)* **do**
    cnt ← ConflictCnt(bundles[j],F[i])
    if *cnt + bundlesConflict[i]* $\leq K$ **then**
      bundles[j].add(F[i]), needNew ← False
      break
  if *needNew* **then**
    Add $F[i]$ as a new bundle to *bundles*
Output: *bundles*

Figure 15: Greedy bundling

Input: $numData$: number of data
Input: F: One bundle of exclusive features
binRanges ← {0}, totalBin ← 0
for f ***in*** F **do**
  totalBin += f.numBin
  binRanges.append(totalBin)
newBin ← new Bin(numData)
for $i = 1$ ***to*** $numData$ **do**
  newBin[i] ← 0
  for $j = 1$ ***to*** *len(F)* **do**
    if *F[j].bin[i]* $\neq$ *0* **then**
      newBin[i] ← F[j].bin[i] + binRanges[j]
Output: $newBin$, $binRanges$

Figure 16: Merge exclusive features

Figure 17: 2 key methods used in LightGBM, greedy bundling (left) and merge exclusive features (right) methods for exclusive feature bundling.

6.2.2 Mathematical Formulation and Relevant Methods

LightGBM utilizes standard gradient boosting, same as XGBoost, the general form of the objective function is defined as

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

where $l(y_i, f_i)$ is the loss function that computes the different value of actual value and predicted value. The model complextity is in control by a regularization term f_k (T. Chen and Guestrin, 2016).

The regularization term is expressed as:

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2$$

where T represents the number of leaves in the tree, w denotes the weight of the leaf, and γ, λ are regularization parameters that control the depth and size of the trees (Ke et al., 2017).
Loss functions used in LightGBM depend on the tasks such as regression, binary classification or, multi-classification.
The mean squared error (MSE) is typically used in regression for LightGBM:

$$l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$$

where y_i is an actual value and $\hat{y}_i$ is a predicted value.(Microsoft, 2024).

For binary classification tasks, LightGBM employs the Log Loss (also known as binary cross-entropy)

$$l(y_i, \hat{y}_i) = -[y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)]$$

where $y \in \{0, 1\}$i represents the actual class label for each instance, y_i is the predicted probability of class 1.

This loss function penalizes the model when the predicted probability differs significantly from the actual label (Ke et al., 2017; Microsoft, 2024). For multiclass classification, LightGBM uses the softmax function combined with multiclass log loss.

$$l(y_i, \hat{y}_i) = -\sum_{k=1}^{K} y_{i,k} log(\hat{y}_i, k)$$

where K is the number of classes, y_i, k is 1 if instance i belongs to class k, and 0 otherwise, y_i, k is the predicted probability that instance i belongs to class k.

The multiclass log loss evaluates how well these probabilities match genuine class labels, whereas the softmax function guarantees that the predicted values are normalised to represent probabilities (Microsoft, 2024).

Histogram-based splitting has efficiency in terms of time-consuming reduction for training process and memory consumption. During training, it applies the algorithm to create feature histograms by grouping continuous feature values into discrete bins instead of searching for the split points on the feature (see Figure 14).

```
Input: I: training data, d: max depth
Input: m: feature dimension
nodeSet ← {0} ▷ tree nodes in current level
rowSet ← {{0, 1, 2, ...}} ▷ data indices in tree nodes
for i = 1 to d do
    for node in nodeSet do
        usedRows ← rowSet[node]
        for k = 1 to m do
            H ← new Histogram()
            ▷ Build histogram
            for j in usedRows do
                bin ← I.f[k][j].bin
                H[bin].y ← H[bin].y + I.y[j]
                H[bin].n ← H[bin].n + 1
            Find the best split on histogram H.
            ...
    Update rowSet and nodeSet according to the best
    split points.
    ...
```

Figure 18: Histogram-based (Ke et al., 2017)

Leaf-wise growth method is similar method used in algorithm like XGBoost, but it is different in terms of growing the tree. LightGBM splits a node into 2 leaves from left to right, which is not layer by layer(level-wise growth) like XGBoost. So, this helps the model maximize the information gain(see Figure 19).

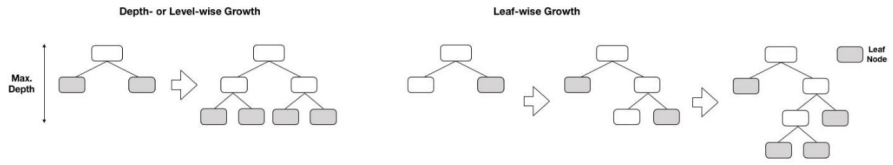


Figure 19: A graph shows how level-wise and leaf-wise growth strategies works Alshari et al., 2021.

6.2.3 Hyperparameter Tuning

The important hyperparameters of LightGBM includes (A. M. A. Ahmed, 2022)

The number of leaf(num_leaves) :It is used to control the tree complexity.

Maximum depth(max_depth) : It is used to control tree complexity working with num_leaves.

The number of iteration(num_iterations) : This parameter affects the training accuracy which requires tuning with learning rate. For example, low num_iterations and high learning rate.

Learning rate(learing_rate) : Determines the impact of each tree on the final prediction.

The minimum number of the data points in a leaf(min_data_in_leaf) : Increasing this parameter can avoid overffiting.

The maximum number of bins(max_bin) : Decreasing this parameter can avoid overffiting.

The percentage of sampled features(feature_fraction) : It is used to sample features for training each tree which can speed up training model.

The percentage of sampled training data points for each tree(bagging_fraction) : It is used to sample training data points to train each tree which can speed up training model.

Bagging frequency(bagging_freq) : It is used to re-sample trees.

Regularization terms(lambda_l1,_l2) : It is used to reduce overfitting along with min_gain_to_split.

The minimum loss reduction(min_gain_to_split) : Adding a split point on a tree to speed up training and solve an overfitting issue.

Early stopping round(early_stopping_round) : It is used to stop the training when validation metric is not improving, which can reduce excessive iterations.

6.3 Multilayer Perceptron (MLP)

The multilayer perceptron is one of neural network that is most frequently used. It is also known as feed-forward neural network which means the input is transmitted to output with one direction (Popescu et al., 2009). The main structure of MLP consists of 3 parts which are input layer, hidden layer, and output layer. The input layer is considered as a standalone as it is used for transmitting the input signals to the next layer, without any processing. The hidden layer, which lies between the input and output layers, plays a crucial role in the MLP's ability to model complex functions. Each neuron in the hidden layer transforms the inputs using a weighted linear summation followed by a non-linear activation function, such as the sigmoid, tanh, or ReLU function. Furthermore, the output layer receives the transformed inputs from the last hidden layer and converts them into output signals.

6.3.1 Architecture

The MLP consists of three main components:

Input layer: The first layer that recieves the input data or features. The number of neurons can be defined depending on the number of features in the dataset.

Hidden layer(s) : This layer is set in the middle parts of the neural network, it will perform computations applied by activation function such as sigmoid, tanh, or ReLU before sending to the output layer. The number of layers and neurons can be adjusted, and define the complex of model.

Output layer : The final output of the network, where the number of neurons corresponds to the number of output classes (for classification) or a single neuron (for regression) (Goodfellow et al., 2016).

Each neuron computes the weighted sum of its inputs, followed by the application of an activation function to introduce non-linearity (see architecture in Figure 20).

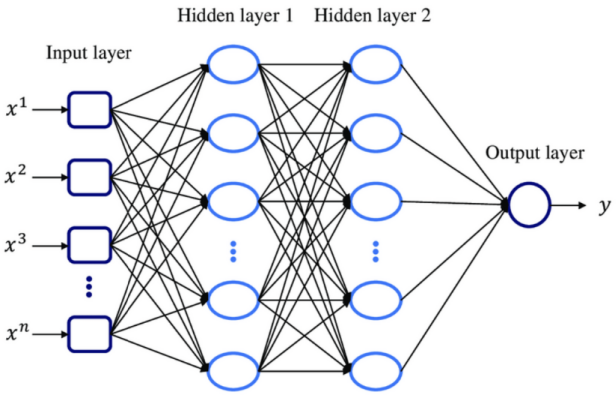


Figure 20: Example of multilayer perceptron architecture with 2 hidden layers (Sarraf Shirazi and Frigaard, 2021)

6.3.2 Mathematical Formulation

In a neural network, the forward pass method is used to compute the output of the network by transforming input data step-by-step through layers. The process can be described by applying a weight matrix and a bias term to an input vector, followed by an activation function to introduce non-linearity. The output of the network can be expressed as:

$$f(x; W, c, b) = \sigma(Wx + c) + b$$

or it can be derived as

$$f(x; z, b) = \sigma(z) + b$$

where W is the weight matrix that applies a linear transformation to the input vector x, c is the bias term that adds flexibity to the model, σ is the activation function applied to the result of the linear transformation, b is the bias term added to the final output to adjust the result.(Goodfellow et al., 2016; Popescu et al., 2009).

This formula represents the output of the model after applying both the linear transformation and the non-linear activation function. The weights W and bias c ensure that the network can learn complex relationships from the input data, while the final bias b adjusts the overall output.

Activation functions is used in MLP depending on tasks such as binary classification and multiclass classification.

Sigmoid function : this function is commonly used in binary classification.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Hyperbolic tangent (Tanh) :

$$tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

Rectified linear unit (ReLU) :

$$ReLU(z) = max(0, z)$$

Softmax function : used in multiclass classification

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$$

Loss functions are same as functions used in other models(XGBoost or LightGBM section) depend on the tasks but MLP typically uses average loss instead of individual loss.
The mean squared error (MSE):

$$l(y_i, \hat{y}_i) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Binary cross-entropy loss: Used for binary classification:

$$l(y_i, \hat{y}_i) = -\frac{1}{N} \sum_{i=1}^{N} [y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)]$$

Categorical cross-entropy loss: Used for multiclass classification.

$$l(y_i, \hat{y}_i) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_{i,k} log(\hat{y}_i, k)$$

where N is the number of samples.

6.3.3 Training Process

The training of an MLP typically involves using the backpropagation algorithm, which allows the network to learn by minimizing the error through adjusting the weights in the network. Backpropagation is crucial for calculating the gradient of the loss function with respect to the weights and biases.

Backpropagation is a method used for training artificial neural networks by propagating the error backward from the output layer to the input layer. It involves two phases: a forward pass and a backward pass.

It starts with a forward pass through the network to compute the prediction.

$$\begin{aligned} z_1 &= W_1 x + b_1, \\ o_1 &= \sigma(z_1), \\ z_2 &= W_2 o_1 + b_2, \\ o_2 &= \hat{y} = \sigma(z_2). \end{aligned}$$

where z is the weighted sum of inputs which are computed before applying activation functionσ, o is the output The goal of backpropagation is to compute the partial derivatives of the loss L with respect to all the parameters in the network(W_1, W_2, b_1, b_2) so that it will update the parameters during gradient descent. Assuming that the loss function is Mean squared error and using sigmoid acitivation function.

$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

Then, we can compute the derivative of the loss function with respect to z_2, applying the chain rule

$$\begin{aligned}
\frac{\partial L}{\partial o_2} &= \frac{\partial}{\partial o_2}(\frac{1}{2}(o_2 - y)^2) = o_2 - y = \hat{y} - y \\
\frac{\partial o_2}{z_2} &= \sigma^{'}(z_2) \\
\sigma^{'}(z_2) &= o_2(1 - o_2)
\end{aligned}$$

Use chain rule to compute the derivative of the loss function

$$\frac{\partial L}{\partial z_2} = \frac{\partial L}{\partial a_2} \cdot \frac{\partial o_2}{\partial z_2} = (\hat{y} - y) \cdot \sigma^{'}(z_2)$$

Given that $\delta_2 = \frac{\partial L}{\partial z_2}$, we can compute the gradient of the loss with respect to z_1

$$\begin{aligned}
\frac{\partial z_2}{o_1} &= W_2 \\
\frac{\partial L}{\partial z_1} &= \frac{\partial L}{\partial z_2} \cdot \frac{\partial z_2}{\partial o_1} \cdot \frac{\partial a_1}{\partial z_1} = \delta_2 W_2 \cdot \sigma^{'}(z_1)
\end{aligned}$$

Then, it can be defined as $\delta_1 = \frac{\partial L}{\partial z_1}$ Compute the gradients with respect to W_1 and b_1 for the parameters in the first layer(hidden layer)

$$\begin{aligned}
\frac{\partial L}{\partial W_1} &= \delta_1 x^T \\
\frac{\partial L}{\partial b_1} &= \delta_1
\end{aligned}$$

The final computation is to update the weight called as gradient descent. This will update the weight and bias for hidden layer W_1, b_1 and output layer W_2, b_2 with learning rate η.

$$\begin{aligned}
W_2 &= W_2 - \eta(\frac{\partial L}{\partial W_2}) \\
b_2 &= b_2 - \eta(\frac{\partial L}{\partial b_2}) \\
W_1 &= W_1 - \eta(\frac{\partial L}{\partial W_1}) \\
b_1 &= b_1 - \eta(\frac{\partial L}{\partial b_1})
\end{aligned}$$

6.3.4 Hyperparameter Tuning

Some of the key hyperparameters include

Learning rate (η) : This parameter works in the backpropagation stage to control the rate of model learning. It should be properly tuned, the model with a high learning rate will jump over the optimal solution, which means that the model is overshooting. Meanwhile, a low learning rate can lead to slow convergence so, the model will get stuck somewhere in local minima. (Bengio, 2012).

Number of hidden layers and neurons : Hidden layers and neurons have the significant impact on the model performance. For example, generally, to learn complex patterns, increasing the number of neurons or layers should be reasonable but this can lead to overfitting. (Goodfellow et al., 2016).

Activation function : Commonly, ReLU function is applied for hidden layers as it is more efficient than other activation functions, sigmoid and tanh to avoid vanishing gradient problem and saturated units , or dead(Nair and Hinton, 2010).

Batch size : In the training state, setting batch size will define the number of sample used in each gradient update. It is considered to be related to generalization performance, learning stability, and training time because it is used to control learning rate. (Hwang et al., 2024).

Number of epochs : Epoch is set to complete iteration through the entire training dataset in one cycle, it is said that there is no optimal number of epochs. When the model reaches the best result, before the error value starts going up, training model should be stopped at that epoch. (Afaq and Rao, 2020).

Regularization (L2 regularization, dropout) : Regularization techniques, such as L2 regularization (weight decay) and dropout, are used to prevent overfitting by penalizing large weights or randomly omitting neurons during training (Srivastava et al., 2014).

6.3.5 Optimization Techniques

Optimization refers to how we adjust the model's weights to minimize the loss function and improve performance. Several optimization techniques can be applied to neural networks

Stochastic gradient descent (SGD) : This algorithm is to compute the gradient by randomly picking example at each iteration. This process is a drastic simplification that can increase the speed of training process but it can cause noise in the loss function.(Bottou, 2010).

$$W = W - \eta \frac{\partial L_i}{\partial W}$$

where Li is the loss for a single training example x_i.

Mini-batch gradient descent : Mini-batch gradient descent is used to partition samples into small batch for data points at each iteration. Increasing the size of a mini-batch can reduce variance and also lead to a faster convergence (Li et al., 2014).

$$W = W - \eta \frac{1}{m} \sum_{i=1}^{m} \frac{\partial L_i}{\partial W}$$

where m is the size of the mini-batch.

Adam optimizer : Adam (Adaptive Moment Estimation) is a popular optimizer that computes adaptive learning rates for each parameter. It operates with sparse gradients, does not require a stationary objective, its stepsizes are roughly constrained by the stepsize hyperparameter, the magnitudes of parameter updates are invariant to gradient rescaling, and it naturally conducts a type of step size annealing (Kingma and Ba, 2014).

Momentum : Momentum is used to accelerate the convergence by accumulating a moving average of past gradients, helping to avoid oscillations in directions where the gradient varies frequently

$$v_{t+1} = \beta v_t - \eta \frac{\partial L}{\partial W}$$
$$W = W + v_t$$

where v_t is the velocity term and β is the momentum factor (Sutskever et al., 2013).

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Figure 21: Adam optimizer algorithm (Kingma and Ba, 2014)

6.4 Re-feature Selection

This process involves selecting a new set of features based on the assessment of ensemble models. The best ensemble model is used to determine feature importance, identifying the most relevant features. The final model aims to correctly predict various classes. To select new features, I considered top 20 feature importance, this may include features not relating to environmental factors or road conditions such as crash context, traffic control, and etc. However, the model performance and confusion matrix are the primary evaluation to consider metrics such as accuracy, precision, recall, and f1-score. Metrics for each class is essential, not overall performance, since overall of model performance may not reflect how well the model predicts specific features. For example, the model with high accuracy might be the best model but performs poorly for minority classes due to class imbalance, leading to overfitting to the majority class. Therefore, the analysis of confusion matrix is crucial to evaluate the prediction quality across all classes.

6.5 Re-training Model

This steps involves re-training the best ensemble model and the multilayer perceptron (MLP) with new features from the previous stage to optimize model performance. Hyperparameter configuration is used in this stage to avoid overfitting or underfitting. For example, for ensemble model and MLP, the main hyperparameters are adjusted such as the number of estimators, maximum depth, regularization and batch size, learining rate, regularization respectively. The goal of this step is to achieve reliable predictions that the model is not overffiting or underfitting to one class like majority class based on confusion matrix while accounting for the imbalanced class distribution.

7 Experimental Setup

All ensemble models were trained on the dataset with the default hyperparameter configuration, setting the number of estimators as 100. Additionally, to address the imbalanced class issue sampling techniques were applied and worked with a cross-validation technique to investigate overfitting and underfitting issues.

7.1 Sampling Methods

Regarding imbalanced classes, Resampling methods are significant techniques for dealing with imbalanced classes. The dataset with 4 imbalanced classes has a negative impact on model performance as the model cannot learn comprehensive patterns. In this case, I will compare 4 different sampling methods including SMOTE, undersampling, SMOTEENN, and SMOTETomek. These sampling techniques are applied into dataset before training models.

7.1.1 SMOTE

Oversampling involves adjusting unequal classes to balanced proportions by generating synthetic samples for minority classes (X. Wang et al., 2023) However, in this dataset, with four minority classes (fatal, minor injury, property damage only). Oversampling may result in increasing these classes to match the majority class (Serious injury). While SMOTE is effective for creating more representative training data, it also can cause overfitting. The risk of overffiting is caused by generating synthetic samples which are too similar to another data like a duplication. This could limit model generalization. However, for most of cases, SMOTE can effectively improve accuracy and overall performance (Chawla et al., 2002).

Algorithm *SMOTE*(T, N, k)
Input: Number of minority class samples T; Amount of SMOTE N%; Number of nearest neighbors k
Output: (N/100) * T synthetic minority class samples
(∗ *If N is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd.* ∗)
if $N < 100$
  then Randomize the T minority class samples
    $T = (N/100) * T$
    $N = 100$
endif
$N = (int)(N/100)$ (∗ *The amount of SMOTE is assumed to be in integral multiples of 100.* ∗)
k = Number of nearest neighbors
numattrs = Number of attributes
Sample[ ][ ]: array for original minority class samples
newindex: keeps a count of number of synthetic samples generated, initialized to 0
Synthetic[ ][ ]: array for synthetic samples
(∗ *Compute k nearest neighbors for each minority class sample only.* ∗)
for $i \leftarrow 1$ **to** T
    Compute k nearest neighbors for i, and save the indices in the *nnarray*
    Populate(N, i, *nnarray*)
endfor

Populate(N, i, *nnarray*) (∗ *Function to generate the synthetic samples.* ∗)
while $N \neq 0$
    Choose a random number between 1 and k, call it nn. This step chooses one of the k nearest neighbors of i.
    for $attr \leftarrow 1$ **to** *numattrs*
      Compute: $dif = Sample[nnarray[nn]][attr] - Sample[i][attr]$
      Compute: gap = random number between 0 and 1
      $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$
    endfor
    $newindex{+}{+}$
    $N = N - 1$
endwhile
return (∗ *End of Populate.* ∗)
End of Pseudo-Code.

Figure 22: SMOTE algorithm (Chawla et al., 2002)

7.1.2 Undersampling

Undersampling focuses on retaining data in the minority class while reducing the data in the majority class. Despite its effectiveness in addressing imbalanced classes, undersampling carries the risk of discarding important data from the majority class (Bach et al., 2019). Undersampling can be divieded into 2 groups. The first group is called as fixed undersampling method and the second group is cleaning-based method. In this project, I utilized fixed undersampling group which is random undersampling algorithm since it requires minimal computational resources and can quickly create a balanced dataset based on the size of dataset. Additionally, by reducing the size of the majority class, the model is less likely to become biased toward the majority class, which helps in improving performance metrics such as recall and F1-score for the minority class (Sole, 2023).

7.1.3 SMOTEENN

SMOTE-ENN combines SMOTE, a well-known oversampling technique, with Edited Nearest Neighbors (ENN) to reduce noise. The randomly selected instances are removed by the same class that do not have k nearest neighbors. This helps eliminate instances whose features belong to a different class than the majority of their neighbors (Sasada et al., 2020).

SMOTEENN is a combination between SMOTE and Edited Nearest Neighbor (ENN). The algorithm of ENN will find the nearest neighbor of the observation among the other observations in the dataset, then return the majority class from the K-nearest neighbor. If the class of the observation and the majority class from the observation's K-nearest neighbor is different, then the observation and its K-nearest neighbor are deleted from the dataset. However, firstly, SMOTE is applied to create synthetic data points for the minority class. It does this by generating new data between existing minority class samples, which helps balance the class distribution and reduce the model's bias toward the majority class. In crash severity data, noise and outliers were included due to the variability in how accidents occurred and were recorded. The ENN will help to remove those unexpected data to ensure that the model does not learn noise which causes misclassified dataBatista et al., 2004.

7.1.4 SMOTETomek

SMOTE-Tomek is a technique that integrates SMOTE with Tomek Links to reduce class overlap. Tomek Links identify pairs of closely situated instances from different classes. When a new instance is created through SMOTE, if it is closer to any instance in the pair than the instances are to each other, that original pair is eliminated. Essentially, Tomek Links help in removing boundary instances that could belong to multiple classes. This process effectively reduces instances that might cause overlap by considering the adjacency of instances near class boundaries (Sasada et al., 2020).

Tomek links belong to one of undersampling methods (Sole, 2023) that will find nearest neighbor using a distance metric (commonly Euclidean distance). For example, If two instances are in different classes and are each other's nearest neighbors, they form a Tomek Link. The resulting dataset after Tomek undersampling contains fewer instances, with an emphasis on removing those instances that were near the decision boundary (Appaji, 2024). In crash severity prediciton, there might be overlapping between certain classes, such as minor injury and serious injury. SMOTE-Tomek helps address this by removing boundary instances that could be contributing to class ambiguity. By doing so, it ensures that the classes are more distinctly separated, improving model performance. Furthermore, it is similar to SMOTEEN in terms of cleaning the data to remove noisy and borderline instances. This is particularly advantageous when dealing with large datasets, like the one used for crash severity prediction, as it ensures a more streamlined and effective approach to balancing the data and improving model accuracy.

7.2 Cross-Validation

A 5-fold cross-validation is utilized to validate the ensemble model which helps the risk assessment of overffiting or underfitting. This divides the dataset into 5 equal subset, where 4 of them is a training set and the remaining one is validation set. Moreover, the data leakage might happen during training model with sampling methods. To prevent this issue, sampling techniques (e.g. SMOTE, undersampling, SMOTEENN, and SMOTETomek) is correctly applied only on the training data within each fold, ensuring that validation set is not touched by any data augmentation techniques (Y. Wang, 2022). This approach maintains the integrity of the validation process, giving a more accurate estimate of the model's ability to generalize to unseen data. Cross-validation also provides a more robust way to evaluate model performance across different subsets of the data and helps determine the optimal configuration of hyperparameters, guiding further tuning if significant performance gaps are observed between the training and validation sets.

8 Model Evaluation

8.1 Confusion Matrix

A confusion matrix is a table that summarizes the performance of a classification model by comparing the predicted labels with the actual labels. It helps to assess how well the model performs across different classes and provides insights into misclassifications. For a classification task with n classes, the confusion matrix C is represented as (Krüger, 2016):

$$C = \begin{bmatrix} C_{1,1} & C_{1,2} & \dots & C_{1,n} \\ C_{2,1} & C_{2,2} & \dots & C_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ C_{n,1} & C_{n,2} & \dots & C_{n,n} \end{bmatrix}$$

where $C_{i,j}$ denotes the number of instances where the true class is i and the predicted class is j. The diagonal elements $C_{i,i}$ represent correct predictions, while the off-diagonal elements correspond to misclassifications.

8.1.1 Evaluation Metrics from the Confusion Matrix

Using the confusion matrix, we can derive several key metrics for evaluating the performance of the model, including accuracy, precision, recall, and F1-score.

Accuracy: Accuracy measures the proportion of correctly classified instances out of the total instances.

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} C_{i,i}}{\sum_{i=1}^{n} \sum_{j=1}^{n} C_{i,j}}$$

Precision: Precision is the proportion of true positive predictions for a specific class out of all instances predicted to be in that class.

$$\text{Precision}_i = \frac{C_{i,i}}{\sum_{j=1}^{n} C_{j,i}}$$

Recall (Sensitivity): Recall, also known as sensitivity or true positive rate, is the proportion of true positive predictions out of all actual instances in that class.

$$\text{Recall}_i = \frac{C_{i,i}}{\sum_{j=1}^{n} C_{i,j}}$$

F1-Score: The F1-score is the harmonic mean of precision and recall, providing a single metric to balance both.

$$\text{F1-Score}_i = \frac{2 \times \text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$$

I decided to use various metrics to evaluate the performance of my multiclass classification model because relying on accuracy alone wouldn't have given a clear picture. Accuracy was misleading, especially when I dealt with imbalanced classes, where one class had significantly more examples than others. Metrics like precision, recall, and F1-score were better for assessing the model's effectiveness for each class. Precision showed how well the model identified actual positives without too many false positives, while recall indicated how many actual positives it successfully captured. Since these metrics sometimes conflicted, I used the F1-score to balance them, focusing more on the F1-score and recall, as these metrics helped to understand if the model performed well even on less common but important classes, like predicting severe injuries in crash data.

9 Result and Analysis

From the previous section, I evaluated model performance and considered confusion matrix to compare trained models with different conditions to find the best model. This section will introduce every steps including before and after class combination.

9.1 Model Performance Comparison

The dataset were trained by ensemble models, Random forest, XGBoost, and LGBM without hyperparameters tuning and sampling techniques.

Table 8: Performance of baseline models before combining classes

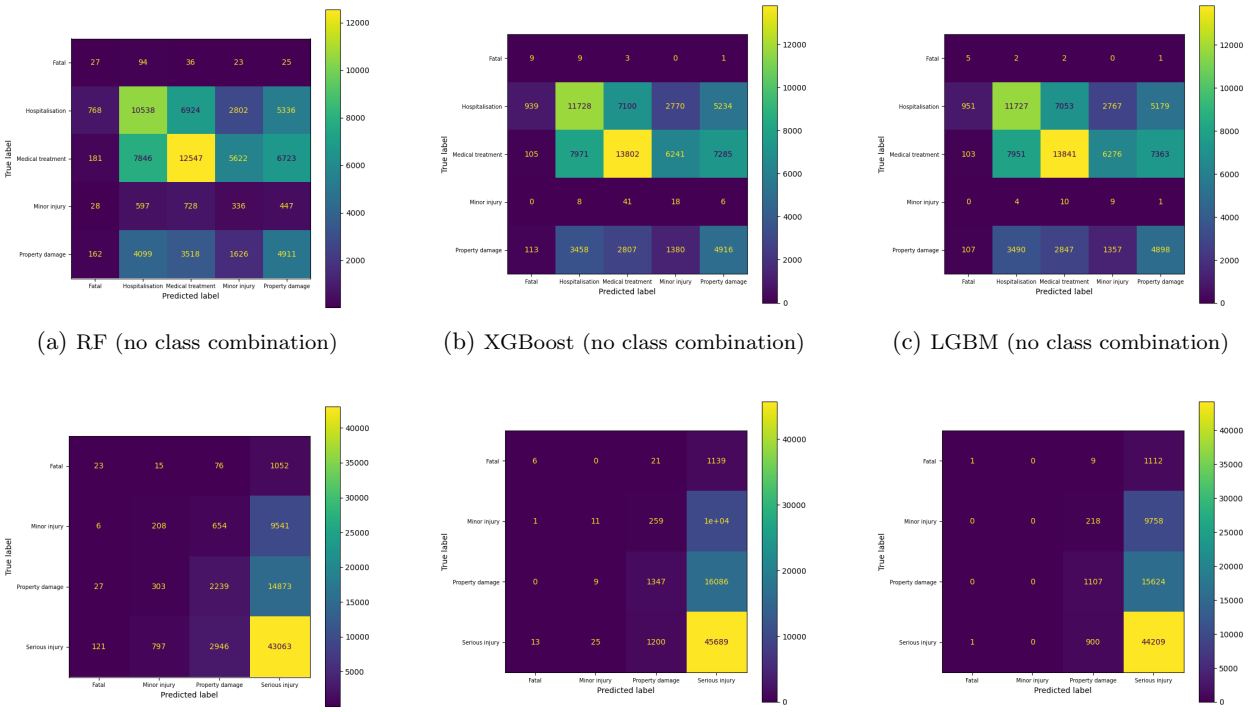
Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Random Forest	0.37	0.34	0.37	0.35
XGBoost	0.40	0.38	0.40	0.36
LGBM	0.40	0.40	0.40	0.36

Table 9: Performance of baseline models after combining classes

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Random Forest	0.60	0.50	0.60	0.51
XGBoost	0.62	0.54	0.62	0.51
LGBM	0.62	0.55	0.62	0.50

The result shows the different performance before and after combining class (hospitalization, medical treatment). Based on this result, baseline models with class combination outperformed models without class combination. However, these results could achieve higher accuracy since the model could be overfitting to predict towards the majority class. Therefore, in this case, it needed to identify the confusion matrix that the model did not correctly predict only majority class (see Figure 23).

According to the confusion matrix (Figure 23), I decided to focus on ensemble models with class combination. Ensembles models using class combination achieved higher accuracy, this indicated that overall performance. Models without class combination could try to predict various classses but it resulted in lower accuracy across specific metrics. The issue of class imbalance can be addressed by applying sampling techniques such as SMOTE, undersampling, SMOTEENN, or SMOTETomek. Although the combined-class models were more overfitting in predicting majority class, this could be improved when applying sampling techniques, generating synthetic data or removing redundant samples. This could improve the model's ability to generalize across all classes.



(a) RF (no class combination) (b) XGBoost (no class combination) (c) LGBM (no class combination)

(d) RF (with class combination) (e) XGBoost (with class combination) (f) LGBM (with class combination)

Figure 23: Confusion matrix for Random Forest, XGBoost, and LGBM models before and after class combination. The top row shows results without class combination, while the bottom row shows results with combined classes.

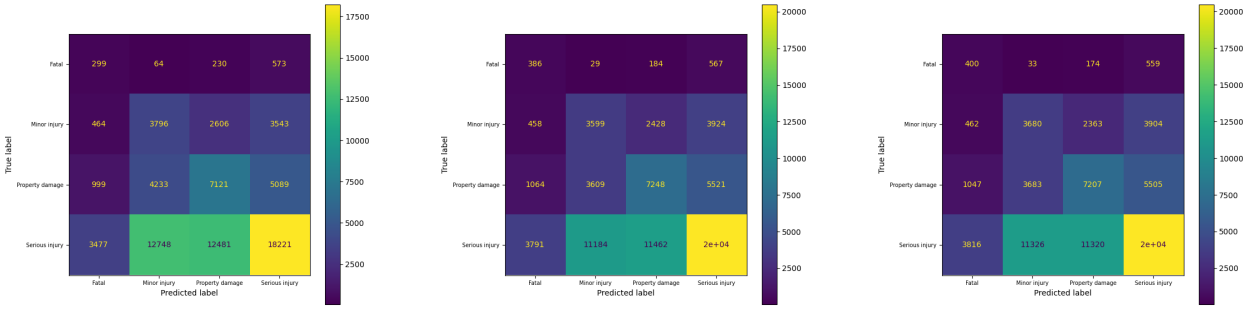
9.2 Model Performance with Different Sampling Methods

9.2.1 SMOTE Results

The classification report and confusion matrices after applying SMOTE to balance classes, L-GBM performance outperformed random forest model but its result slightly better than XGBoost (see in Figure 24).

Table 10: Classification report for models with SMOTE

Model	Class	Accuracy	Precision	Recall	F1-Score
Random forest	Fatal	0.26	0.06	0.26	0.09
	Minor Injury	0.36	0.18	0.36	0.24
	Property Damage	0.41	0.32	0.41	0.36
	Serious Injury	0.39	0.66	0.39	0.49
XGBoost	Fatal	0.33	0.07	0.33	0.11
	Minor injury	0.35	0.20	0.35	0.25
	Property damge	0.43	0.34	0.42	0.37
	Serious injury	0.44	0.67	0.64	0.53
L-GBM	Fatal	0.34	0.07	0.34	0.12
	Minor injury	0.35	0.20	0.35	0.25
	Property damage	0.41	0.34	0.41	0.37
	Serious injury	0.44	0.67	0.44	0.53



(a) RF with SMOTE (b) XGBoost with SMOTE (c) LGBM with SMOTE

Figure 24: Confusion matrix for Random Forest, LGBM, and XGBoost models with SMOTE applied to address class imbalance.

Random forest struggled with precision and recall in minority class (fatal, minor injury) showing lower values than other two classes with precision as low as 0.06 and 0.18 respectively. The F1-score also shows 0.09 for fatal and 0.24 for minor injury. This can ensure that this model struggled to predict minority class. For majority class predictions, the model could perform well on particular property damage and serious injury class, but the imbalance issue significantly affects its predictions across all classes. This suggests an imbalance in the dataset where one class dominates or the model is biased towards one class and almost always predicts one class (likely the majority class).

The XGBoost model demonstrates enhancement better than Random forest. All metrics show higher values than random forest. It achieves higher precision (0.07) and F1-score (0.11) for the Fatal class, which, while still low, is slightly better than RF. The model demonstrates more balanced metrics for other classes, especially serious injury, with an F1-score of 0.53. Moreover, the confusion matrix also reveals the fewer misclassification, showing higher accuracy, which could suggests that improved generalization compared to the random forest model.

Similarly, the LGBM model demonstrates a slight improvement compared to RF and a different prediction compared to XGBoost, better predicting fatal and minor injury classes. Each metrics achieved slightly higher scores than XGBoost, except for property damage. Although most predicted classes were improved, it still struggled with fatal cases , which are critically underrepresented, achieving an F1-score of only 0.12. This could indicates that the model can further benefit from tuning hyperparameters.

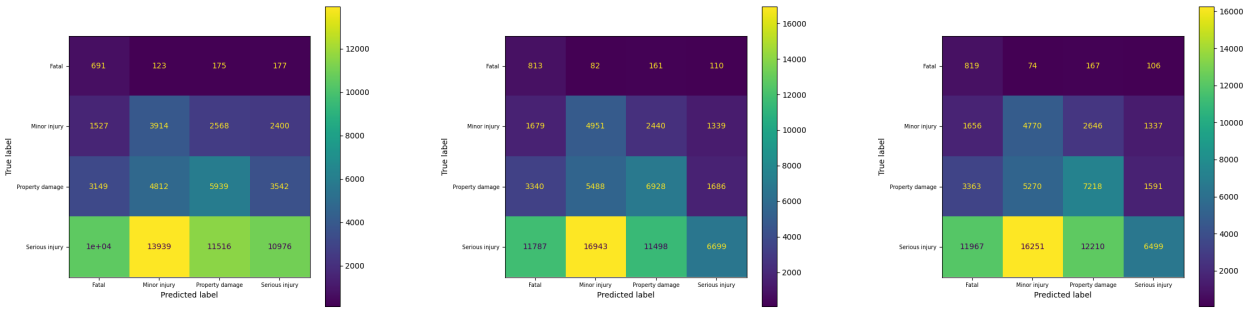
Even though, all models could have limitations for minority class predictions like fatal and minor injury, LGBM offered the most balanced performance across the evaluated metrics. Therefore, in this comparison, the model can be preferred model with the best-performing model as it can handle various classes more effectively.

9.2.2 Undersampling Results

Three models were trained with undersampling technique, for this experiment, XGBoost shows the most balanced metrics. Although undersampling decreased the number of majority class to balance with the least number of minority class, three models had a trade-off of predicted classes between majority and minority class (see in Figure 25).

Table 11: Classification report for models with undersampling

Model	Class	Accuracy	Precision	Recall	F1-Score
Random forest	Fatal	0.59	0.04	0.59	0.08
	Minor injury	0.38	0.17	0.38	0.24
	Property damage	0.34	0.29	0.34	0.32
	Serious injury	0.23	0.64	0.23	0.34
XGBoost	Fatal	0.70	0.05	0.70	0.09
	Minor injury	0.48	0.18	0.48	0.26
	Property damage	0.40	0.33	0.40	0.36
	Serious injury	0.14	0.68	0.14	0.24
L-GBM	Fatal	0.70	0.05	0.70	0.09
	Minor injury	0.46	0.18	0.46	0.26
	Property damage	0.41	0.32	0.41	0.36
	Serious injury	0.14	0.68	0.14	0.23



(a) RF with undersampling (b) XGBoost with undersampling (c) LGBM with undersampling

Figure 25: Confusion matrix for Random Forest, LGBM, and XGBoost models with undersampling applied to address class imbalance.

For fatal class, the random forest model reached an accuracy (0.59), this can be seen in confusion matrix. The model obtained a recall of 0.59, indicating that many cases were correctly predicted. In comparison to precision and f1-score, they were in low scores, 0.04 and 0.08 respectively. This could suggest that many of fatal class still was misclassified as non-fatal cases. In terms of serious injury class, it shows misclassification of predicting this class with precision (0.64) and only recall (0.23). The random forest model still struggled in predicting minority class showing the trade-off between precision and recall.

The XGBoost could be seen that it offered a more balanced performance compared to random forest. It acheived an accuracy of 0.70 for predicted fatal class with a slight improvement of precision (0.05), resulting in f1-score of 0.09. This could indicate that the model can manage trade-off between false positives and false negatives better than random forest although misclassification still occurred. The consistency of predicted minor injury and property damage classes show f1-score at 0.26 and 0.36 respectively. However, the model obtained lower scores of f1-score in predicting serious injury class showing that distinguishing this class remains challenging.

The LGBM model provides a performance profile similar to XGBoost but with slightly lower accuracy, achieving 0.70. However, its f1-scores for most classes are marginally lower than XGBoost's, with minor injury and serious injury classes achieving f1-scores of 0.26 and 0.23, respectively. The confusion matrix reveals that LGBM has a tendency to misclassify Serious Injury cases as minor injury or property damage, limiting its generalization capability.

Therefore, for this sampling method, none of the three models performed reliably enough, as all showed inconsistencies across class predictions. Despite XGBoost's relatively balanced metrics, the models faced challenges in distinguishing between critical classes, such as Fatal and Serious Injury. Further improvements could be made by experimenting with hybrid sampling methods such as SMOTEENN and SMOTETomek.

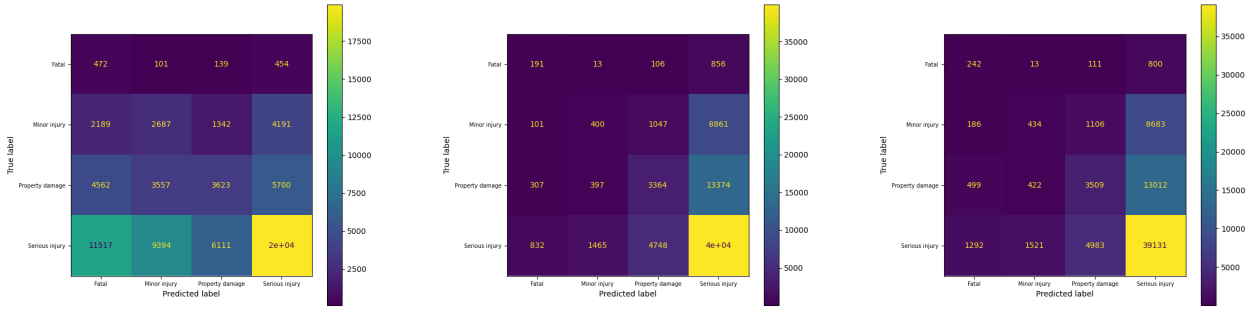
9.2.3 SMOTEENN Results

Based on the results, all three models still struggle to accurately predict the minority class. I found that the SMOTEENN method is more likely to cause overfitting, focusing on the characteristics of the majority class (see in Figure 26).

Table 12: Classification report for models with SMOTEENN

Model	**Class**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Random forest	Fatal	0.40	0.03	0.40	0.05
	Minor injury	0.26	0.17	0.26	0.21
	Property damage	0.21	0.32	0.21	0.25
	Serious injury	0.42	0.66	0.42	0.52
XGBoost	Fatal	0.16	0.13	0.16	0.15
	Minor injury	0.04	0.18	0.04	0.06
	Property damage	0.19	0.36	0.19	0.25
	Serious injury	0.85	0.63	0.85	0.73
L-GBM	Fatal	0.21	0.11	0.21	0.14
	Minor injury	0.04	0.18	0.04	0.07
	Property damage	0.20	0.36	0.20	0.26
	Serious injury	0.83	0.63	0.83	0.72

For the random forest model, I noticed that the model achieved an accuracy of 0.42, precision of 0.66, and an F1-score of 0.52 for serious injury. However, it struggled with other classes, suggesting that the model was overfitting in predicting majority class. However, the model's precision and f1-score were quite low, meaning that the model extremely misclassified fatal cases. I concluded that the Random Forest model faced difficulties in balancing precision and recall, particularly when predicting minority classes like Fatal.



(a) RF with SMOTEENN (b) XGBoost with SMOTEENN (c) LGBM with SMOTEENN

Figure 26: Confusion matrix for Random Forest, LGBM, and XGBoost models with SMOTEENN applied to address class imbalance.

For the XGBoost model, I found that the model predicted well for serious injury and its result was much better than random forest. In terms of confusion matrix, it shows that this prediction was still unreliable since the model could not learn in predicting other minority class, only focusing on majority class prediction. As the result, the minor injury class was poorly predicted, a precision (0.18) and an f1-score (0.06).

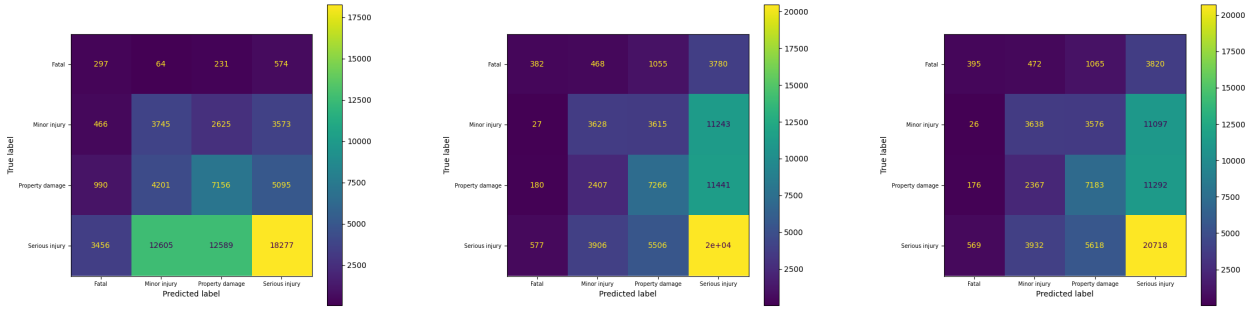
When analyzing the LGBM model, I found that it performed similarly to XGBoost, especially for Serious Injury cases, achieving an accuracy of 0.83, precision of 0.63, and an F1-score of 0.72. However, like XGBoost, LGBM struggled with Fatal cases, achieving a precision of only 0.11 and an F1-score of 0.14. I noticed that the confusion matrix revealed misclassification of Fatal and Minor Injury cases, limiting the model's ability to generalize across critical classes effectively. Therefore, I concluded that the high rate of misclassification for other classes reduced its overall reliability.

9.2.4 SMOTETomek Results

In training models with SMOTETomek sampling, each model responded differently to the balancing method, models were less overfitting in predicting towards majority class than applying SMOTEENN (see in Figure 27), except for random forest. I observed that models endeavored to predict various classes, especially XGBoost and LGBM. This improvements indicates that both models were more consistent in predicting across all classes, capturing their distinct characteristics.

Table 13: Classification report for models with SMOTEtomek

Model	Class	Accuracy	Precision	Recall	F1-Score
Random forest	Fatal	0.25	0.06	0.25	0.09
	Minor injury	0.36	0.18	0.36	0.24
	Property damage	0.41	0.32	0.41	0.36
	Serious injury	0.39	0.66	0.39	0.49
XGBoost	Fatal	0.33	0.33	0.07	0.11
	Minor injury	0.35	0.35	0.20	0.25
	Property damage	0.42	0.42	0.34	0.38
	Serious injury	0.79	0.44	0.67	0.53
L-GBM	Fatal	0.34	0.34	0.07	0.11
	Minor injury	0.35	0.35	0.20	0.25
	Property damage	0.41	0.41	0.34	0.37
	Serious injury	0.44	0.44	0.67	0.53



(a) RF with SMOTETomek (b) XGBoost with SMOTETomek (c) LGBM with SMOTETomek

Figure 27: Confusion matrix for Random Forest, XGBoost, and LGBM models with SMOTETomek applied to address class imbalance.

The random forest model struggled to maintain balanced performance across the severity classes. For the fatal class, it achieved an accuracy (0.25), with recall (0.25) but a low precision (0.06) and f1-score (0.09). When comparing to the previous sampling, random forest was more likely to misclassify across all classes, although property damage class achieved better prediction with 0.41 accuracy. Nonetheless, f1-score in table between property damage (0.36) and serious injury(0.49) indicates that the model was inconsistent across classes and less effective with SMOTETomek.

In contrast, XGBoost model became a more balanced compared to SMOTEENN result. It reached an accuracy of 0.79 with a precision of 0.44, a recall of 0.67, and an F1-score of 0.53 for serious injury class. Predicting minority class was challenging, showing that the fatal was less predicted accurately. Nonetheless, minor injury and property damage, with f1-scores of 0.25 and 0.38, respectively were improved, suggesting that it demonstrated better consistency.

The LGBM model delivered similar performance to XGBoost, especially for the serious injury class, where it achieved an accuracy of 0.44 and an f1-score of 0.53. However, it also struggled with the fatal class, obtaining an f1-score of only 0.11. Like XGBoost, LGBM faced difficulties distinguishing between minor injury and property damage, achieving f1-scores of 0.25 and 0.37, respectively. Although LGBM's overall performance was solid for serious injuries, its high misclassification rate for the other classes, particularly fatal cases, limits its reliability.

From all previous experiments with different sampling techniques, I observed that each model exhibited varying performance and none were able to correctly predict all classes. There were consistent limitations, particularly with the trade-offs in predicting between different severity levels. Each sampling methods originated trade-offs influencing the model performance. As the results, SMOTE introduced noise which reduced precision for classes like fatal injury. In contrast, undersampling method focused on minority classes leading to the decrease of accuracy for the majority classes, this was inconsistent predictions. Moreover, SMOTEENN and SMOTETomek had a similar performance, focusing on predicting majority classes. For SMOTEENN, it had better balance for recall and precision, still harder to distinguish between overlapping classes due to aggressive cleaning noisy samples. In the same way, SMOTETomek enhanced class separation and improved serious injury predictions but struggled with subtle class distinctions after removing borderline cases. Ultimately, LGBM with SMOTE stood out as the best-performing model in these experiments because it managed the trade-offs more effectively, offering higher accuracy across critical classes like fatal injury and serious injury. I decided to prioritize the model's ability to accurately predict critical injury classes, specifically fatal and serious injury, since these categories carry the most importance in real-world applications.

9.3 Feature Importance

Based on the feature importance analysis shown in Figure 30, I re-selected new features including the top 20 features extremely affecting the model performance and other features not shown in the chart (see in Figure 28).

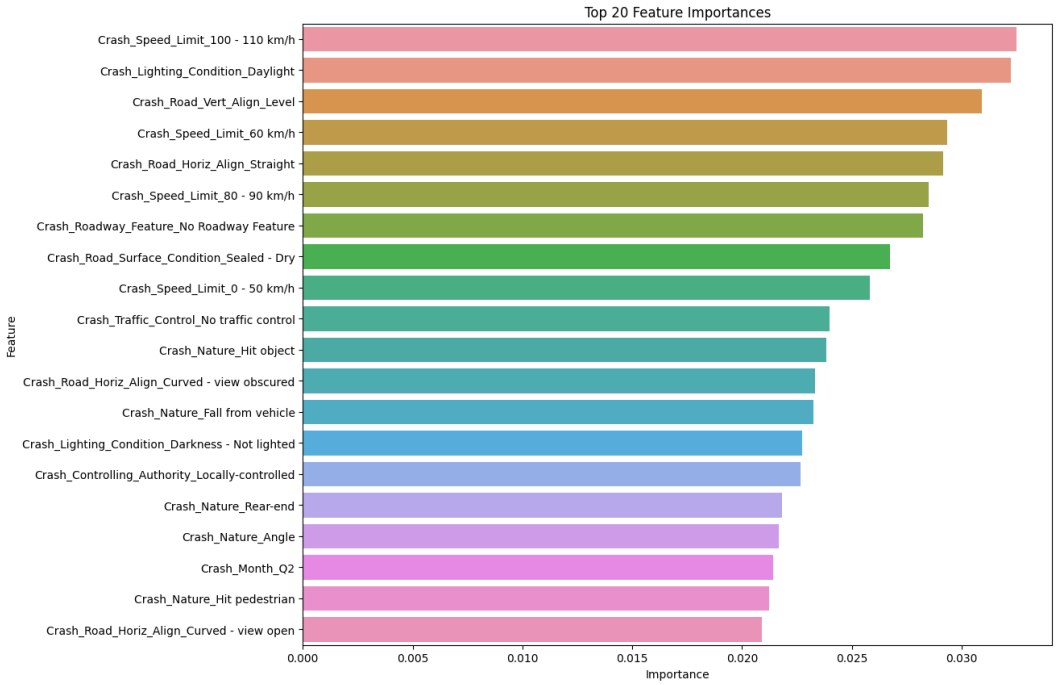


Figure 28: The Top 20 Feature Importance for L-GBM

The selection aligned with the project's objective: understanding the impact of weather and road conditions on crash severity. In addition to the 20 ranking qualities, I included additional aspects to explore a more thorough study and analysis (see in Table 14). These extra features—such as lighting, smoke/dust, rainy conditions, and others—were essential for determining the final accident severity. By doing this, the model's predictive power was increased, and knowledge of the interactions between road and environmental elements and crash outcomes was acquired.

Table 14: Re-Selected Crash Features

Feature	**Name**
Crash Road Surface Condition	Sealed - Dry Sealed - Wet Unsealed - Dry Unsealed - Wet
Crash Atmospheric Condition	Clear Fog Raining Smoke/Dust
Crash Lighting Condition	Darkness - Lighted Darkness - Not lighted Dawn/Dusk Daylight

Feature	Name
Crash Road Alignment (Horizontal)	Curved - view obscured Curved - view open Straight
Crash Road Alignment (Vertical)	Crest Dip Grade Level
Crash Speed Limit	100 - 110 km/h 60 km/h 80 - 90 km/h 0 - 50 km/h
Crash Roadway Feature	No Roadway Feature
Crash Traffic Control	No traffic control
Crash Nature	Hit object Fall from vehicle Rear-end Angle Hit pedestrian
Crash Controlling Authority	Locally-controlled
Crash Month	Q2

This expanded feature set aims to improve the model's predictive capability and provide deeper insights into how environmental and road factors shape crash outcomes. This approach also helps avoid missing potentially significant variables that could refine safety interventions.

By concentrating on new features chosen based on the feature importance, it could ensure that the model generalized to unseen data. In the next step, the L-GBM and neural network model (MLP) were retrained. By reducing the features, the model can reduce training time, especially MLP, and learn effectively without being overwhelmed by irrelevant or redundant data. This aimed to reduce data complexity and enhance the model's predictive accuracy across all classes, particularly for critical injuries like fatal and serious injury.

9.4 Re-trained L-GBM Performance

The re-trained LGBM showed the better performance in predicting minority classes, essentially critical class like fatal, minor injury and property damage. This indicates that selected features helped the model focus on more relevant attributes. As the result, it is clearly seen that accuracy of predicting those class was improved, suggesting that the model's ability could understand the feature difference to distinguish these classes. However, there are some misclassification of majority and minority classes, which seems to cause trade-off between serious injury class and other classes by decreasing the correct predicted serious injury but it could accurate the predictions of other classes. The model may not predict well due to overlapping characteristics between classes and the complexity of balancing performance across multiple classes (see in Figure 29).

Table 15: Classification report for L-GBM

Model	Class	Accuracy	Precision	Recall	F1-Score
LGBM	Fatal	0.45	0.05	0.46	0.09
	Minor injury	0.39	0.19	0.39	0.26
	Property damage	0.46	0.31	0.46	0.37
	Serious injury	0.27	0.69	0.27	0.38

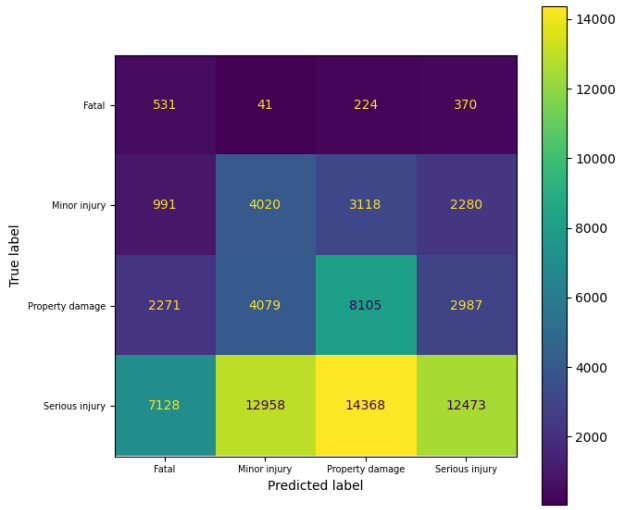


Figure 29: Confusion Matrix for Re-trained L-GBM

9.5 Multilayer Perceptron Performance

The neural network (MLP) was developed using three different architectures to explore various model complexities and evaluate their impact on performance. The architectures included configurations with different hidden layer sizes: a single hidden layer of 64 units, two hidden layers of 64 and 32 units, and a deeper architecture with three hidden layers of 128, 64, and 32 units. These choices allowed for a comprehensive assessment of how layer depth and size affect the model's predictive capabilities. The input and output layers were fixed at 32 (to match the input features) and 4 (for the target classes), respectively.

In terms of hyperparameters, the MLP models were tuned with learning rates of 1e-4, 1e-5, and 1e-6, a dropout rate of 0.7 to reduce overfitting, and various batch sizes (128, 256, and 512). Each model was trained for 30 epochs to allow sufficient learning while monitoring for signs of overfitting.

After experimentation, the optimal configuration was identified as a model with

Hidden layer: 64 units

Learning rate: 1e-6

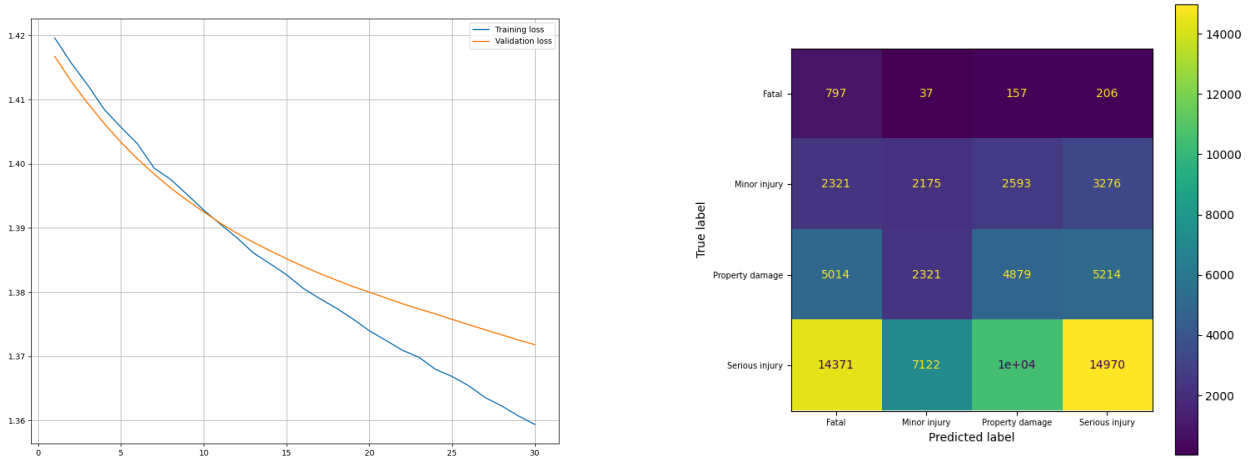
Batch size: 512

Epoch: 30

This configuration showed the best balance between training and validation loss, although overfitting was still observed across all architectures. The model's training and validation losses demonstrated a continuous downward trend, but the gap between them suggested that the model was learning the training data more effectively than it was generalizing to unseen validation data(see in Figure 30).

Table 16: Classification report for MLP

Model	Class	Accuracy	Precision	Recall	F1-Score
MLP	Fatal	0.67	0.04	0.67	0.07
	Minor injury	0.21	0.21	0.20	0.07
	Property damage	0.28	0.27	0.28	0.27
	Serious injury	0.32	0.63	0.32	0.42



(a) Training Loss and Validation Loss

(b) Confusion Matrix for MLP

Figure 30: Comparison of training/validation loss and the confusion matrix for the best MLP model configuration with 1e-6 learning rate, 512 batch size, and 30 epochs.

9.6 SHAP Analysis

SHAP assigns each feature an importance value, or SHAP value, that indicates how much it either increases or decreases the predicted outcome. This approach helps understand the influence of features across different classes or outcomes (Lundberg and Lee, 2017). The SHAP value plots provide insights into the influence of various features on the model's predictions across different crash severity classes. By interpreting these plots, we can identify which features most strongly impact the model's outputs, with a particular focus on weather and road conditions

9.6.1 Interpretation of SHAP Values

In SHAP plots, features are represented using a color gradient where red indicates high feature values and blue indicates low feature values. For instance, in a feature like Crash Speed Limit 100 - 110 km/h, red points signify crashes occurring at high-speed limits, while blue points denote lower speeds (Cooper, 2021). The + (positive) SHAP values indicate features that push the model's prediction toward a higher outcome class, such as a more severe injury or fatality. In contrast, - (negative) SHAP values pull the prediction toward a lower outcome class, like minor injury or property damage. For example, if Crash Atmospheric Condition Raining has a positive SHAP value for serious injuries, it means that rain increases the likelihood of a crash being classified as severe (Cooper, 2021). Each dot in the SHAP summary plot represents an individual data

point's impact on the model's prediction. The distribution and clustering of points help visualize the overall influence of a feature on the prediction (Cooper, 2021).

Based on the SHAP summary plot (see in Figure 31), across all classes, features such as 100 - 110 speed limit, road surface with sealed dry condition, and straight road alignment appear frequently as influential factors. This suggests that the condition and layout of the road, as well as speed limits, are significant determinants in predicting crash severity. High-speed limits generally increase the likelihood of severe outcomes, while road alignment affects how crashes are distributed across severity levels.

9.6.2 Class-Specific Feature Impacts

Fatal injury: For this class, Crash Nature Rear end and Crash Speed Limit 100 - 110 km/h have a high impact. Interestingly, the presence of Crash Traffic Control No traffic control also stands out, indicating that a lack of traffic regulation increases the risk of fatal crashes. Weather-related features, such as Crash Atmospheric Condition Raining, show that poor weather conditions (e.g., rain) exacerbate crash outcomes, likely due to reduced visibility and traction.

Minor injury: The SHAP values for minor injuries highlight the importance of road features, like Crash Road Horiz Align Curved - view open, and the presence of traffic control. Speed limits play a critical role as well, but weather conditions seem to have a relatively lesser impact compared to the fatal injury class.

Property damage: This class is significantly influenced by features like Crash Nature Hit pedestrian and Crash Atmospheric Condition Raining. The impact of Crash Road Surface Condition Sealed - Dry indicates that even in seemingly safe road conditions, crashes resulting in property damage are common. Weather conditions, especially rain, and the type of crash (e.g., hitting a pedestrian) contribute notably to the severity.

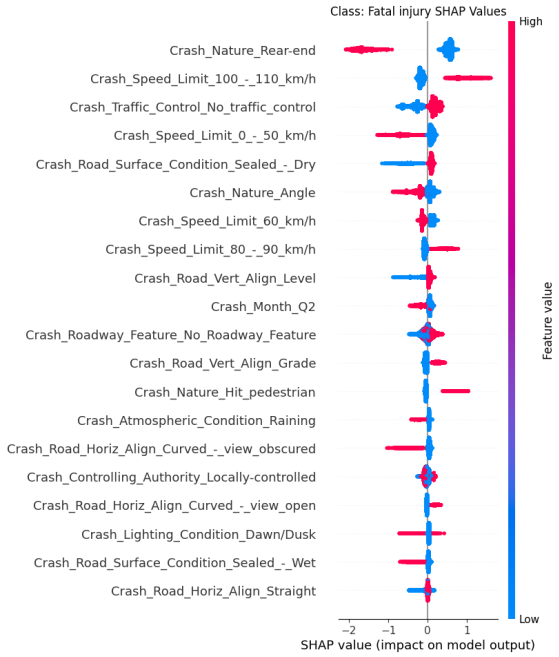
Serious injury: The SHAP plot for serious injuries shows high impacts from Crash Atmospheric Condition Raining and Crash Speed Limit 100 - 110 km/h. Road surface conditions, both dry and wet, are crucial here, indicating that adverse weather and road conditions increase the likelihood of serious injuries.

9.6.3 Impact of Weather and Road Conditions

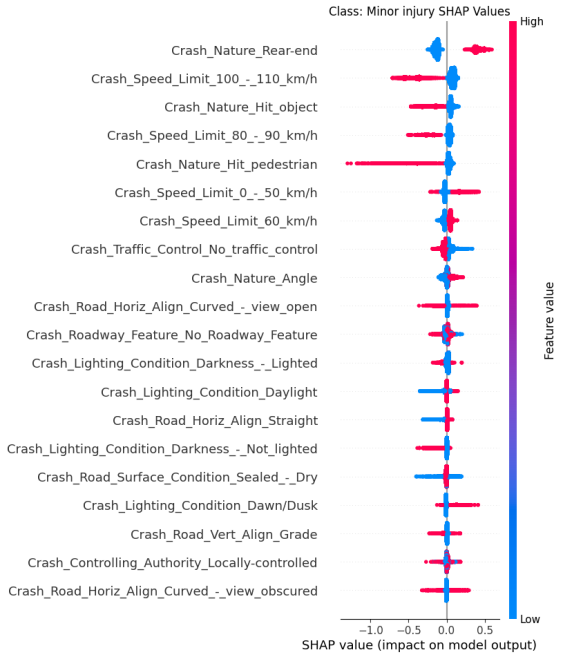
Weather conditions: Crash Atmospheric Condition Raining appears prominently, especially in classes like property damage and serious injury. This underscores the hazardous nature of wet weather, which increases the risk of crashes leading to significant outcomes. Rain likely reduces visibility, increases braking distances, and causes slipperier surfaces, all contributing to higher severity.

Road surface conditions: Features such as Crash Road Surface Condition Sealed - Dry and Crash Road Surface Condition Sealed - Wet are frequently highlighted. While dry conditions reduce risk, the presence of wet or poorly maintained surfaces can drastically worsen crash outcomes, especially when combined with high-speed travel or complex road alignments.

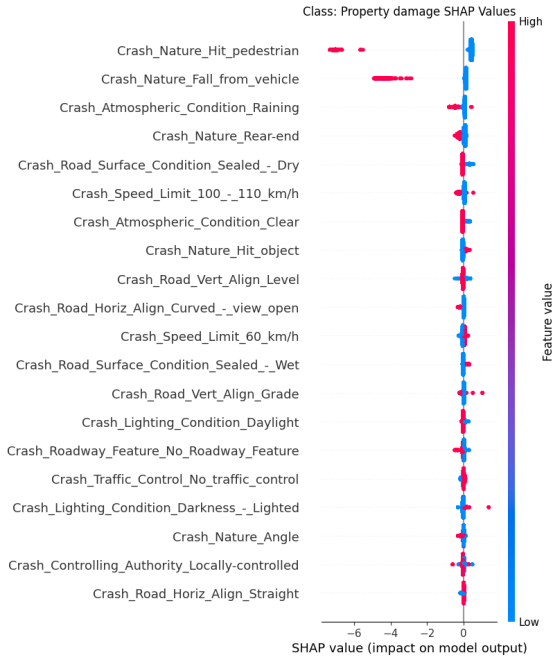
Road alignment: Features like Crash Road Horiz Align Straight and Crash Road Horiz Align Curved also significantly influence predictions. Curved road alignments tend to be associated with a higher risk of severe crashes, as they require more driver awareness and control.



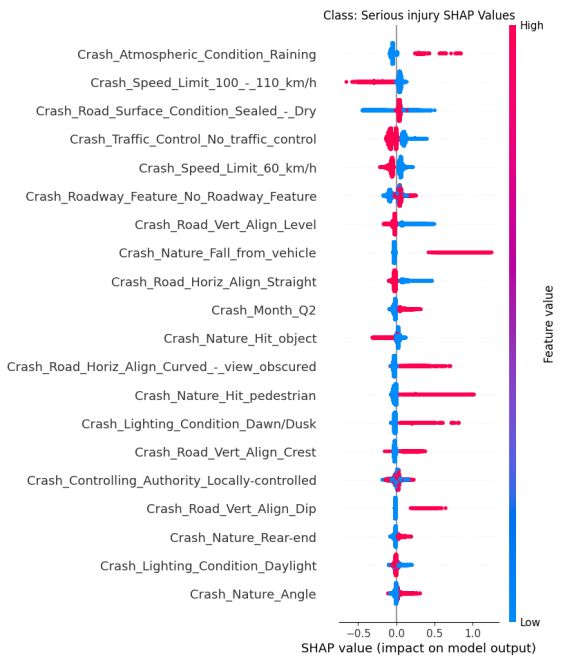
(a) SHAP values for Fatal Injury Class



(b) SHAP values for Minor Injury Class



(c) SHAP values for Property Damage Class



(d) SHAP values for Serious Injury Class

Figure 31: SHAP Summary Plot: SHAP value distributions for different crash severity classes. Each plot shows the impact of features on the model's output, with high and low feature values highlighted.

Based on the SHAP analysis, when focusing on critical crash classes like fatal and serious injuries, high-speed environments, such as roads with speed limits of 100-110 km/h, greatly increase the risk. This is especially true when combined with the lack of traffic control and specific road features. These factors make crashes more severe, likely because drivers have less time to react, and accidents at high speeds are more deadly. Dry, sealed road surfaces combined with straight or sloped roads also make it harder to recover from a crash, raising the chances of fatal injuries. On the other hand, weather conditions like rain increase the likelihood of serious but non-fatal injuries. Wet and slippery roads make driving more dangerous, though crashes in these conditions may not always be deadly. Poor lighting, such as at dawn or dusk, also adds to the risk by reducing visibility. This analysis shows that while speed and road design are key factors in fatal crashes, worse weather and poor visibility are more important for serious, non-fatal injuries. Together, these factors show how different conditions affect the severity of crash outcomes.

10 Limitation

The main limitations of this study were related to the data imbalance, which made it difficult for the models to perform well. Although sampling techniques were applied, like SMOTE, undersampling, and hybrid methods (SMOTEENN and SMOTETomek), the models still had trouble accurately predicting the minority classes, especially the fatal and minor injury categories. This data imbalance led to more misclassifications in these important classes, even when using methods like ensemble models, suitable for classification tasks, and neural networks, having capability to capture complex data. Another challenge was that some models tended to overfit to the majority class. For instance, models like L-GBM and XGBoost achieved higher overall accuracy but didn't generalize well across all crash severity categories. Additionally, the combination of classes and sampling techniques did not fully address the challenge of class overlap and noise within the dataset, leading to inconsistencies in model performance. Finally, the results were also limited by the chosen features, as certain attributes might not have captured the full complexity of the relationships between weather, road conditions, and crash severity, which could have affected the model's predictive capabilities. Further research may be required to explore additional features or advanced balancing techniques to enhance model reliability.

11 Conclusion

In conclusion, I investigated how weather and road conditions affect the severity of road accidents using a multiclass classification method. The project focused on models like Random Forest, XGBoost, re-tuned LightGBM (L-GBM), and Multilayer Perceptron (MLP). The goal was to improve the prediction of crash severity in an imbalanced dataset by carefully using sampling methods and selecting the right features. My results showed that ensemble models, especially XGBoost and re-tuned L-GBM, generally outperformed Random Forest, achieving higher precision and recall. XGBoost provided balanced results across different severity classes, while re-tuned L-GBM gave more consistent performance after fine-tuning the settings. Although the MLP model also performed well, especially critical class like fatal, it had difficulty predicting across all classes.

A key limitation of my research was the persistent difficulty in accurately predicting minority classes, like "Fatal" and "Minor Injury," even after applying various sampling techniques. SHAP analysis provided insights into feature importance, confirming that road surface conditions and atmospheric factors were the most influential predictors. However, the need for additional features, such as road surface temperature, humidity levels, wind speed, and precipitation intensity, may enhance the model's predictive power. To address the imbalance issue, future work could explore advanced balancing methods, such as class-weighted loss functions or deeper neural networks. Including these additional features, identified as significant through SHAP analysis, would help the model make more nuanced predictions, particularly for distinguishing between different severity levels. Overall, my investigation highlighted the importance of carefully selecting models, features, and methods that balance performance across all classes, ultimately supporting my decision to use ensemble models with appropriate sampling techniques.

12 Recommendations

I have three main suggestions to improve crash severity predictions related to weather and road conditions. First, I recommend adding more detailed environmental data, like road surface temperature, humidity, wind speed, and how strong the rain is. These details are important for understanding how weather affects road safety (Malin et al., 2019). Including these features would likely make the model more accurate and better match real-world driving conditions.

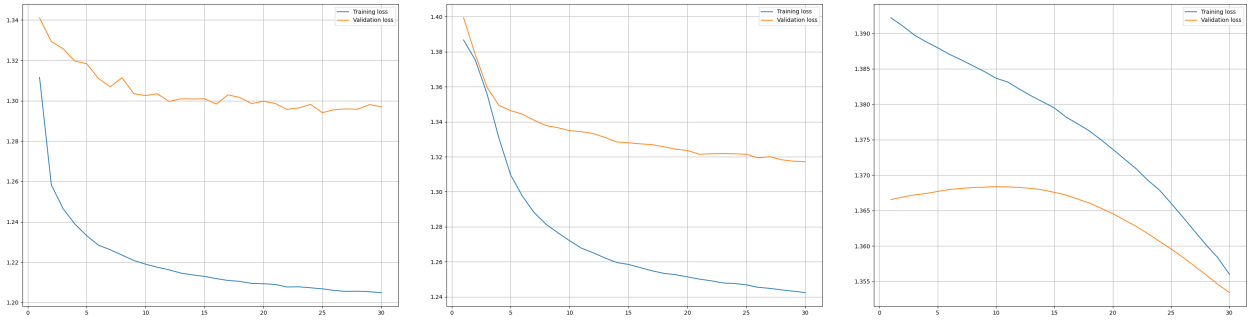
Second, I recommend trying advanced techniques to handle class imbalance, like class-weighted loss functions or ensemble methods specifically designed for imbalanced data. Dealing with the uneven distribution of severity classes is essential for improving the model's ability to predict less common outcomes, like "Fatal" and "Minor Injury". For example, investigating other sampling methods to handle data imbalance such as ADASYN(Adaptive Synthetic Sampling) and BorderlineSMOTE which are extensions of SMOTE (TurinTech, 2022). could also boost model performance.

Third, I suggest using SHAP analysis to explain how the model makes predictions and to identify which features are most important. However, this project did not explore selecting features based on SHAP values (Lundberg and Lee, 2017). To further enhance model performance, future research could explore additional types of SHAP plots and continue investigating which factors are the most critical.

Overall, this project shows how important it is to predict crash severity accurately. By adding more detailed environmental features and using advanced analysis methods, we could make valuable contributions to public safety, helping plan better interventions and use resources more effectively.

13 Appendix

Appendix A: Training and Validation Loss Curves for (64, 32) MLP Models with SMOTE Sampling, Batch Size 256 and Learning Rates of 10^{-4}, 10^{-5}, and 10^{-6}



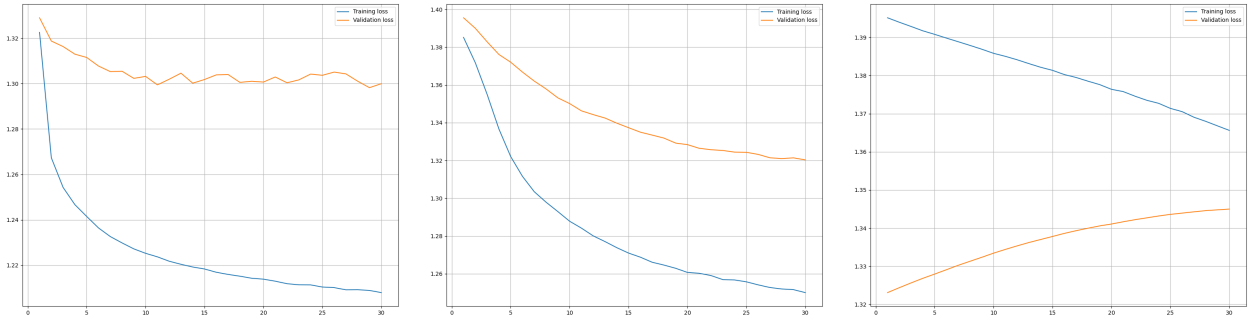
(a) Loss curves for learning rate 10^{-4}, batch size 256.

(b) Loss curves for learning rate 10^{-5}, batch size 256.

(c) Loss curves for learning rate 10^{-6}, batch size 256.

Figure 32: Training and validation loss curves for (64, 32) MLP models with SMOTE sampling: (a) shows the curves for learning rate 10^{-4}, (b) shows the curves for learning rate 10^{-5}, and (c) shows the curves for learning rate 10^{-6}, all using a batch size of 256.

Appendix B: Training and Validation Loss Curves for (64, 32) MLP Models with SMOTE Sampling, Batch Size 512 and Learning Rates of 10^{-4}, 10^{-5}, and 10^{-6}



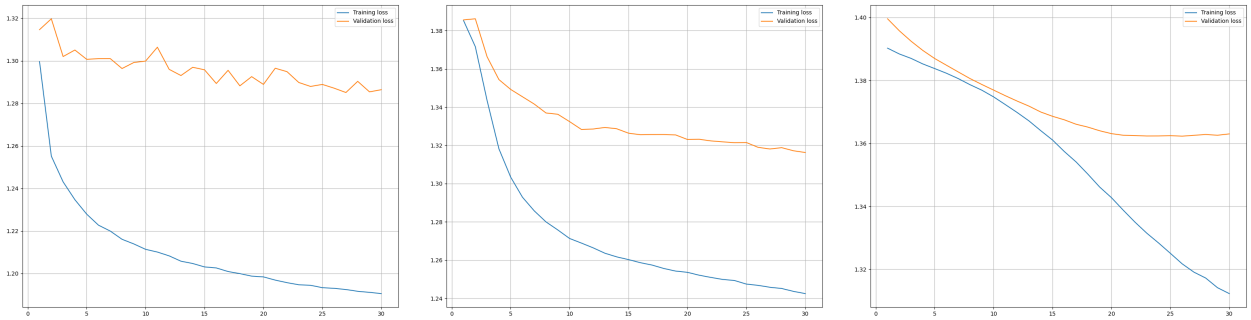
(a) Loss curves for learning rate 10^{-4}, batch size 512.

(b) Loss curves for learning rate 10^{-5}, batch size 512.

(c) Loss curves for learning rate 10^{-6}, batch size 512.

Figure 33: Training and validation loss curves for (64, 32) MLP models with SMOTE sampling: (a) shows the curves for learning rate 10^{-4}, (b) shows the curves for learning rate 10^{-5}, and (c) shows the curves for learning rate 10^{-6}, all using a batch size of 512.

Appendix C: Training and Validation Loss Curves for (128, 64, 32) MLP Models with SMOTE Sampling, Batch Size 256, and Learning Rates of 10^{-4}, 10^{-5}, and 10^{-6}



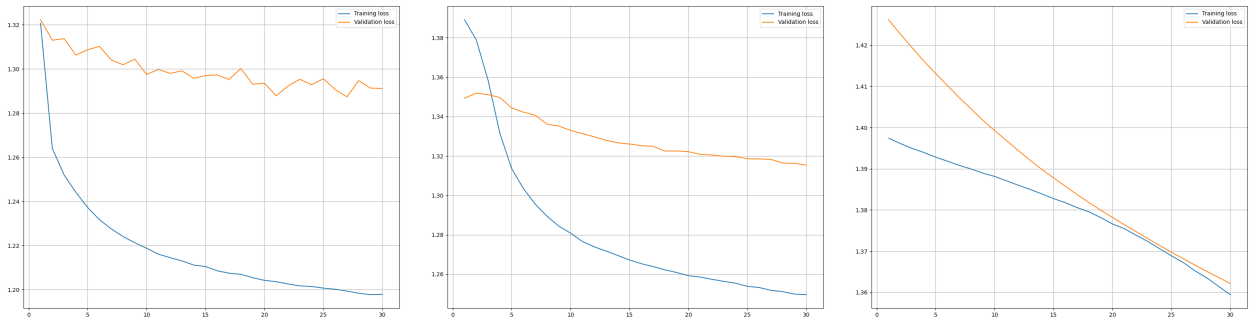
(a) (a) Loss curves for learning rate 10^{-4}, batch size 256.

(b) (b) Loss curves for learning rate 10^{-5}, batch size 256.

(c) (c) Loss curves for learning rate 10^{-6}, batch size 256.

Figure 34: Training and validation loss curves for (128, 64, 32) MLP models with SMOTE sampling: (a) shows the curves for learning rate 10^{-4}, (b) shows the curves for learning rate 10^{-5}, and (c) shows the curves for learning rate 10^{-6}, all using a batch size of 256.

Appendix D: Training and Validation Loss Curves for (128, 64, 32) MLP Models with SMOTE Sampling, Batch Size 512, and Learning Rates of 10^{-4}, 10^{-5}, and 10^{-6}



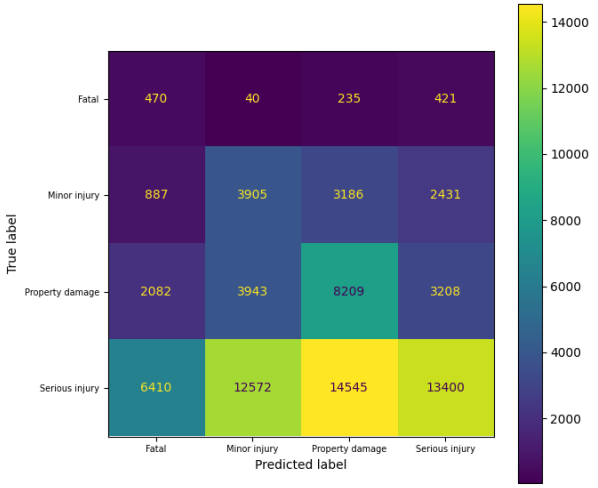
(a) Loss curves for learning rate 10^{-4}, batch size 512.

(b) Loss curves for learning rate 10^{-5}, batch size 512.

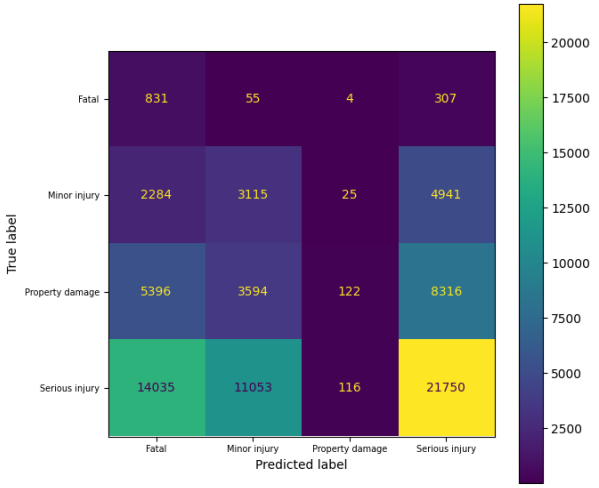
(c) Loss curves for learning rate 10^{-6}, batch size 512.

Figure 35: Training and validation loss curves for (128, 64, 32) MLP models with SMOTE sampling: (a) shows the curves for learning rate 10^{-4}, (b) shows the curves for learning rate 10^{-5}, and (c) shows the curves for learning rate 10^{-6}, all using a batch size of 512.

Appendix E: Confusion Matrix for (128, 64, 32) MLP Models with SMOTE Sampling, Batch Size 512, and Learning Rates of 10^{-6}, and L-GBM Tuning Hyperparameters with Learning Rate 0.3, n_estimator 100, and Subsample 0.6



(a) Confusion matrix for L-GBM tuning hyperparameters with learning rate 0.3, n_estimator 100, and subsample 0.6.



(b) Confusion matrix for (128, 64, 32) MLP models with SMOTE sampling and learning rate 10^{-6}.

Figure 36: (a) shows the tuned hyperparameters for the LightGBM (L-GBM) model, and (b) displays the confusion matrix for the (128, 64, 32) MLP model trained with SMOTE sampling, a batch size of 512, and a learning rate of 10^{-6}.

References

Afaq, S., & Rao, S. (2020). Significance of epochs on training a neural network. *International Journal of Scientific & Technology Research*, *9*, 485–488. https://api.semanticscholar.org/CorpusID:225647672

Ahmed, A. M. A. (2022). Lightgbm cheat sheet [Accessed: 2024-10-10]. *Medium*. https://awadrahman.medium.com/lightgbm-cheat-sheet-ddde43f9eaf1

Ahmed, S., Hossain, A., Ray, S. K., Bhuiyan, M. M. I., & Sabuj, S. R. (2023). A study on road accident prediction and contributing factors using explainable machine learning models: Analysis and performance. *Transportation Research Interdisciplinary Perspectives*. https://doi.org/10.1016/j.trip.2023.100814

Ahmed, S., Raza, B., Hussain, L., Aldweesh, A., Omar, A., Khan, M. S., Tageldin, E., & Nadim, M. (2023). The deep learning resnet101 and ensemble xgboost algorithm with hyperparameters optimization accurately predict the lung cancer. *Applied Artificial Intelligence*, *37*. https://doi.org/10.1080/08839514.2023.2166222

Alshari, H., Saleh, A., & Odabas, A. (2021). Comparison of gradient boosting decision tree algorithms for cpu performance. *Erciyes Tip Dergisi*, 157–168.

Appaji, N. (2024). Balancing act: Mastering imbalanced data with smote and tomek-link strategies. *Medium*. https://medium.com/@niranjan.appaji/balancing-act-mastering-imbalanced-data-with-smote-and-tomek-link-strategies-289f39597122

Bach, M., Werner, A., & Palt, M. (2019). The proposal of undersampling method for learning from imbalanced datasets. *Procedia Computer Science*, *159*, 125–134. https://doi.org/10.1016/j.procs.2019.09.167

Batista, G., Prati, R., & Monard, M.-C. (2004). A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations*, *6*, 20–29. https://doi.org/10.1145/1007730.1007735

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. https://arxiv.org/abs/1206.5533

Birfir, S., Elalouf, A., & Rosenbloom, T. (2023). Building machine-learning models for reducing the severity of bicyclist road traffic injuries. *Transportation Engineering*, *12*, 100179. https://doi.org/https://doi.org/10.1016/j.treng.2023.100179

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Y. Lechevallier & G. Saporta (Eds.), *Proceedings of compstat'2010* (pp. 177–186). Physica-Verlag HD.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140. https://doi.org/10.1007/BF00058655

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Çeven, S., & Albayrak, A. (2024). Traffic accident severity prediction with ensemble learning methods. *Computers and Electrical Engineering*, *114*, 109101. https://doi.org/https://doi.org/10.1016/j.compeleceng.2024.109101

Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. (2002). Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, *16*, 321–357. https://doi.org/10.1613/jair.953

Chen, S., Saeed, T. U., & Labi, S. (2017). Impact of road-surface condition on rural highway safety: A multivariate random parameters negative binomial approach. *Analytic Methods in Accident Research*, *16*, 75–89. https://doi.org/10.1016/j.amar.2017.09.001

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. https://doi.org/10.1145/2939672.2939785

Cooper, A. (2021, November). Explaining machine learning models: A non-technical guide to interpreting shap analyses. https://www.aidancooper.co.uk/a-non-technical-guide-to-interpreting-shap-analyses/

Friedman, J. (2000). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*. https://doi.org/10.1214/aos/1013203451

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Hermans, E., WETS, G., & Van den Bossche, F. (2007). Frequency and severity of belgian road traffic accidents studied by state-space methods. *Journal of Transportation and Statistics*, *9*.

Hwang, J.-S., Lee, S.-S., Gil, J.-W., & Lee, C.-K. (2024). Determination of optimal batch size of deep learning models with time series data. *Sustainability*, *16*(14). https://doi.org/10.3390/su16145936

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning with applications in r*. Springer.

Kavlakoglu, E., & Russi, E. (2024). What is xgboost? [Accessed: 2024-10-09]. https://www.ibm.com/topics/xgboost

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 3149–3157.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, *abs/1412.6980*. https://api.semanticscholar.org/CorpusID:6628106

Krüger, F. (2016). Activity, context, and plan recognition with computational causal behaviour models. *ResearchGate*. https://www.researchgate.net/publication/314116591_Activity_Context_and_Plan_Recognition_with_Computational_Causal_Behaviour_Models

Li, M., Zhang, T., Chen, Y., & Smola, A. J. (2014). Efficient mini-batch training for stochastic optimization. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 661–670. https://doi.org/10.1145/2623330.2623612

Liaw, A., & Wiener, M. (2001). Classification and regression by randomforest. *Forest*, *23*.

Liu, Y., Yong, S., He, C., Wang, X., Bao, Z., Xie, J., & Xing, Z. (2022). An earthquake forecast model based on multi-station pca algorithm. *Applied Sciences*, *12*(7), 3311. https://doi.org/10.3390/app12073311

Louppe, G. (2015). Understanding random forests: From theory to practice. https://arxiv.org/abs/1407.7502

Lundberg, S., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. https://arxiv.org/abs/1705.07874

Malin, F., Norros, I., & Innamaa, S. (2019). Accident risk of road and weather conditions on different road types. *Accident Analysis Prevention*, *122*, 181–188. https://doi.org/https://doi.org/10.1016/j.aap.2018.10.014

Mazarei, A., Sousa, R., Mendes-Moreira, J., Molchanov, S., & Ferreira, H. M. (2024). Online boxplot derived outlier detection. *International Journal of Data Science and Analytics*. https://doi.org/10.1007/s41060-024-00559-0

Microsoft. (2024). Lightgbm documentation [Accessed: 2024-10-10]. https://lightgbm.readthedocs.io

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. *International Conference on Machine Learning*. https://api.semanticscholar.org/CorpusID:15539264

Popescu, M.-C., Balas, V., Perescu-Popescu, L., & Mastorakis, N. (2009). Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, *8*.

Queensland Government. (2024). Queensland government open data policy statement [Accessed: 28 October 2024]. https://www.data.qld.gov.au/_resources/documents/qld-data-policy-statement.pdf

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. https://doi.org/10.1007/BF00116251

Sarraf Shirazi, A., & Frigaard, I. (2021). Slurrynet: Predicting critical velocities and frictional pressure drops in oilfield suspension flows. *Energies*, *14*(5), 1263. https://doi.org/10.3390/en14051263

Sasada, T., Liu, Z., Baba, T., Hatano, K., & Kimura, Y. (2020). A resampling method for imbalanced datasets considering noise and overlap [Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 24th International Conference KES2020]. *Procedia Computer Science*, *176*, 420–429. https://doi.org/https://doi.org/10.1016/j.procs.2020.08.043

Sole. (2023). The role of undersampling in tackling imbalanced datasets in machine learning. *Train in Data Blog*. https://www.blog.trainindata.com/undersampling-techniques-for-imbalanced-data/

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, *15*(1), 1929–1958.

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013, June). On the importance of initialization and momentum in deep learning [Presented on 17–19 June.]. In S. Dasgupta & D. McAllester (Eds.), *Proceedings of the 30th international conference on machine learning* (pp. 1139–1147, Vol. 28). PMLR. https://proceedings.mlr.press/v28/sutskever13.html

Tarwidi, D., Pudjaprasetya, S. R., Adytia, D., & Apri, M. (2023). An optimized xgboost-based machine learning method for predicting wave run-up on a sloping beach. *MethodsX*, *10*, 102119. https://doi.org/10.1016/j.mex.2023.102119

Transport & Main Roads, Q. G. (2024). Crash data from queensland roads [Accessed: 28 October 2024]. https://www.data.qld.gov.au/dataset/crash-data-from-queensland-roads

TurinTech. (2022, January). What is imbalanced data and how to handle it? [Accessed: 2024-11-05].

Vanitha, R., & Swedha, M. (2023). Prediction of road accidents using machine learning algorithms. *Middle East Journal of Applied Science & Technology*, *06*(02), 64–75. https://doi.org/10.46431/mejast.2023.6208

Wang, X., Li, Y., Zhang, J., Zhang, B., & Gong, H. (2023). An oversampling method based on differential evolution and natural neighbors. *Applied Soft Computing*, *149*, 110952. https://doi.org/10.1016/j.asoc.2023.110952

Wang, Y. (2022). Avoiding leakage in cross-validation when using smote. *Medium*. https://yijiew.medium.com/avoiding-leakage-in-cross-validation-when-using-smote-b63fdd3d159d

Zhou, X., Xu, X., Zhang, J., Wang, L., Wang, D., & Zhang, P. (2023). Fault diagnosis of silage harvester based on a modified random forest. *Information Processing in Agriculture*, *10*(3), 301–311. https://doi.org/https://doi.org/10.1016/j.inpa.2022.02.005

Zhu, F., Chen, X., & Li, G. (2022). Multi-classification assessment of personal credit risk based on stacking integration. *Procedia Computer Science*, *214*, 605–612. https://doi.org/10.1016/j.procs.2022.11.218