A Machine Learning Approach to Predict Traffic Accident Severity in Addis Ababa

Samuel Asres Geda

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree

of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

September, 2024

Adama, Ethiopia

A Machine Learning Approach to Predict Traffic Accident Severity in Addis Ababa

Samuel Asres Geda

Advisor: Feyissa Woyano Gobana (PhD)

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies
Adama Science and Technology University

September, 2024
Adama, Ethiopia

DECLARATION

I declare that this Master Thesis entitled “A Machine Learning Approach to Predict Traffic Accident Severity in Addis Ababa” is my own work and has not been submitted to any university for similar purpose. The references used in this thesis are duly recognized by proper citations.

_____	_____	_____
Signature	Name of student	Date

RECOMMENDATION

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled “A Machine Learning Approach to Predict Traffic Accident Severity in Addis Ababa” prepared under my guidance by Samuel Asres submitted in partial fulfillment of the requirements for the degree of Masters of Science in Computer Science and Engineering.

Therefore, I recommend the submission of the revised version of the thesis to the department following the applicable procedures.

Major Advisor	Signature	Date
Co-advisor	Signature	Date

APPROVAL SHEET

I, the advisor of the thesis entitled “A Machine Learning Approach to Predict Traffic Accident Severity in Addis Ababa” developed by Samuel Asres hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Major Advisor	Signature	Date
Co-advisor	Signature	Date

We, the undersigned, members of the Board of Examiners of the thesis Samuel Asres Geda have read and evaluated the thesis entitled “A Machine Learning Approach to Predict Traffic Accident Severity in Addis Ababa” and examined the candidate during the open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in Computer Science and Engineering.

Chairperson	Signature	Date
Internal Examiner	Signature	Date
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis are contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

Department Head	Signature	Date
School Dean	Signature	Date
Office of Postgraduate Studies, Dean	Signature	Date

ACKNOWLEDGMENT

I would like to begin by expressing my deepest gratitude to the Almighty God for granting me the strength, perseverance, and guidance to complete this thesis. My sincere thanks go to my esteemed advisor, Dr. Feyissa Weyano, for his limitless support and invaluable guidance throughout this journey. His expertise, patience, and encouragement were instrumental in shaping this thesis. My sincere thanks also for Dr. Getnet Yilma his instrumental review of my thesis and his expertise helped refine my work significantly. I am incredibly grateful to my wife for her unwavering love, understanding, and encouragement during this challenging yet rewarding endeavor. I also extend my heartfelt thanks to my children, Kineab, Meba, and Jonathan, for their love and patience as I dedicated time and effort to this project. Special thanks to my friend, Solomon Tesfaye, for informing me to this program and providing me with all the necessary information. My deepest appreciation goes to my classmates, Besufikad Eshetu and Beyen Awlachew, for facilitating my registration and communication with the university registrar, finance department, and other departments while I was in Addis Ababa. Their assistance saved me valuable time and resources. Finally, I would like to thank all those who have directly or indirectly supported me throughout this journey.

TABLE OF CONTENT

CHAPTER PAGE

LIST OF TABLES

TABLES PAGES

LIST OF FIGURES

FIGURES PAGES

LIST OF ABBREVIATIONS AND ACRONYMS

AA – Addis Ababa

AACA - Addis Ababa City Administration

AACRA – Addis Ababa City Road Authority

AACRTMA - Addis Ababa City Road Traffic Management Agency

AAPC– Addis Ababa Police Commission

AATB - Addis Ababa Transport Bureau,

Ada Boost - Adaptive Boosting

Bagging – Bootstrap Aggregation

CNN- Convolution Recurrent Neural Network

CTA – Central Statistics Agency

DRSs - Data Registry Systems

EU – European Union

EU-OSHA - European Agency for Safety and Health at Work

FPC – Federal Police Commission

GDP - Growth Domestic Product

GSRRS - Global Status Report of Road Safety

HICs - High-Income Countries

HIV /AIDS - HIV/AIDS-Human immune-Deficiency Syndrome,

KNN – K-Nearest Neighbors

LMIC- Low- And Middle-Income Countries

RF – Random Forest

RNN- Recurrent Neural Networks

RTIs- Road Traffic Injuries

SDG - Sustainable Development Goals

SVM - Support Vector Machine

TMA - Traffic Management Agency

UN- United Nation

USD – United States of America Dollar

WHO - World Health Organization

XGBoost – Extreme Gradient Boosting

ABSTRACT

According to the annual report by WHO, the increase in population, urban expansion, and the rapid growth of motorized transportation have made traffic accidents a critical global problem. Currently road fatalities are 1.35 million annually. In Ethiopia, traffic accidents death rate is 26.7 per 100,000 populations. In Addis Ababa 403 fatal crashes were registered in 2022/23. This study aimed to use machine learning techniques to predict the severity of traffic accidents in Addis Ababa by developing a predictive model. Ten machine-learning algorithms were selected and tested using historical traffic accident data from Addis Ababa Traffic Management Agency. The dataset contains 9532 instances with 19 variables, from these 80% are used for training and 20% for testing. Association rule mining reveals that less severe traffic accidents are frequently associated with dry road conditions, with a confidence value of 13.56% and a lift value of 1.066. Severe accidents are commonly linked to the absence of road junctions, with a confidence value of 78.97% and a lift value of 1.004. Fatal accidents are strongly associated with poor lighting conditions, with a confidence value of 20.41% and a lift value of 1.456. This analysis highlights the significant impact of these factors on accident severity, providing valuable insights for developing targeted prevention and mitigation strategies. Model experimentation shows that the XGBoost model achieved the highest accuracy at 84%. Decision Trees had the lowest accuracy of 72% while Bagging with SVM improved to 76% accuracy. Random Forests achieved 79% accuracy, and Ada Boost and Naïve Bayes also reached 79%. SVM, KNN, Logistic Regression, and RNN all achieved an accuracy of 80%. Despite the strong performance of several models, XGBoost remains the best performing model with the highest accuracy, underscoring its capability to make accurate predictions based on this study.

Keyword: Naive Bayes, K-Nearest Neighbors, Logistic Regression, Decision Trees, Recurrent Neural Network, Random Forest, AdaBoost, Extreme Gradient Boosting, Bagging, Support Vector Machine, traffic accident severity predication

CHAPTER ONE

1 INTRODUCTION

1.1 Background of the Study

As populations increase, cities expand rapidly, and motorized transportation explodes, ensuring road safety becomes paramount in building a society that is sustainable, inclusive, and prioritizes safety for everyone. The UN's 2030 Agenda for Sustainable Development has a crucial target: to cut the number of global road fatalities in half. Currently, this number stands at a staggering 1.35 million annually. This translates to a life lost every 24 seconds or roughly 3,700 deaths each day. Among these deaths, a shocking 1,000 occur daily among young people. In fact, road traffic injuries are the leading cause of death for young people between the ages of 5 and 29. It's important to note that over 90% of these fatalities happen in low- and middle-income countries. (UN Road Safety Fund Brochure 2021)

According to data found from WHO, traffic accident death rate in Ethiopia is 26.7 per 100,000 populations one of the highest in the world. The numbers of reported fatal crash victims are 408 in 2022/23. Pedestrians accounted for 86% of the reported deaths. Males made up 77% of the reported fatalities, 41% of the reported deaths were among those aged 20-39 years. Deaths were frequently reported following crashes which occurred between 6 and 8 p.m. 50% of deaths occurred from crashes on weekends (Fridays to Sundays). (Addis Ababa Road Safety Report 2022-2023).

Traffic Management Agency of Addis Ababa is responsible for ensuring peaceful and convenient traffic flow in the city, for awareness creation and traffic enforcement. Generally, TMA is given mission of determining, informing, improving, implementing and controlling appropriate road use in the city of Addis Ababa. (TMA Strategic Plan, 2020 - 2029).

On the other hand Traffic Police Commission is responsible for traffic enforcement and controlling activity throughout the city. TMA and Addis Ababa Police commission has implemented different kinds of traffic accident

prevention in the city. Moreover; several researches have been done on traffic accident situation by government agencies and different scholars all of these researches has proposed a number of solution in order to minimize and prevent traffic accident. However; traffic accident is still a major concern that take live of hundreds annually, brings minor to serious injuries of thousands, extreme property damage. This research aimed to provide its own contribution to suggest and show machine learning based traffic accident prevention method towards the ongoing struggle of traffic accident prevention in Capital City Addis Ababa.

1.2 Motivation of the Study

As described in the background section traffic accident is major issue in the world as well in Ethiopia Addis Ababa. Awareness Creation and enforcement measures are not made based of scientific analysis of past accidents and contributing factors. Unless effective awareness creation is made and efficient intervention is taken the death of productive man power and other innocent lives from small hopefully growing youngsters to honored elderly person will continue seriously. Moreover; the number of vehicle and pedestrians in Addis Ababa is increasing highly. Whereas the safest road network expansion and prevention mechanism implementation compared to the growth of vehicles and pedestrians is low.

This motivated us to analyze the traffic accident and develop a model that predict level of accident based on registered contributing factors, in order to support the struggle against the death of thousands of innocent people from accident. Based on this, awareness creations are made and other interventions mechanisms can be performed.

1.3 Statement of the Problem

Efforts by the Traffic Management Agency (TMA) and the Addis Ababa Police Commission to enforce traffic laws and create public awareness have not significantly reduced the high accident rates. This ongoing issue indicates that

current measures are insufficient, primarily due to a lack of comprehensive scientific analysis of past accidents and their contributing factors.

The rapid increase in both vehicular traffic and pedestrian movement in Addis Ababa further aggravates the situation. The growth in road users has not been matched by an expansion in safe road networks and effective prevention mechanisms. This imbalance highlights the urgent need for innovative and effective strategies to reduce traffic accidents.

To address this critical issue, this research aims to analyze traffic accident data and develop a predictive model to identify risk factors and predict the severity of accidents. By using machine learning techniques, the model will be based on an analysis of historical data, providing a scientific foundation for targeted interventions and awareness programs. This approach seeks to support ongoing efforts to reduce traffic accidents, thereby saving lives and reducing injuries and property damage.

1.4 Research Questions

This research is dedicated to replay the subsequent research questions:

1. How can machine learning techniques be used to identify the most significant contributing factors and their co-occurring patterns associated with traffic accident severity in Addis Ababa?
2. How can insights gained from machine learning models be used to develop targeted interventions for reducing traffic accidents in Addis Ababa?
3. Which machine learning algorithms are most effective in predicting traffic accident severity in Addis Ababa?

1.5 Objectives of the study

1.5.1 General Objective

To develop a machine learning model to predict the severity of traffic accidents in Addis Ababa, improving prevention mechanisms and safety measures.

1.5.2 Specific Objectives

Specific objectives of this research include;

- To utilize machine learning techniques to identify the most significant factors contributing to traffic accident severity in Addis Ababa.
- To implement association rule mining to discover co-occurring patterns of factors associated with high accident severity.
- To analyze the relationships between identified factors and accident severity based on the model's results.
- To suggest targeted interventions based on the model's insights focusing on high-impact factors or frequently occurring patterns linked to severe accidents.
- To compare the effectiveness of different machine learning algorithms in predicting traffic accident severity in Addis Ababa.
- To select the most effective algorithm based on performance metrics like accuracy, precision, and recall.

1.6 Scope and limitation of the study

1.6.1 Scope of the Study

This research is on traffic accidents within Addis Ababa city. The research utilizes the historical traffic accidents data obtained from AAPC and TMA. And the research will define and categorize the different levels of accidents and contributing factors. Experiment with various algorithms suitable for classification tasks, considering factors. Then assess model performance using appropriate metrics. Finally, explore the potential implications of using such a model, including benefits for accident prevention, and resource allocation for traffic safety interventions.

1.6.2 Limitation of the Study

This study has used three years traffic accident, hence limitation on not using more historical accident data and quality of data used. Moreover, inconsistent missing data on accidents or contributing factors can affect the model's

performance. And it is also impossible or difficult to use personal data associated with accidents because of ethical considerations and permissions.

The research may also have a limitation from algorithm selection and performance choosing the best machine learning algorithm for the data and problem might involve experimentation and potential suboptimal outcomes. The model developed may inherit biases from the data they are trained on, potentially leading to unfair predictions for certain driver groups or accident types. On the other hand depending on the chosen algorithm, understanding how the model makes predictions might be challenging, limiting its practical application.

The model's predictions might be less accurate for future accidents due to changing traffic patterns, regulations, or vehicle technologies. The model works only for Addis Ababa not for other cities or regions with different traffic characteristics and regulations. The model might not account for all factors contributing to accident factors, such as driver intent, witness testimonies, or legal interpretations.

1.7 Significance of the Study

As discussed in the background section of this study road traffic accident is a great challenging issue in Addis Ababa. The data used in this study indicates almost all groups of the society if a victim of RTA. This study gives additional methods of accident prevention to Addis Ababa traffic management agency. Association rule mining show which factors contribute most to or happen frequently in causing RTA. Machin learning models give better understanding of RTA categories like sever, less severe and fatal. Machine learning models shows which are the most happening categories of RTA. Based on this organized results responsible bodies like Addis Ababa Traffic Management Agency and Addis Ababa Traffic Police Commission can implement different prevention methods like targeted intervention, resource allocation, awareness creation and improved decision making. This study will help Addis Ababa by providing valuable insights for various stakeholders to develop and implement effective interventions.

1.8 Organization of the Thesis

In this section, we presented a summary of the chapters covered in this thesis report.

Chapter One deals with the background of the study, the motivation behind the study, statement of the problem, research question, general and specific objective of the study, scope and limitations and significance of the stud.

Chapter two presents the literature review and related works on traffic accident predication, the overview of road traffic accident, global facts about RTA, methods of RTA prevention including machine learning and related work are briefly seated.

Chapter three presents materials and methods. In this section, we mentioned research design, data collection, data preprocessing, association rule mining, machine algorithm selection, libraries used in model development and model evaluation that we used in our study.

Chapter four, in this chapter we have, discussed the proposed model architecture, dataset pre-processing, feature selection, proposed machine learning algorithms, model evaluation and proposed model prototype architecture.

Chapter five describes the experimentation and implementation of the proposed solution and also includes some sample.

Lastly, in chapter six, we have discussed results from experimentation and implementation. We have also included conclusion, recommendation and future works in this chapter.

CHAPTER TWO

2 LITERATURE REVIEW

2.1 Introduction

Traffic accidents are a significant public health concern globally as well as in Ethiopia specifically in Addis Ababa, causing fatalities, injuries, and economic losses. Traditional methods of traffic accident analysis often rely on statistical techniques and may not fully capture the complex relationships between various contributing factors. Machine learning (ML) offers a promising approach to analyze vast amounts of traffic accident data and identify patterns that can be used to predict future accidents and improve road safety. This thesis explores the application of ML techniques predict traffic accidents in Addis Ababa. (World Health Organization, 2023; Addis Ababa Traffic Management Agency Report, 2023),

This literature review explores the scientific background of road traffic accident, different researches as well as reports on road traffic accident and the existing researches on applying machine learning predict traffic accidents in order to reduce the consequence of vehicle traffic accident, with a specific focus on studies conducted in Addis Ababa or similar urban environments in developing countries. The review examines the types of data employed, the machine learning algorithms used, and the key findings regarding accident analysis and prediction.

2.2 Road Traffic Accident

2.2.1 What is Road Traffic Accident?

Getting people and things from point A to point B is what transportation is all about. We achieve this by using a complex network of roads, railways, waterways, and airways, along with various vehicles that travel on them. It's not just about the technology – cars, fuel, and infrastructure – it also involves people's time and energy. While transportation delivers the goods, it can also

bring unwanted baggage: dirty air, noise, traffic jams, accidents, and even deaths. (D. Levinson et al., 2009).

Getting around safely is a major part of our daily routines. Whether on foot or behind the wheel, we all rely on roads and sidewalks to navigate our lives. From workplaces and schools to shops and back home, these paths connect us to our destinations. While authorities strive to keep these routes free from trouble, accidents unfortunately happen. These crashes involving vehicles cause a lot of harm and injuries, posing a serious threat to public health. (D. Gelinne et al., 2017). Any mishap on a public roadway, or even a private one that's open to everyone, that involves a moving vehicle and leaves someone hurt or dead is considered a traffic accident. (United Nations, European Union and the International Transport Forum at the OECD, 2019).

To ensure everyone's on the same page, this is what we consider a road transport accident (United Nations, European Union and the International Transport Forum at the OECD, 2019):

Crash with Injuries: This refers to any incident involving a moving vehicle on a public or accessible private road, where at least one person is hurt or killed. This can involve collisions between vehicles, pedestrians, animals, or stationary objects. Even single-vehicle crashes count if there are injuries. It's important to note that accidents involving only property damage and terrorist attacks are excluded.

- **Fatal Crash:** When a crash with injuries results in death.
- **Non-fatal Crash:** Any crash with injuries that doesn't result in death.
- **Casualty:** Anyone injured or killed in a crash.
- **Fatality:** A person who dies immediately or within 30 days of a crash, excluding suicides.

- **Injury:** When someone is hurt in a crash but doesn't die within 30 days. These injuries typically require medical attention and exclude attempted suicides.
- **Serious Injury:** Injuries that require hospitalization for more than 24 hours.
- **Minor Injury:** Any injury that doesn't fall under the categories of fatality or serious injury.
- **Driver in a Crash:** The person operating a vehicle during the crash.
- **Passenger in a Crash:** Anyone other than the driver who is in, on, or getting in/out of a vehicle during the crash.
- **Pedestrian in a Crash:** Someone involved in a crash who isn't a driver or passenger (as defined above).
- **Vehicle-Pedestrian Crash:** Any crash involving one or more vehicles and one or more pedestrians.
- **Single-Vehicle Crash:** A crash involving only one vehicle.
- **Multi-Vehicle Crash:** A crash involving two or more vehicles. This includes:
    - **Rear-End Collision:** Crashing into a vehicle in front of you, traveling in the same direction.
    - **Head-On Collision:** Crashing into a vehicle traveling in the opposite direction, in the same lane.
    - **Turning or Crossing Collision:** Crashing into a vehicle moving laterally while crossing, entering, or leaving a road.

2.2.2 Global Facts about Road Traffic Accident

Every year, approximately 1.2 million lives are tragically lost on the world's roads. These accidents leave a devastating mark, causing injuries to an estimated 20 to 50 million people. Pedestrians, cyclists, and motorcyclists are particularly vulnerable to these crashes. Alarmingly, road traffic injuries are the leading cause of death for young people aged 5 to 29. While low- and middle-income countries account for only 60% of the world's vehicles,

a shocking 90% of road traffic fatalities occur in these regions. The human cost is immense, but the economic burden is significant as well. These crashes place a heavy financial strain on victims' families due to medical bills and lost wages. In addition, entire countries lose an estimated 3% of their GDP annually as a result of these tragedies. Fortunately, solutions exist to prevent these devastating accidents. The global community has set an ambitious goal: to halve road traffic deaths and injuries by 2030. (World Health Organization, 2023)

2.2.3 Facts in Addis Ababa about Road Traffic Accident

Addis Ababa found in the central part at an elevation of around 2400 meters (7874 feet) above sea level with a population of nearly 4 million people (Central Statistics Agency, 2013). There are about 727,651 registered vehicles in the city (DVCA, 2023). RTA accidents are concerning issue in the city, with a high number of accidents and fatalities every year. Between 2017 and 2022 there where over 140,000 accident reported in Addis Ababa, with over 16,000 injury and fatal. Several factors contribute to the accidents, like driver behavior, pedestrian behavior and vehicle age. The city has implemented strategies to improve traffic safety, but challenges remain (TMA five years report).

2.3 Factors Contributing to Road Traffic Accident

Risk is like betting on the future. It's the possibility of things going wrong when you take an action. The unsure you are of what will happen, the higher the risk. For example, driving involves navigating a web of uncertainties. Risk management in this situation is about balancing the potential dangers with the expected benefits, all while considering your own comfort level with risk. (L. Crane, 2013).

Car accidents are complex and can be caused by many things, like the design of roads, the condition of vehicles, the behavior of drivers, and even the weather. Some of these factors cause the accident itself, while others make the injuries worse. Not all factors are obvious, and some causes happen right before the crash, while others have been building up for a long time. To prevent these accidents, we need to identify the risks involved. Traditionally, people have blamed drivers, cars, or the road itself, but a better approach is to look at how all these things work together. This way, we can find ways to fix the system to make roads safer for everyone. Traffic accidents are a big problem, and to solve them, we need to consider everything that plays a role, from the design of roads to the way people behave behind the wheel. By understanding how these things interact, we can make our roads safer. (D. Mohan et al., 2006)

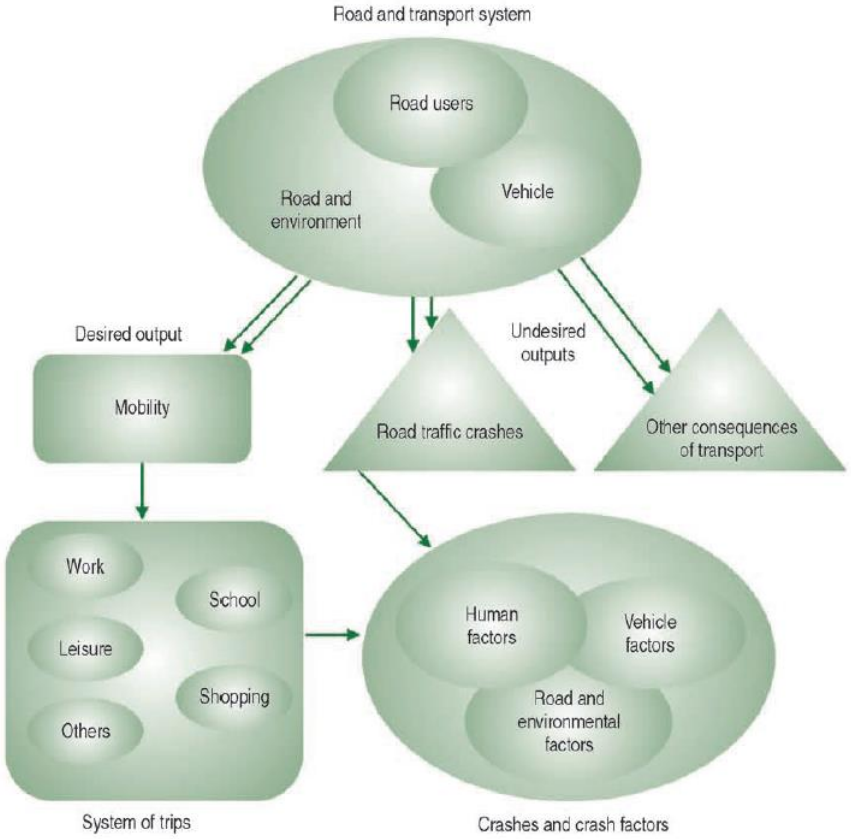


Figure 2-1: System Approach *(Source: World Health Organization. (2006). Road traffic injury prevention)*

According to World Health Organization December, 2023 report causes of vehicle crashes are:

- Careless speeding
- Drinking and driving and other drags
- Without helmets,
- Not using car seat-belts and child restraints
- Confused driving
- Risky roads
- Risky vehicles
- Poor law enforcement of traffic laws

2.4 Methods of Traffic Accident Prevention

One-size-fits-all solutions for road safety don't exist. Programs that work in one place might need tweaking to succeed elsewhere. Careful adjustments and testing are crucial. In areas where effective strategies are missing entirely, research is needed to create and assess new approaches. But some key principles apply everywhere, from wealthy nations to developing countries: reducing risk through transportation and land-use planning, designing safer roads, improving user visibility, pushing for crash-resistant vehicle designs, enforcing clear road rules, and ensuring compliance. (D. Mohan et al., 2006).

Road safety isn't a one-man job! Different specialists work together to make our travels safer. This team effort, sometimes called the "Four E's," includes engineers who design safe roads, educators who teach safe driving habits, law enforcement who keep the roads orderly, and emergency responders who help in case of accidents. Sometimes, a fifth "E" for evaluation is added to track what works and what doesn't. This highlights the importance of good data in making our roads even safer. (D. Gelinne et al., 2017).

Traffic flow can be controlled with signs or by physically shaping the road itself. Signs like "stop" or "one way" tell drivers what to do. Road design

can also achieve this. For example, a road can be narrowed to slow drivers down, or a roundabout can be built to reduce accidents. These design features are especially useful in areas with a lot of pedestrians or homes. (K. Weerasekera, 2019)

2.5 ML Based Traffic Accident Prevention

Machine learning is like a code-cracker that automatically discovers hidden rules in data. These rules are then used to build a system, like a crystal ball, that predicts what will happen next. Supervised learning is a special technique where the system is trained on past examples, allowing it to learn the connections between different pieces of information and predict what value will appear for a missing piece. (D. Kellerher et al., 2015).

With all the advanced tech around us, we're drowning in data, both organized and not. In the late 1900s, a new field called machine learning emerged from artificial intelligence. It uses special algorithms to learn from this data and make predictions. Instead of us writing out rules by hand, machine learning can find patterns in the data itself, getting better and better at making predictions as it goes. This lets us ditch the guesswork and rely on data to make smarter decisions. (S. Raschka, et al., 2022)

Traditional data analysis follows a set recipe to get an answer, while machine learning (ML) explores data to uncover hidden patterns. In other words, traditional methods use general rules to reach conclusions (deduction), whereas ML learns from examples to make predictions (induction). Traditional analysis also needs some understanding of the problem beforehand, and needs to be adjusted if things change. On the other hand, ML can automatically adapt to new situations because it learns from the data it's given. This makes ML a good choice for complex problems with unclear relationships, especially when dealing with non-standard data like images or text. However, both traditional and ML methods can work together, and traditional analysis remains useful for simpler problems. (H. Nyman, 2019).

2.5.1 Machine Learning Algorithms

Machine learning algorithm can be understood as a computational procedure encapsulated within a black box that takes input data and produces outputs. The distinguishing factor between different machine learning algorithms lies in the assumptions embedded within them. These assumptions dictate how the algorithm processes data to generate predictions or classifications. Unlike algorithms, which implicitly hold these assumptions, models in machine learning explicitly define these assumptions about the problem domain. Models represent a structured framework that combines these assumptions with inference methods, which are generic techniques used to compute variables of interest from the data. This model-based approach allows for flexibility in applying various inference methods across different models. For instance, deep learning exemplifies model-based machine learning where neural networks serve as the model, encoding assumptions through network architecture and activation functions. This model-based view not only clarifies the assumptions driving predictions but also facilitates improvements and extensions to tasks such as classification and clustering by modifying the underlying model, thereby enhancing accuracy and interpretability in machine learning applications. (D Barber, 2012)

Random Forest

Random Forest is a powerful machine learning algorithm that aggregates predictions from multiple decision trees to generate a final output. Renowned for its simplicity and versatility, Random Forest excels in both classification and regression tasks by leveraging the collective wisdom of diverse decision trees. Its robustness lies in mitigating overfitting and handling complex datasets comprising both continuous and categorical variables. This algorithm's popularity stems from its ability to deliver reliable performance across various predictive tasks in machine learning,

making it a preferred choice for data scientists and researchers alike. (Breiman, L. 2001)

Extreme Gradient Boosting (XGBoost)

XGBoost, short for Extreme Gradient Boosting, is an advanced implementation of gradient boosting algorithms designed for efficiency and performance in supervised learning tasks. XGBoost employs a gradient boosting framework to sequentially train ensemble models, iteratively improving predictive accuracy by minimizing the errors of previous models. It integrates techniques like regularization and parallel processing to handle large datasets efficiently while preventing overfitting. XGBoost's key strengths lie in its ability to handle diverse data types, its robustness against outliers, and its versatility in both regression and classification tasks. It has become a popular choice in competitions and real-world applications due to its superior performance and ease of use, making it a cornerstone in modern machine learning pipelines. (Chen, T. & Guestrin, C.2016)

Decision Tree (DT)

Decision Tree (DT) is a statistical model used for classification that organizes data into classes and represents the results in a tree-like flowchart structure. The model classifies data by flowing through a sequence of queries from the root node, which represents the most significant attribute, to the leaf nodes, which represent the classes. The classification process involves placing all training examples at the root, dividing them based on selected attributes, and recursively partitioning the data until no training examples or attributes remain, or the remaining examples belong to the same class. This method, known as Top-Down Induction of Decision Trees (TDIDTs), is the most common strategy for learning DTs from data. (Mohammed, Khan, & Bashier, 2017)

K-Nearest Neighbors Algorithm (k-NN)

The k-nearest neighbors algorithm (k-NN) is a nonparametric method used for classification and regression, where the input consists of the k closest training examples in the feature space. For classification, k-NN assigns a class based on a majority vote of its neighbors, while for regression, it predicts a value by averaging the values of its neighbors. It is an instance-based, or lazy learning algorithm, meaning computations are deferred until classification. Neighbors can be weighted based on their distance, with nearer neighbors contributing more. The k-NN algorithm is simple but sensitive to the local structure of the data, and it should not be confused with k-means clustering. (Mohammed, Khan, & Bashier, 2017)

Naïve Bayesian Algorithm

Naïve Bayesian classifiers are simple probabilistic classifiers based on Bayes' theorem, assuming strong independence among features. Bayes' theorem calculates the probability of an event given the occurrence of another event, where P (A|B) is the posterior probability, P (A) and P(B) are the prior probabilities, and P(B|A) is the likelihood. In classification, given a vector of features X and a class cj, the classifier calculates the posterior probability P (cj|X) from the prior probabilities P (cj) and P(X), and the likelihood P(X|cj). The naïve assumption, known as class conditional independence, posits that the effect of one predictor on a given class is independent of other predictors. An instance X is classified into the class with the highest posterior probability P (cj|X).

Support Vector Machine (SVM)

A Support Vector Machine (SVM) is a powerful and versatile machine learning model used for linear or nonlinear classification, regression, and outlier detection. It is particularly effective for classifying complex but small- to medium-sized datasets. The fundamental idea behind SVMs is to find a decision boundary that not only separates different classes but also

maximizes the margin between the closest training instances from each class. This approach, known as large margin classification, ensures the model performs well on new instances by fitting the widest possible margin (or "street") between the classes. (A. Géron, 2019)

Bootstrap Aggregation (Bagging)

Bagging, short for bootstrap aggregating, is an ensemble technique where multiple predictors are trained on different random subsets of the training set. When sampling is performed with replacement, it's called bagging; without replacement, it's called pasting. Both methods allow training instances to be sampled multiple times across different predictors, but only bagging permits multiple samplings of the same instance for a single predictor.

After training, the ensemble makes a prediction by aggregating the predictions of all predictors, typically using the statistical mode for classification or the average for regression. This aggregation reduces both bias and variance compared to a single predictor, leading to better generalization. Bagging can be efficiently implemented in parallel, leveraging multiple CPU cores or servers for training and prediction. This parallelism, combined with its ability to reduce variance while maintaining bias, makes bagging a popular and scalable method in machine learning. (A. Géron, 2019)

Adaptive Boosting (AdaBoost)

AdaBoost (Adaptive Boosting) is an ensemble learning method that combines multiple weak classifiers to create a strong classifier. It works by sequentially building models from the training data, where each subsequent model focuses on correcting the errors of its predecessor. Initially developed for binary classification, AdaBoost aims to improve the overall accuracy of learning algorithms. It is particularly effective when used with weak learners—models that perform slightly better than random chance,

such as decision trees with very shallow depth (often just one level). AdaBoost iteratively adjusts the weights of misclassified instances, prioritizing those that were harder to classify correctly in previous iterations. This method enhances the predictive power of the final ensemble model by emphasizing instances that are more challenging to classify correctly, thus improving overall accuracy on the training set. (J. Brownlee 2016)

Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) is a specialized class of neural networks designed for processing sequential data. Analogous to how convolutional networks excel in processing grid-like data such as images, RNNs are optimized for sequences of values they can efficiently handle sequences of varying lengths, a capability essential for tasks where sequence length may vary, unlike fixed-length feedforward networks. This flexibility is enabled by parameter sharing across different parts of the model, a concept fundamental to RNN architecture since the 1980s. Parameter sharing allows RNNs to generalize across different sequence lengths and positions in time, crucial for tasks like language processing where the same information can appear at different positions within the sequence. This contrasts with traditional feedforward networks that require separate parameters for each input feature, limiting their ability to recognize patterns across different parts of the sequence. RNNs, thus, offer significant advantages in capturing temporal dependencies and handling variable-length sequential data effectively. (Christopher M. Bishop, 2016)

Logistic Regression

Logistic regression is a supervised machine learning algorithm designed for binary classification tasks, where its objective is to predict the probability that an instance belongs to a specific class. Unlike linear regression, logistic regression employs a sigmoid function to map input variables to a probability value between 0 and 1, determining class membership based on

a specified threshold (typically 0.5). This statistical approach analyzes the relationship between independent variables and the likelihood of belonging to a particular class, making it a fundamental tool in predictive modeling, especially when dealing with categorical outcomes in various fields of study and application. (Cox, D. R. 1958)

2.6 Related Works on Traffic Accident

Some related works have been reviewed in this study. The detail is summarized Table2.1.

Table 2-1 Summary of Related Works

S. No.	Author and Title	Algorithm /Method	Gap
1	Deme D. (2019) Road Traffic Accident in Ethiopia from 2007/08-2017/18	Descriptive and inferential statistical analysis approach	This study focused solely on analyzing the connection between the growth rate of road traffic accidents, road network coverage, and the number of motorized vehicles. It did not explore solutions to address the ongoing or increasing problem of traffic accidents.
2	. Beshah, T., Ejigu, D., Abraham, A., Snášel, V., & Krömer, P. (2012, January) Knowledge discovery from road traffic accident data in Ethiopia: Data quality, ensembling and trend analysis for improving road safety	TreeNet, Classification and Adaptive Regression Trees (CART), Random Forest (RF) and hybrid ensemble approach	The research prioritizes ensuring the data used is reliable and accurate. The data used in the research over four years old and potentially not relevant to current traffic accident trends.
3	Mosisa, G. L. (2018, October) Analysis Of Road Traffic Violations In Addis Ababa City (The Case Of Arada Sub-City)	Excel and SPSS; Independent t-test, one-way ANOVA test, Chi-square, Descriptive and frequencies	The study's analysis of past data may not be relevant to preventing future accidents The research may be geographically limited. The data is four years ago, it may not reflect current traffic patterns and accident risks.

S. No.	Author and Title	Algorithm /Method	Gap
4	Beyene Gerbi, H. (2020) Statistical Analysis of Road Traffic Accident in Addis Ababa: Application of SARIMA and SETAR model	SARIMA and SETAR	The study employed traditional data analysis but suggests using Artificial Neural Networks (ANN) and Multinomial Logit models for potentially improved future research. Data older than five years may not accurately reflect current situations.
5	Deneke, S. S., & Endalkachew, A. A. (2020) Predicting Factors Contributing to Road Traffic Accident and Implying Driver's Driving Behavior in Addis Ababa City	data mining, priori algorithm	This study employed data mining techniques on 4,285 data points. While it highlights the importance of driver sex, vehicle status, and road type in various weather conditions, the authors recommend using more recent data for drawing conclusions about current traffic accident scenarios.
6	Endalie, D., & Abebe, W. T. (2023) Analysis and Detection of Road Traffic Accident Severity via Data Mining Techniques: Case Study Addis Ababa, Ethiopia	Apriori algorithm, support vector machine, K-nearest neighbors, decision trees, and random forests	More similar research relatively more recent historical data is used and other types of algorithm used.

CHAPTER THREE

3 Materials and Methods

3.1 Introduction

To address research problems require a systematic approach, and this is where research methodology comes in. It's the science behind conducting research, guiding the selection of techniques and methods for a thorough investigation. These techniques and methods themselves are the building blocks, forming the vast toolbox of research methods. (Dr. Mishra, Dr. Alok, 2017). This research utilizes a machine learning approach to predict traffic accidents in Addis Ababa. To achieve this, the chapter delves into the various methods and materials employed. Firstly, it explores the techniques used to preprocess the acquired data. Subsequently, it details the process of data acquisition itself. Finally, the chapter culminates in a discussion of the different machine learning algorithms used for the analysis.

3.2 Research Design

To clarify the research question involves a strategic steps, each informing the next in a dynamic choreography. (Dr. Mishra, Dr. Alok, 2017). The design of

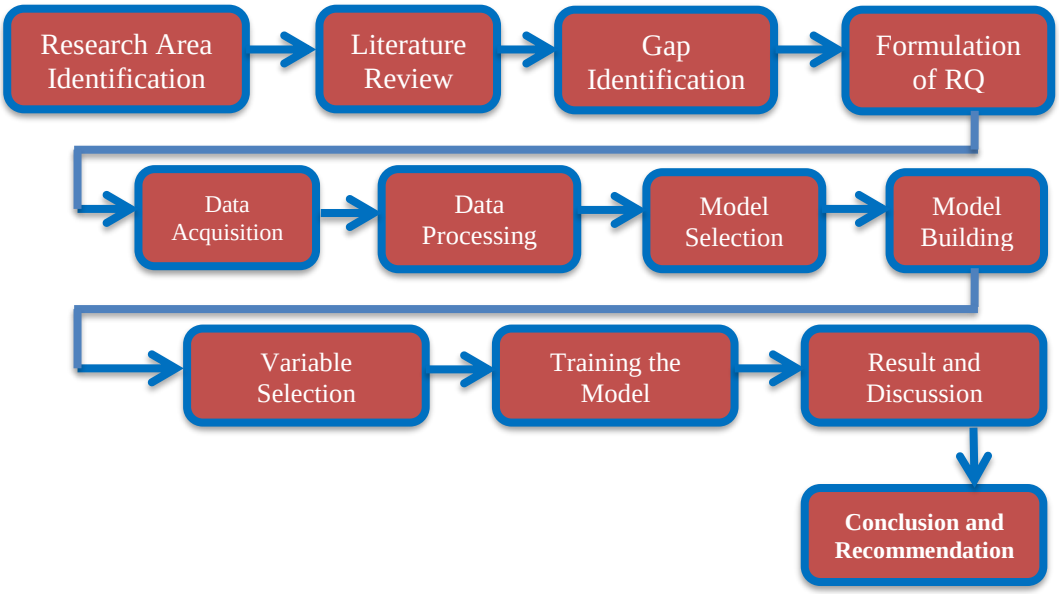


Figure 3-1: Research Design Flow Chart

this research is shown as follows in Figure 3.1. This research is a model development because of this quantitative method specifically experimental model will be used for model development in order to answer research question and successfully meet objectives of the research.

3.3 Data Collection

This section outlines the data collection methods to apply machine learning techniques accident predication model development. Understanding the contributing factors to the accident is essential to achieving the overall research goals of accident predication. The specific data collection procedures will be addressed in the following sections.

This research will cover traffic accidents registered in AAPC and TMA in conjunction perform the control and the enforcement task in the City regarding vehicle traffic. Those drivers who offend the traffic rules associated with accident and obliged to registered and enforced to practice complement measures from fines to sanctions even criminal prison. Traffic accident data is stored in a web based system that can be accessed in those eleven sub cities. The data is mainly owned by AAPC with controlled permission to TMA and FTA. This research has used three years accident data.

This data is obtained through official letter written from ASTU to both TMA and AAPC. The data is obtained from TMA. This data is three years data from 2019 to 2021 extracted from the web based traffic accident registration system. The data is categorized as minor injury, serious injury and death victims. It is raw data not processed and cleaned. While collecting the data some personal data features were removed by TMA experts because of ethical consideration. For the rest of the data features full privilege is given to the researcher for this academic model development.

3.4 Data Pre-processing

The raw data which was collected is in an excel format. The data is organized in different excel sheets by year and accident type. Traffic accidents that are minor and serious injury are grouped together in same sheet and death accidents are grouped in separate sheet for each year. The original raw and uncleaned data has 9,678 vehicle traffic accident data of the year 2019 to 2021.

Unprocessed data has about thirty five variables like Accident Id Number, Sub City, Age, Gender, Latitude, Longitude , Level of Accident, Accident Location, Date, Accident Road, Crash Area, Accident Time, Day, Road Condition, Road Junction Type, Road Pavement Type, Road Lay Out, Road Division, Light Condition, Specific Type of Accident, Weather Condition, Pedestrian Movement, Pedestrian fitness and health, On site traffic Control, Vehicle Name, Driver Type, Vehicle Plate Number, Vehicle Driver Relationship, Vehicle Movement, type of Crash, Vehicle age, Ownership, Type of Vehicle, Driver Experience and Accident Cause (offence type). However; due to ethical consideration and permission instances of some variables have been removed from the data by TMA experts.

Finally nineteen(19) variables where approved to the final dataset and these are: Sub City, Age, Gender, Date, Road Condition, Road Junction Type, Road Pavement Type, Road Lay Out, Road Division, Light Condition, Type of Accident, Weather Condition, Pedestrian Movement, Vehicle Name, Vehicle Plate Number, Vehicle Movement, Driver Experience and Accident Cause (offence type), Level of Accident.

Feature Engineering, transforms raw observations into powerful ingredients. This process involves sifting for the most impactful elements, conjuring new features by fusing existing ones, and extracting hidden insights from the raw data cauldron. pen_spark (Russell, 2018). There were totally 9,678 instances for those select variables. A data cleaning task has been performed on this dataset. Some of the tasks performed are: finding missing values: some of them that

were missing entirely and not crucial have been removed and the others are estimated based on available data by averaging surrounding values. Detecting duplicates: some duplicates that can skew analysis were removed. Inconsistent formats: some distorted values such as spelling error, special characters, and data point outside the expected range for a particular variable have been corrected. Some error outliers and some values that range out of the logical range have been also removed. After these entire cleaning task a total of 9,532 instance were selected.

From those nineteen (19) variables eighteen (18) are independent variables and one of the nineteen variable namely Level of Accident is an output or dependent variable. This variable has three output categories:

3.5 Association Rule Mining

To identify the most frequent factors contributing to the severity of traffic accidents we explore apply Association Rule Mining (ARM). ARM is a powerful data mining technique widely used for discovering interesting relationships or associations between variables in large datasets. By analyzing the patterns and correlations among different attributes of traffic accident data, ARM helps to uncover hidden insights and frequent item sets that contribute to varying levels of accident severity. This method allows us to understand the complex interplay between multiple factors such as road conditions, weather, driver behavior, and vehicle types. By identifying these key contributing factors, we can better inform traffic management and policy decisions aimed at reducing the severity and frequency of accidents. In our study, ARM serves as a crucial step towards building a comprehensive predictive model for traffic accident severity, providing a foundation for targeted interventions and improved road safety measures in Addis Ababa.

3.6 Machine Learning Algorithm Selection

For our research on traffic accident prediction, we have carefully selected ten machine learning algorithms: Adaptive Boosting (AdaBoost), Bootstrap

Aggregation (Bagging), Naive Bayes, K-Nearest Neighbors (KNN), Logistic Regression, Decision Trees, Recurrent Neural Network, Random Forest, Extreme Gradient Boosting (XGBoost), and Support Vector Machine (SVM). Each of these algorithms offers unique strengths and characteristics that contribute to creating a robust predictive model. AdaBoost and Bagging are ensemble methods that enhance predictive performance by combining the outputs of multiple weak learners, thereby improving accuracy and reducing variance. Naive Bayes, known for its simplicity and effectiveness, is particularly suitable for handling categorical data and provides probabilistic interpretations of predictions. KNN is a non-parametric method that leverages proximity-based predictions, making it effective for capturing local data patterns. Logistic Regression, a widely-used linear model, excels in binary classification tasks and provides interpretable coefficients for each feature. Decision Trees offer intuitive and easy-to-understand decision-making processes, making them valuable for understanding the underlying factors in accident occurrences. Random Forest, an extension of decision trees, mitigates overfitting and improves generalization by averaging the results of multiple trees. XGBoost is a powerful gradient boosting algorithm that enhances prediction accuracy by optimizing performance through regularization techniques and handling missing data effectively. SVM is known for its robustness in high-dimensional spaces and effectiveness in non-linear classification problems. Finally, Recurrent Neural Networks (RNNs), despite being more commonly used for sequential data, are included to explore potential benefits in capturing temporal dependencies within the dataset. By utilizing this diverse set of algorithms, we aim to comprehensively evaluate different approaches and identify the most effective methods for accurately predicting traffic accidents, ultimately enhancing road safety in Addis Ababa.

3.7 Libraries Used in the model development

While developing a machine learning model the research has used some basic libraries. Here are some popular Python libraries the research can leverage to

implement these algorithms for the traffic accident data analysis predication model development:

- Scikit-learn: This versatile library provides implementations for all the algorithms selected (SVM, Random Forest, and Decision Tree). It's a user-friendly option with extensive documentation and tutorials, making it a great choice for beginners.
- General-Purpose Libraries like
    - Pandas: A high-level data analysis and manipulation library offering data structures like DataFrames for easy handling of tabular data.
    - Matplotlib: A comprehensive library for creating various visualizations like static, animated, and interactive plots.
- Other libraries like: seaborn. Statsmodel,etc

3.8 Development Environment

3.8.1.1 Hardware Tools

To implement this research, a personal computer equipped with the following hardware specifications has been used. The system with an internal hard disk capacity of 1TB, 16GB of physical memory, 11th Gen Intel(R) CPU processor operating at 2.40GHz, Windows 10 Pro as the operating system, and a 64-bit operating system architecture on an x64-based processor.

3.8.1.2 Software Tools

For model development Anaconda Platform with Jupyter Notebook that is a web-based interactive environment for coding, data analysis, and visualization, popular for data exploration and creating reports. In addition to this google colab environment is used. As programming language python 3 is used.

3.9 Model Evaluation

Checking how good our model is at predicting accident severity isn't enough. We need to dig deeper and use different fancy score things (metrics) besides

just "how many got it right?" These score things will tell us how good the model is at finding certain types of accidents, like ones with bad injuries. They will also show us where the model might struggle. By looking at these scores, we will get a much clearer picture of what the model does well and where it needs improvement. (Kelleher, Mac Namee, and D'Arcy (2018))

Accuracy = (True Positives + True Negatives) / Total Samples

High accuracy indicates the model performs well on average across all severity categories (low, medium, high). However, it doesn't reveal details about specific categories. An imbalanced dataset (more minor-injury accidents) can lead to high overall accuracy even if the model performs poorly on less frequent categories (serious injury).

Precision: focuses on the positive predictions (e.g., accidents classified as serious injury). It measures the proportion of accidents identified as serious injury that are truly serious injury.

Precision (Serious Injury) = True Positives (Serious Injury) / (True Positives + False Positives)

High precision for serious injury signifies it better result for ignoring false alarms it identifies most accidents classified as serious injury as genuinely serious. This is crucial if the cost of misclassifying a minor-injury accident as serious or death is significant.

Recall: how good the model is at finding all the important injury accidents, tells us what percent of real bad accidents the model actually spotted. It measures the proportion of actual serious injury accidents that the model correctly classified as serious injury.

Recall (serious injury) = True Positives (serious injury) / (True Positives + False Negatives)

High recall for serious level indicates the model excels at identifying most serious injury accidents. This is important if missing a serious injury accident has severe consequences.

F1-Score: is a harmonic mean that provides an average between precision and recall. It considers two how precise the model is for the serious injury class and how well it captures all serious injury accidents.

$$\text{F1-score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

CHAPTER FOUR

4 Proposed Solution

4.1 General

Chapter four will describe about a proposed machine learning-based method to tackle traffic accidents that are a significant concern in Addis Ababa, leading to injuries, fatalities, and economic losses. Addressing this issue requires a proactive approach that can identify potential accident zones by predicting accident severity. It also details the proposed process of acquiring, cleaning, and transforming traffic accident data into a best format for machine learning algorithms. It also explores proposed techniques to select finest features from the data that might improve the model's predictive capacity. The selection of appropriate machine learning algorithms based on the nature of the data and the expected outcomes is also done in this section. Finally, model training and evaluation has been detailed.

4.2 Proposed Model Architecture

The aim of this research is mainly to build a predication model to predict severity of vehicle traffic accident in Addis Ababa City using Machine learning algorithms. The model uses historic traffic accident data collected in Addis Ababa City and utilize those patterns to predict future accident severity. By understanding the contributing factors to accidents, it is possible to develop plan and controlling to improve traffic road safety in Addis Ababa. This proposed model architecture can be divided into several key stages and they are described below and summarized as in figure--.

Data Preprocessing*:* This initial stage involves preparing the raw accident data. Tasks like cleaning the data to check and remove inconsistencies or missing values, integrating data from various sources, and encoding categorical features into numerical representations.

Machine Learning Model Training*:* once data cleaning finalized the research has utilized the chosen ML algorithms to train from the preprocessed data. This research has explored ten algorithms.

Model Selection and Evaluation*:* the research has train the model using 80% of the preprocessed data. Since evaluating a model's performance on unseen data is crucial 20% is used for test.

The research has used various evaluation techniques like accuracy, precision, recall, and F1 score for checking model performance and also confusion matrix. The model with the best overall performance on the validation set has been chosen as the final model for accident severity prediction.

The proposed model architecture provides a robust framework for harnessing the power of machine learning predict traffic accidents in Addis Ababa. By understanding the factors contributing to accidents, we can develop targeted interventions and improve road safety for all citizens.

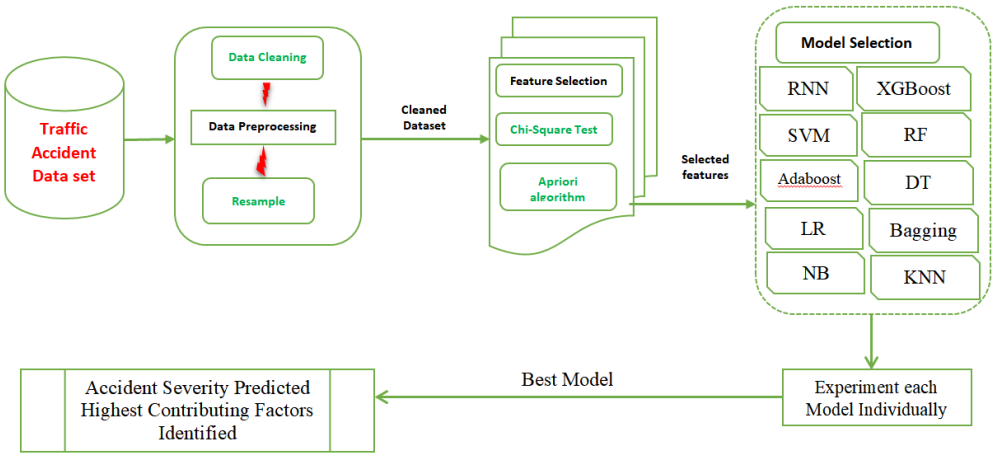


Figure 4-1: Accident Severity Predication Model

4.3 Traffic Accident Dataset

This research targets to develop a traffic accident prediction model using ML. Therefore, for this new dataset is compiled. Currently, there is a scarcity of published or readily available datasets for traffic accident prediction in Addis

Ababa. The process of constructing the dataset for traffic accident prediction involves several key steps, including data collection from the Traffic Management Agency (TMA) and the Addis Ababa Police Commission (AAPC), followed by data preparation to ensure suitability for model development.

To build the traffic accident dataset, the researcher gathered relevant data from TMA and AAPC databases, focusing specifically on records pertaining to traffic accidents within Addis Ababa. Due to constraints such as incomplete information and time limitations, only a subset of the available data was utilized. Primarily, data related to traffic accidents resulting in severe injuries or fatalities were selected for inclusion in the dataset. Information such as accident location, time, weather conditions, and vehicle types involved, and severity of injuries were collected and organized in a structured format suitable for model training and evaluation.

4.3.1 Dataset Pre-processing

This section describes the process of identifying and addressing missing values, inconsistencies, and errors within the data. One of the critical stages in developing a robust traffic accident prediction model is the preprocessing of the collected data. The dataset obtained from the Traffic Management Agency (TMA) and the Addis Ababa Police Commission (AAPC) contains inconsistencies and lots of missing values that could affect the operation of the ML model. Therefore, it has been cleaned before proceeding with model training and evaluation.

4.3.1.1 Data Cleaning

Data cleaning ensures that the dataset is steady, reliable, and suitable for machine learning. Among the common techniques the research has employed during data cleaning include handling missing values and addressing outliers. Regarding the missing values, all records with missing values were dropped based on a specified threshold, and the remaining missing values were handled manually through imputation methods such as mean, median, or mode replacement depending on the

nature of the feature. Outliers were identified using statistical methods like Z-score analysis or visualization techniques such as box plots. They were subsequently handled by either removing them if they were deemed anomalies due to data entry errors or correcting them through transformations or adjustments to ensure they do not unduly influence model training.

4.3.2 Feature Selection

This section is like figuring out the most important ingredients in a technique for predicting accidents. We have a bunch of data points about past accidents, but not all of them are equally helpful. Here, we'll explore different ways to pick the most important clues from this data. Imagine checking each clue one by one to see how well it helps predict accidents. We can also try different combinations of clues to see which ones work best together. There's even a way to let the computer program itself figure out the best clues while it's learning embedded methods. By finding the best clues, we can make our accident prediction program more accurate and easier to understand. This will help people make better decisions about road safety.

4.3.3 Proposed Machine Learning Algorithm

The focus of this research is to build a predictive model for accident severity (fatal, severe, less severe) using historical data from traffic accidents registered in Addis Ababa City. Given the dataset's characteristics, the selection of appropriate machine learning algorithms suited for classification tasks is crucial. This section explores ten machine learning algorithms known for their effectiveness in predictive modeling of traffic accidents: Adaptive Boosting (AdaBoost), Bootstrap Aggregation (Bagging), Naive Bayes, K-Nearest Neighbors (KNN), Logistic Regression, Decision Trees, Recurrent Neural Network (RNN), Random Forest, Extreme Gradient Boosting (XGBoost), and Support Vector Machines (SVM).

4.3.4 Proposed Model Evaluation

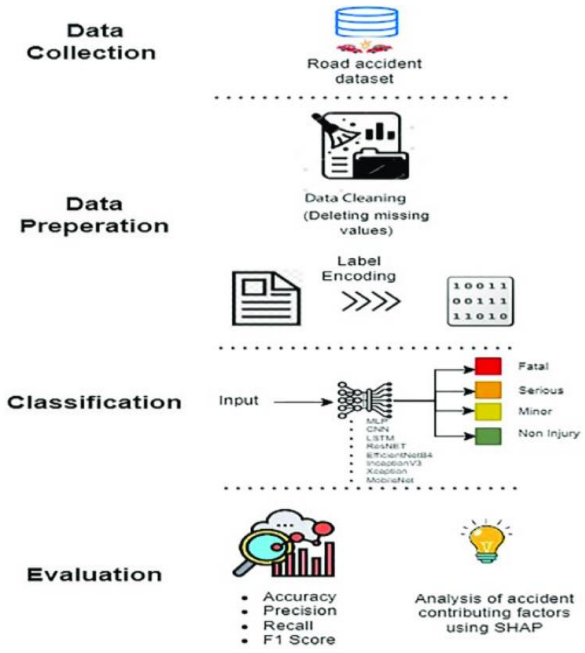
After building our awesome accident prediction model, it's time to test it out. This section dives into how we will check how well the model in predicting accident severity. We will feed it some information it hasn't seen before and see

how close its guesses are to real-life accidents. This will help us understand how well our model works and if there's room for improvement. This is done using classification metrics like precision, recall and F1 score.

4.3.5 Proposed Model Prototype Architecture

In this section of the research present detailed overview of the architecture and design principles underlying the proposed traffic accident severity prediction model. This section serves as a foundational framework for understanding the structural components and computational mechanisms employed in our model's development. By delineating the key architectural elements, including data preprocessing pipelines, feature extraction modules, model architectures, and inference mechanisms, we aim to provide clarity on the structural organization and functional capabilities of our proposed prototype.

Through a holistic examination of the model architecture, the study gives insights into the underlying methodologies and computational strategies driving the approach towards accurate and reliable accident severity prediction.



Source: Aboulola, O. I. (2024)

Figure 4-2: Proposed Model Architecture

CHAPTER FIVE

5 EXPERIMEMTATION AND IMPLEMENTAION

5.1 Introduction

This chapter is the core of the research, where ideas from earlier chapters are applied and experimented. Here, the research uses machine learning algorithms to predict accident severity. Through rigorous experimentation and evaluation, the research aims to identify the most effective algorithm(s) capable of accurately predicting traffic accidents in Addis Ababa.

5.2 Deployment environment

In this section, the research has detailed the deployment environment utilized for conducting experimentation and implementing machine learning algorithms. The choice of implementation environment is vital, as it directly influences the efficiency, scalability, and reproducibility of the research efforts. Similar to the experimentation setup, the selection of tools and technologies is based on robustness, versatility, and ease of use, ensuring that implementation activities are conducted with precision and efficacy.

The implementation environment is Anaconda based Jupyter Notebook, a powerful interactive computing platform. Using Jupyter Notebook's interactive nature, the research seamlessly combines code execution, visualization, and narrative documentation, facilitating a cohesive and transparent approach to the implementation activities. In addition two this google colab is also used to run some experiments.

Python programming language is used with Jupyter Notebook, a ubiquitous choice in the field of machine learning due to its rich ecosystem of libraries, extensive community support, and readability. Python serves as the backbone of the implementation efforts, providing a versatile and expressive framework for developing, testing, and deploying models with ease.

5.3 Libraries used

This section presents the process of importing necessary libraries from the Anaconda distribution. These libraries provide a rich set of functionalities that will be instrumental throughout the research experimentation. Libraries imported can be generalize in three section as: Data manipulation: Libraries like pandas will enable us to work with data frames, perform cleaning operations, and explore data characteristics. And finally, Data visualization: Libraries like matplotlib and seaborn provide helpful tools to create informative visualizations that enhance understanding of the data and model performance.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import OneHotEncoder
from sklearn.svm import SVC
import folium
from sklearn import svm
from numpy import loadtxt
from sklearn.utils import resample
from xgboost import XGBClassifier
from sklearn.feature_selection import SelectKBest, chi2
```

Figure 5-1: Code Snippet showing imported libraries and modules

5.4 Starting with Dataset

This section shows the gate to introduce with dataset world used. The dataset serves as the foundational building block upon which the entire analysis and modeling efforts are based. As such, it is essential to approach its exploration, preprocessing, and understanding with particular attention to detail and working accuracy. The research has used the following python code snippet to call the dataset prepared for machine learning based analysis and predication modeling.

```
df = pd.read_csv("C:/Users/PC/Desktop/Accident_Dataset.csv")
```

```
df = pd.read_csv("/content/Accident_Dataset.csv")
```

Figure 5-2: Code Snippet for Loading the Dataset in Anaconda and Colab

5.5 Data Processing Deployment

Raw data often contains inconsistencies, missing values, and irrelevant information that can hinder the performance of machine learning models. To ensure our model can effectively learn patterns and relationships within the data, a crucial step is data preprocessing. This section will detail the process of cleaning, transforming, and preparing our dataset. The dataset is loaded to the development environment and displayed using the code snippet shown.

5.5.1 Deployment of Data Cleaning

After loading the data set to the development environment data cleaning has been executed. So as to make the source dataset suitable or ready for machine learning model deployment, first missing values were addressed using *dropna* function of python.

In the data preprocessing phase, missing values are a critical consideration for ensuring the integrity and reliability of the dataset. Firstly, the extent of missing ness in each attribute using the *isnull().sum()* function performed. Subsequently, to maintain data quality and consistency, a threshold-based approach is implemented to drop rows with a substantial number of missing values. Specifically, rows containing at least nine (9) missing values are eliminated from the dataset using the *dropna()* function with the thresh parameter. The original dataset is retained for reference, while the processed dataset, empty of rows surpassing the designated threshold, is stored in a new CSV file. This particular data cleaning procedure ensures the reliability and accuracy of the subsequent modeling tasks.

For the remaining missing value has been filled manually based on a technique called domain based knowledge, by discussing with experts of TMA. Moreover detailed visual inspection and correction has been done to avoid and correct any deviation of every single values of the dataset.

```
# Check for missing values
missing_values = df.isnull().sum()
print("Missing values:\n", missing_values)

# Drop rows based on a threshold of missing values per row
missing_threshold = 9 # Drop rows with at least 9 missing values
df_dropped_thresh = df.dropna(axis=0, thresh=missing_threshold)

print("Original DataFrame:\n", df)

print("\nDrop rows with at least",missing_threshold,"missing values:\n", df_dropped_thresh)

df_dropped_thresh.to_csv("cleaned_dataset.csv", index=False)
```

Figure 5-3: Code Snippet for handling missing records

5.5.2 Deployment of Missing Value Removal

The raw dataset contains a lot of missing values in each instance under all variables. Instance with more than 9 missing variables were removed.

```
missing_threshold = 9
df_dropped_thresh = df.dropna(axis=0, thresh=missing_threshold)
```

Figure 5-4: Figure Showing dropping missing values

5.5.3 Deployment of Removal of Outliers

The dataset contains outliers in order to make the dataset suitable for model development the outliers were removed using IQR function.

```
def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
```

Figure 5-5: Removal of Outliers

5.5.4 Deployment of Data Encoding

The original dataset contains almost all string values. As shown in figure 5-6 below.

	Sub_City	Gender	Age	Date_Accident_Occur	Road_Condition	JTR(Junction Type of the Road)	Road_Pavement_Type	Road_Layout	Road_Division	Light_Condition	Type_Of
0	Yeka	Male	50	February/2022	Dry	No Junction	Good Asphalt	Staright and Flat	One way	Dark With Weak Road Light	Pedest
1	Kolfe Keranio	Male	35	Februry/2022	Dry	Cross Intersection	Good Asphalt	Staright and Flat	One way	Day Light	Pedest
2	Kolfe Keranio	Unknown	29	Febraury/2021	Dry	Cross Intersection	Good Asphalt	Staright and Flat	Two way	Day Light	Pedest
3	Addis Ketema	Male	20	July/2021	Dry	Cross Intersection	Good Asphalt	Staright and Flat	Devided by Roundabout	Day Light	Pedest
4	Bole	Male	34	March/2022	Dry	No Junction	Good Asphalt	Staright and Flat	Divided by Island	Day Light	Pedest

Figure 5-6: original dataset before encoding

In order to make the model development possible this string data should be encoded to integer values using the following code snippets:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Load the dataset
df = pd.read_csv('cleaned.csv')

# Select the categorical columns to encode
categorical_columns = ['Age_band_of_driver', 'Sex_of_driver', 'Educational_level',
       'Vehicle_driver_relation', 'Driving_experience', 'Lanes_or_Medians',
       'Types_of_Junction', 'Road_surface_type', 'Light_conditions',
       'Weather_conditions', 'Type_of_collision', 'Vehicle_movement',
       'Pedestrian_movement', 'Cause_of_accident', 'Accident_severity']

# Apply label encoding to each categorical column
label_encoder = LabelEncoder()
for column in categorical_columns:
    df[column] = label_encoder.fit_transform(df[column])
# Display the updated dataset with encoded categorical variables
df.head(100)
```

Figure 5-7: Code Snippet to encode the dataset

The encoded dataset is shown in the following figure 5-6 bellow:

	Sub_City	Gender	Age	Date_Accident_Occur	Road_Condition	Road_Junction_Type	Road_Pavement_Type	Road_Layout	Ro
0	10	1	46	127	0	5	1	7	
1	6	1	29	324	1	4	1	7	
2	6	2	22	299	0	4	1	7	
3	0	1	13	111	0	4	1	7	
4	3	1	28	131	0	5	1	7	

Figure 5-8: Few Sample of encoded dataset

5.5.5 Deployment of Data Correlation

To know the connection between variables of the dataset correlation matrix is done and show in figure 5-7 bellow. This will help us to understand the tendency of for the values of two features to move together in a linear fashion.

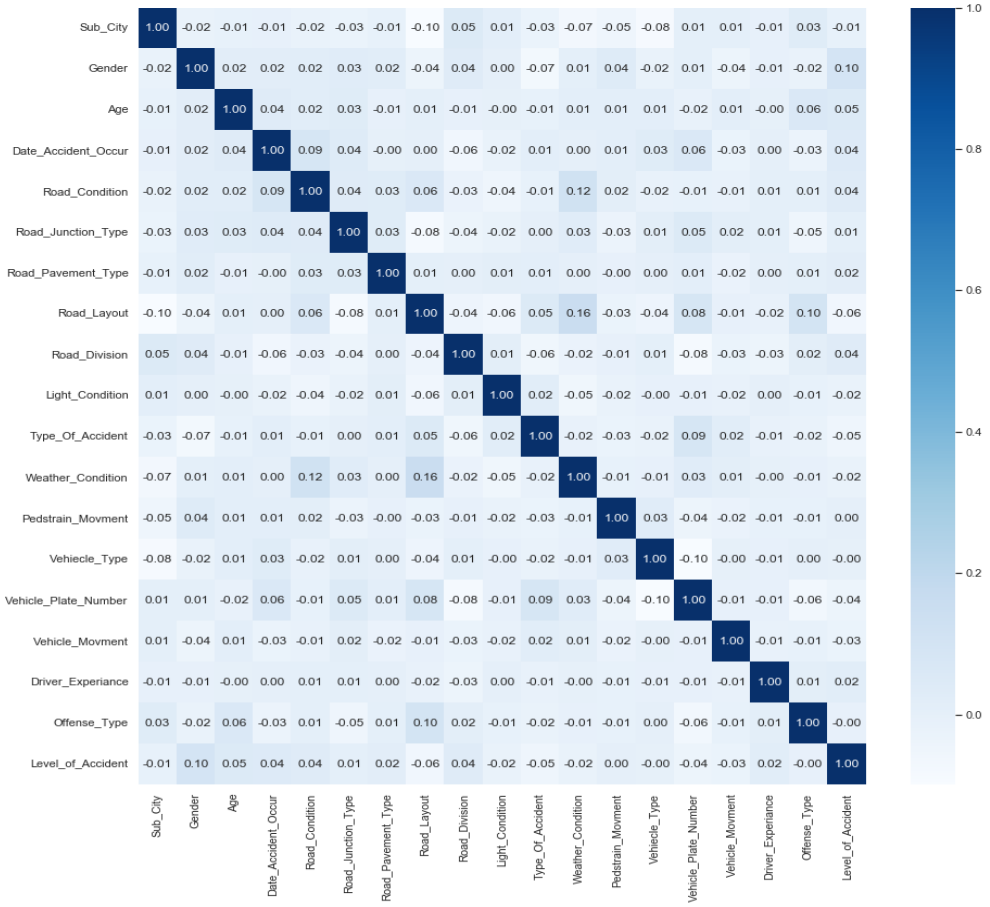


Figure 5-9: Correlation matrix of the dataset

5.5.6 Deployment of Feature Selection

This section talks about choosing the best features for a machine learning model. It's like picking the most important ingredients in a recipe. Here, the chi-squared test is used to see how much each feature is related to the final outcome we're trying to predict. Features that are strongly connected are more likely to be useful for the model. The scikit-learn library provides a tool called SelectKBest that picks the top k features based on their chi-squared scores. In

short, this helps us focus on the most relevant information and build a more efficient model.

The SelectKBest object with chi2 as the scoring function and specifies k=50, indicating that the top 50 features with the highest chi-squared scores will be selected. The encoded_df represents the input features in encoded format in one-hot encoding, and y_en represents the target variable (output) encoded into numerical labels. After fitting the SelectKBest object to the data, the X_new variable holds the transformed dataset containing only the selected features. Additionally, the cols variable stores the names of the selected features, obtained using the get_feature_names_out method.

```
# feature seleciton method using chi2 for categorical output, categorical input
fs = SelectKBest(chi2, k=50)
X_new = fs.fit_transform(encoded_df, y_en)

# Taking the selected features
cols = fs.get_feature_names_out()

# converting selected features into dataframe
fs_df = pd.DataFrame(X_new, columns=cols)
fs_df.shape, y_en
```

Figure 5-10: Code Snippet for Selecting Features

5.5.7 Deployment of Association Rule Mining

Association rule mining is a powerful data mining technique used to discover interesting relationships and hidden patterns within large datasets. In the context our traffic accident analysis, this method can be particularly valuable for identifying frequent factors and their combinations that contribute to the severity of accidents. By analyzing historical traffic accident data, association rule mining helps to reveal associations between various attributes such as weather conditions, road types, time of day, vehicle types, and more.

In traffic accident datasets, numerous variables interact in complex ways, making it challenging to pinpoint the key contributors to accident severity. Association rule mining addresses this challenge by systematically evaluating the data to find rules that describe how combinations of factors are related to

different levels of accident severity. For example, a rule might indicate that accidents occurring at night and involving heavy vehicles on wet roads have a higher likelihood of being severe.

The primary objective of using association rule mining in this research is to discover the most frequent factors that contribute to different levels of accident severity fatal, severe, and less severe. This process involves generating a set of association rules from the dataset and evaluating their significance and strength using metrics like support, confidence, and lift. Support measures the frequency of occurrence of a rule in the dataset, confidence indicates the likelihood of the rule's consequence given its antecedent, and lift provides a measure of how much more likely the rule's consequence is compared to a random occurrence.

By identifying these key associations, the research aims to provide insights that can inform traffic safety measures, policy decisions, and targeted interventions to reduce the severity and frequency of traffic accidents in Addis Ababa City. The findings from association rule mining will complement the predictive modeling efforts, enhancing the overall understanding of traffic accident dynamics and contributing to more effective road safety strategies.

```
# Set the minimum confidence threshold
min_confidence = 0.5
# Generate association rules
association_rules = generate_association_rules(frequent_itemsets, min_confidence)
# Convert the results to a DataFrame for better readability
rules_df = pd.DataFrame(association_rules, columns=['Antecedents',
                                                    'Consequents', 'Support',
                                                    'Confidence', 'Lift'])
# Filter rules where 'Level_of_Accident' is in the consequents
rules_of_interest = rules_df[rules_df['Consequents'].apply(lambda x:
                                                           'Level_of_Accident' in x)]
```

Figure 5-11: Association Rule Mining

5.6 Deployment Training and Test Sets Classification

In this research we have used twenty (20) to eighty (80) ration train test split. This means twenty percent (20%) of the dataset is used for test and the other

eighty percent (80%) of the data set is used for training the machine learning model. The following figure shows the distribution of the three classes

	Level_of_Accident	Count	Set
0	Less Severe	989	Training
1	Severe	5982	Training
2	Fatal	654	Training
0	Less Severe	223	Test
1	Severe	1511	Test
2	Fatal	173	Test

Table 5-1 Train Test Distribution:

Figure 5-12: Class Distribution in Training and Test Sets

5.7 Deployment of Machine Learning Model

5.7.1 Deployment of Random Forest (RF)

To implement the Random Forest classifier for predicting vehicle traffic accidents, we followed a systematic approach. First, necessary libraries such as pandas, numpy, sklearn, and seaborn were imported into the development environment. The cleaned dataset, saved as “Accident_Dataset.csv,” was then loaded. A Random Forest Classifier object was created, specifying the number

of decision trees (n_estimators=100) to construct the forest. This ensemble approach ensures more robust predictions by leveraging the diversity of multiple decision-making trees. A random_state was set to ensure consistent results across multiple runs. The model was trained using the fit method, where training features and corresponding target labels were provided to learn their relationships. The trained model then generated predictions for accident severity levels on unseen test data points using the predict method. Finally, the predicted numerical labels were converted back to their original categorical form using the inverse_transform method of the label encoder, enhancing the interpretability of the results.

```
# Create and train the Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)
```

Figure 5-13: Implementing Random Forest Model

5.7.2 Deployment of Extreme Gradient Boost (XGBoost)

Implementation of XGBoost classifier to predict accident severity levels has been done as shown in figure 5-11. It starts by gathering necessary libraries for data manipulation, model building, evaluation, and visualization. The accident data is loaded and prepared, separating the target variable from the features. Categorical labels are transformed into numerical values for XGBoost compatibility.

Next, the data is split into training and testing sets. The training data is converted into a format suitable for XGBoost and used to train the model with specified parameters. These parameters define the classification problem type (multi-class with softmax activation), the number of classes to predict based on severity levels.

After training, the model predicts accident severity levels for unseen data points in the testing set. The predicted numerical labels are then converted back to their original categories for easier interpretation.

```
# Train the XGBoost model
num_rounds = 100
xgb_model = xgb.train(params, dtrain, num_rounds)
```

Figure 5-14: Implementing XGBoost Model

5.7.3 Deployment of Support Vector Machine (SVM)

This section outlines the process of implementing and assessing an SVM model. After all the necessary libraries were imported and the dataset is loaded to identify the most relevant features from the input data and improve model performance and reduce training time feature selection performed. Categorical features are converted into numerical representations suitable for SVM processing. The target variable is also transformed numerically. Then model implementation was done as shown in figure 5-12 bellow.

```
clf = svm.SVC(kernel='poly', degree=2, C=1.0, gamma='scale', class_weight=None)
clf.fit(X_train, y_train)
```

Figure 5-15: Implementing SVM Model

eployment of Adaptive Boosting (AdaBoost)

Adaptive Boosting (AdaBoost) is implemented to enhance the prediction accuracy by combining multiple weak classifiers. Libraries such as pandas, numpy, and sklearn were imported. The cleaned dataset was loaded, and AdaBoostClassifier was initialized with a base estimator and a specified number of estimators. The model was trained using the fit method, and predictions were made on the test set. The results were then evaluated and interpreted.

```
# Build an AdaBoost classifier with Decision Trees as base estimators
base_estimator = DecisionTreeClassifier(max_depth=1)  # Weak learner
ada_boost_model = AdaBoostClassifier(base_estimator=base_estimator,
                                     n_estimators=50, random_state=42)

# Fit the AdaBoost model
ada_boost_model.fit(X_train, y_train)
```

Figure 5-17: Implementing AdaBoost

5.7.5 Deployment of Bootstrap Aggregation (Bagging)

The Bagging classifier aggregates the predictions of multiple base learners to improve stability and accuracy. Necessary libraries were imported, and the dataset was loaded. A BaggingClassifier was initialized with a base estimator and a specified number of estimators. The model was trained and used to make predictions on the test set. The aggregated predictions were then evaluated for accuracy.

```
# Build Bagging ensemble model using Decision Tree as base estimator
base_estimator = DecisionTreeClassifier()
model = BaggingClassifier(base_estimator=base_estimator, n_estimators=50, random_state=42)
```

Figure 5-18: Implementing Bagging

5.7.6 Deployment of Naive Bayes

Naive Bayes classifier, known for its simplicity and effectiveness, was implemented by importing the necessary libraries and loading the dataset. The data was preprocessed, and the GaussianNB classifier was initialized. The model was trained on the training set and used to predict the test set. The predictions were evaluated to determine the model's performance.

```
# Naive Bayes classifier
model = MultinomialNB()
```

Figure 5-19: Implementing Naive Bayes

5.7.7 Deployment of K-Nearest Neighbors (KNN)

The KNN classifier was implemented to leverage proximity-based predictions. Necessary libraries were imported, and the dataset was loaded. The data was scaled, and the KNeighborsClassifier was initialized with a specified number of neighbors. The model was trained and used to make predictions on the test set, which were then evaluated for accuracy.

```
# Build K-Nearest Neighbors (KNN) model
knn_model = KNeighborsClassifier(n_neighbors=20)  # Specify the number of neighbors
knn_model.fit(X_train, y_train)
```

Figure 5-20: Implementing KNN

5.7.8 Deployment of Logistic Regression

Logistic Regression was implemented for its ability to handle binary classification tasks. Libraries were imported, and the dataset was loaded. The LogisticRegression classifier was initialized, trained on the training set, and used to predict the test set. The predictions were evaluated for accuracy and interpretability.

```
# Build Logistic Regression model
model = LogisticRegression(max_iter=1000, random_state=42)
```

Figure 5-21: Implementing Logistic Regression

5.7.9 Deployment of Decision Trees

Decision Trees were implemented for their intuitive decision-making process. Necessary libraries were imported, and the dataset was loaded. A DecisionTreeClassifier was initialized, trained on the training set, and used to predict the test set. The tree's structure was analyzed for interpretability.

```
# Build Decision Tree model
model = DecisionTreeClassifier(random_state=42)
```

Figure 5-22: Implementing Decision Trees

5.7.10 Deployment of Recurrent Neural Networks (RNNs)

RNNs were implemented to explore potential benefits in capturing temporal dependencies. Libraries such as TensorFlow and Keras were imported. The dataset was preprocessed, and the RNN model was built and trained on the training set. Predictions were made on the test set, and the model's performance was evaluated.

```
# Build Reccurent Neural Network model
model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(3, activation='softmax')  # 3 output classes (0, 1, 2) for Level_of_Accident
])

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Early stopping to prevent overfitting
early_stop = EarlyStopping(monitor='val_loss', patience=50, restore_best_weights=True)
```

Figure 5-23: Implementing RNN

5.8 Deployment of Model Evaluation

To evaluate the model's performance we have used two key metrics: a classification report and a confusion matrix. The classification report, generated by classification_report, provides a detailed breakdown of the model's accuracy for each predicted accident severity level. It includes metrics like precision (percentage of correctly predicted positives), recall (percentage of actual positives the model identified), and F1-score (harmonic mean of precision and recall). The confusion matrix, generated by confusion_matrix, visualizes the number of correct (predicted correctly) and incorrect (predicted incorrectly) classifications for each possible combination of true and predicted labels. Seaborn's heatmap function creates a color-coded representation of the matrix, making it easier to interpret the model's performance across different accident severity levels. These evaluation methods provide valuable insights into the strengths and weaknesses of the models for predicting accident severity. Figure 5-24 below shows model evaluation for Random Forest Model. Model evaluations for the other models were done in similar way. We have done evaluation of the other models using similar code and replacing the specific name of the model to be evaluated.

```
# Generate the classification report
report = classification_report(y_test_labels, predicted_labels)
print("Classification Report:\n", report)
# Generate the confusion matrix
matrix = confusion_matrix(y_test_labels, predicted_labels)
print("Confusion Matrix:\n", matrix)
annot_kws={'fontsize':14}
classes = label_encoder.classes_
# Create a color-coded confusion matrix
sns.set(font_scale=1.4)
plt.figure(figsize=(9,7))
sns.heatmap(matrix,annot=True,cmap='Blues', fmt="d", annot_kws=annot_kws,xticklabels=classes, yticklabels=classes)
# cbar=0, cmap="YlGnBu",linewidths=2, ax=ax0,vmax=3000, vmin=0
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Random Forest Confusion Matrix")
plt.savefig("RFheatmap.png")
plt.show()
```

Figure 5-24: Model Evaluation for Random Forest Model

CHAPTER SIX

6 RESULTS, EVALUATION AND DISCUSSIONS

6.1 Introduction

The previous chapters built a machine learning model to predict severity of vehicle traffic accidents in Addis Ababa. This chapter explores into how well it works. Researchers will use different measurements to see how accurate the model is at predicting accidents. By carefully looking at the results, they hope to understand what causes these accidents better. This will help them develop better ways to manage traffic and create policies to keep the roads safer. Finally, the chapter will discuss what the findings mean, what the study's limitations are, and what future research could look at. It will then conclude with recommendations to improve traffic safety in Addis Ababa.

6.2 Source Dataset Target Class Distribution Result

The source dataset used in this research after cleaning has nineteen feature these are: Sub City, Age, Gender, Date, Road Condition, Road Junction Type, Road Pavement Type, Road Lay Out, Road Division, Light Condition, Type of Accident, Weather Condition, Pedestrian Movement, Vehicle Name, Vehicle Plate Number, Vehicle Movement, Driver Experience and Accident Cause (offence type) and Level of Accident. The features can be categorized as road related feature, driver related features, pedestrians related features and vehicle related feature. The data set contain a total of 9532 features. The nineteenth Level of Accident is a target variable categorized into "Severe," "Less Severe," and "Fatal." There are 7493 instances classified as "Severe," making it the most frequent category and suggesting that a large proportion of recorded traffic accidents are severe. The "Less Severe" category has 1212 instances, which, while less frequent than "Severe," still represents a significant number of accidents with less severe consequences. The "Fatal" category, with 827

instances, is the least frequent, indicating the number of accidents that resulted in fatalities. This distribution provides valuable insight into the severity of traffic accidents, highlighting that "Severe" accidents are the most common, followed by "Less Severe" and "Fatal" accidents.

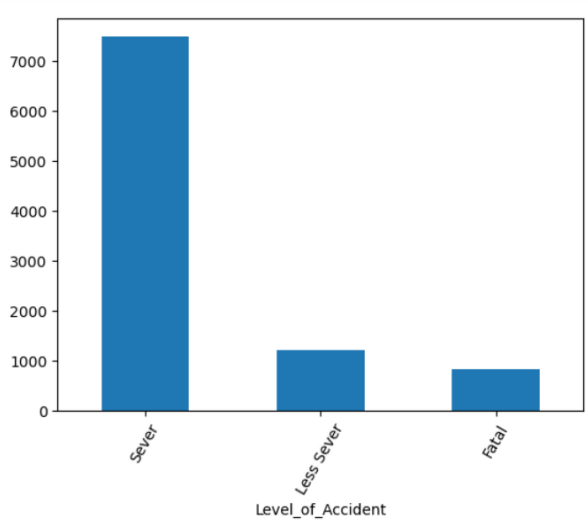


Figure 6-1: Class Distribution of Level of Accident

The target class distribution appears to be imbalanced. This means that the number of accidents across different severity levels is not uniform.

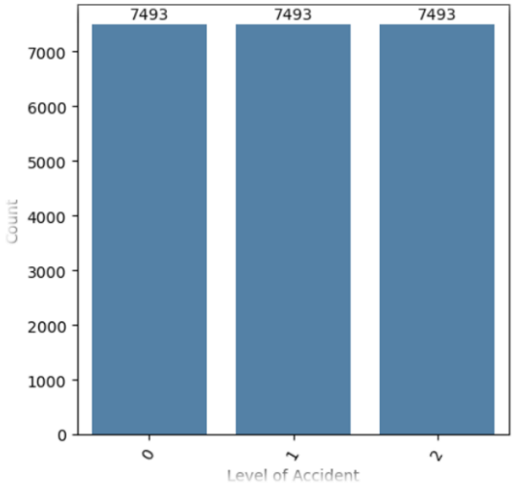


Figure 6-2: Distribution of Level of Accident after Resample

The dataset shows a significant skew towards 'Sever' accidents, with 7493 records, which constitutes approximately 78.61% of the data. 'Less Sever' accidents come in second at 12.72% (1212 records), followed by 'Fatal' accidents, the least frequent category, at 8.68% (827 records). This class imbalance can potentially affect the performance of machine learning models trained on this data, as they might be biased towards the majority class. In order to address this major issue one of the imbalance management technique resample has been used by the research and the result is shown in figure 6-2.

6.1 Association Rule Mining Result

Association rule mining is a crucial data mining technique used to uncover interesting relationships hidden in large datasets. In the context of traffic accidents, understanding the factors contributing to different severity levels of accidents can help in developing effective strategies for prevention and mitigation. This analysis explores the association rules extracted from traffic accident data, focusing on three levels of accident severity: less severe (Level_of_Accident_0), severe (Level_of_Accident_1) and fatal (Level_of_Accident_2).

Using the Apriori algorithm, we generated association rules with a minimum support threshold of 0.1 and a minimum confidence threshold of 0.5. The analysis focused on rules where the consequent was one of the accident severity levels (Level_of_Accident_0, Level_of_Accident_1, Level_of_Accident_2).Less severe accidents (Level_of_Accident_0) are frequently associated with dry road conditions. Specifically, if the road condition is dry (Road Condition Dry), the likelihood of the accident being less severe (Level_of_Accident_0) increases. This is supported by a support value of 0.101762, a confidence value of 13.56%, and a lift value of 1.066510. The analysis indicates that dry road conditions correlate with a higher probability of less severe accidents, as suggested by the confidence value. A lift value greater

than 1 implies a positive correlation between dry road conditions and less severe accidents.

Severe accidents (Level_of_Accident_1) are commonly associated with the absence of road junctions. For example, if the road junction type is no junction (Road Junction Type No Junction), the probability of the accident being severe (Level_of_Accident_1) is high. This is reflected in a support value of 0.524864, a confidence value of 78.97%, and a lift value of 1.004644. The analysis highlights that accidents occurring at locations with no road junctions tend to be severe, with a high confidence value indicating a significant likelihood of severe accidents in such scenarios. The lift value close to 1 suggests this is a common pattern in the dataset.

Fatal accidents (Level of Accident 2) are strongly associated with poor lighting conditions. Specifically, if the light condition is poor (Light Condition Poor), the likelihood of the accident being fatal (Level_of_Accident_2) increases. This is supported by a support value of 0.067564, a confidence value of 20.41%, and a lift value of 1.455678. The analysis shows that poor lighting conditions have a strong association with fatal accidents. The confidence value indicates that 20.41% of accidents under poor lighting conditions result in fatalities, and the lift value significantly greater than one underscore the strong impact of poor lighting on the severity of accidents.

6.2 Accident Severity Predication Model Evaluation Result

This section analyzes how well different models predict accident severity. The research used machine learning algorithms like Random Forest, Support Vector Machine, and Extreme Gradient Boosting (mentioned in section 3.5) because they're successful for classifying data and use different methods. To assess each model, standard metrics like precision, recall, F1-score, and accuracy were used. Additionally, confusion matrices were created to see how the models performed for various accident severity categories. These matrices provide a detailed picture of the models' strengths and weaknesses, including their ability

to correctly identify different severities. This detailed examination allows us to understand not only the overall performance of the models but also their specific strengths and weaknesses in predicting accident severity.

6.3 Model Evaluation Using Classification Matrix

6.3.1 Result of Classification Metrics for RF

Vehicle traffic accident predictive model using Random Forest has been implemented and executed. Its performance was evaluated using classification matrix and gave the following result. As shown in Table 6.1 below, random forest performed to the aggregate accuracy of 79.02% in predicating the severity of vehicle traffic accident. Regarding the individual categories of the severity level it shown 39% precision, 11% recall and 17% f1-force for category '0' which is fatal. Similarly, 81% precision, 96% recall and 88% f1-force for category '1' which is less severe accident. And also 56% precision, 19% recall and 28% f1-force for category '2' which are severe vehicle traffic accident. The weighted average of RF model is 74% precision, 79% recall and 74% f1-force. This result of RF is shown in Figure 6.3

Table 6-1: Classification Report for Random Forest Model

```
Encoded labels: [0 1 2]
Accuracy in Decimal: 0.7902464604090194
Accuracy in Percentage: 79.02%
Classification Report:
              precision    recall  f1-score   support

           0       0.39      0.11      0.17       223
           1       0.81      0.96      0.88      1511
           2       0.56      0.19      0.28       173

    accuracy                           0.79      1907
   macro avg       0.59      0.42      0.45      1907
weighted avg       0.74      0.79      0.74      1907
```

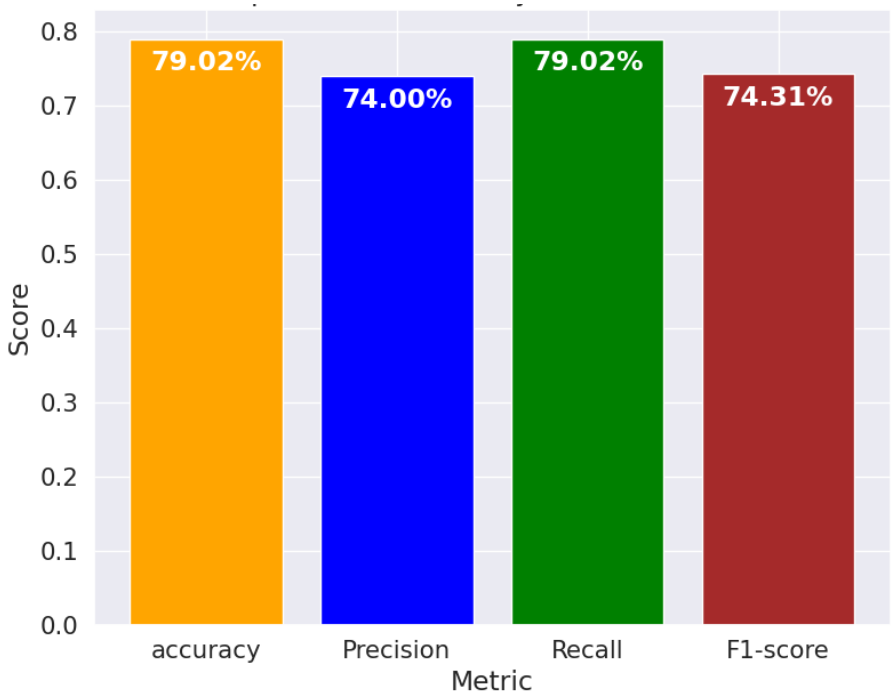


Figure 6-3: Classification Report of RF Model

6.3.2 Result of Classification Metrics for XGBoost

XGBoost Predictive model has been implemented and executed. Its performance was evaluated using classification matrix and gave the following result. As shown in Table 6.2 below, XGBoost performed to the aggregate accuracy of 84.48% in predicating the severity of vehicle traffic accident. Regarding the individual categories of the severity level it shown 0.80 precision, 0.35 recall and 0.49 f1-force for category '0' which is fatal.

Table 6-2 Classification Report for XGBoost Model:

Classification Report:

	precision	recall	f1-score	support
0	0.80	0.35	0.49	223
1	0.85	0.98	0.91	1511
2	0.75	0.32	0.45	173
accuracy			0.84	1907
macro avg	0.80	0.55	0.62	1907
weighted avg	0.84	0.84	0.82	1907

Similarly, 0.85 precision, 0.98 recall and 0.91 f1-force for category ‘1’ which is less sever accident. And also 0.75 precision, 0.32 recall and 0.45 f1-force for category ‘2’ which is sever vehicle traffic accident. The weighted average of XGBoost model is 84% precision, 84% recall and 72% f1-force. This result of XGBoost is shown in Figure 6-4.

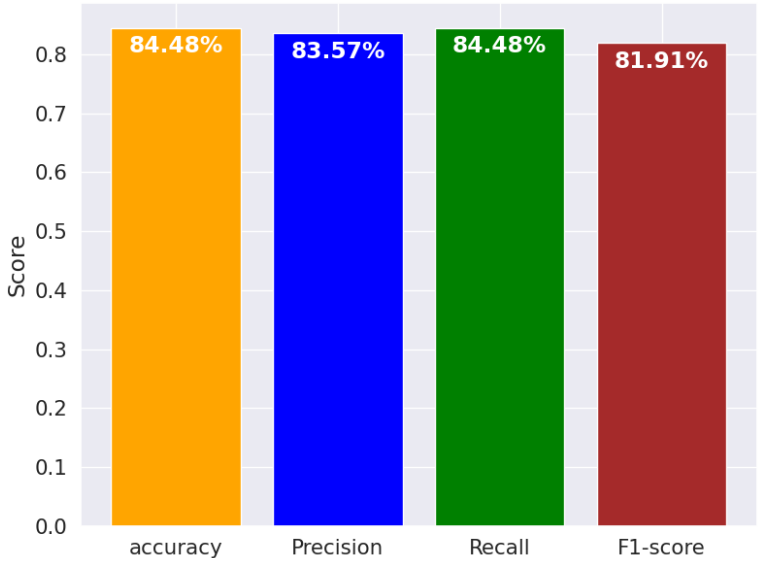


Figure 6-4: Classification Result of XGBoost Model

6.3.3 Result of Classification Metrics for SVM

Predication model using SVM has been implemented and executed. Its performance was evaluated using classification matrix and gave the following result. As shown in Table 6.2 below, SVM performed to the aggregate accuracy of 80% in predicating the severity of vehicle traffic accident. Regarding the

Table 6-3: Classification Report for SVM

Classification Report:

	precision	recall	f1-score	support
0	0.59	0.09	0.16	223
1	0.80	0.99	0.89	1511
2	1.00	0.07	0.13	173
accuracy			0.80	1907
macro avg	0.80	0.38	0.39	1907
weighted avg	0.80	0.80	0.73	1907

individual categories of the severity level it shown 0.59 precision, 0.09 recall and 0.16 f1-force for category ‘0’ which is fatal. Similarly, 0.8 precision, 0.99 recall and 0.89 f1-force for category ‘1’ which is less sever accident. And also 1.00 precision, 0.07 recall and 0.13 f1-force for category ‘2’ which is sever vehicle traffic accident. The weighted average of SVM model is 80% precision, 80% recall and 74% F1-force. This result of SVM is shown in Figure 6-3.

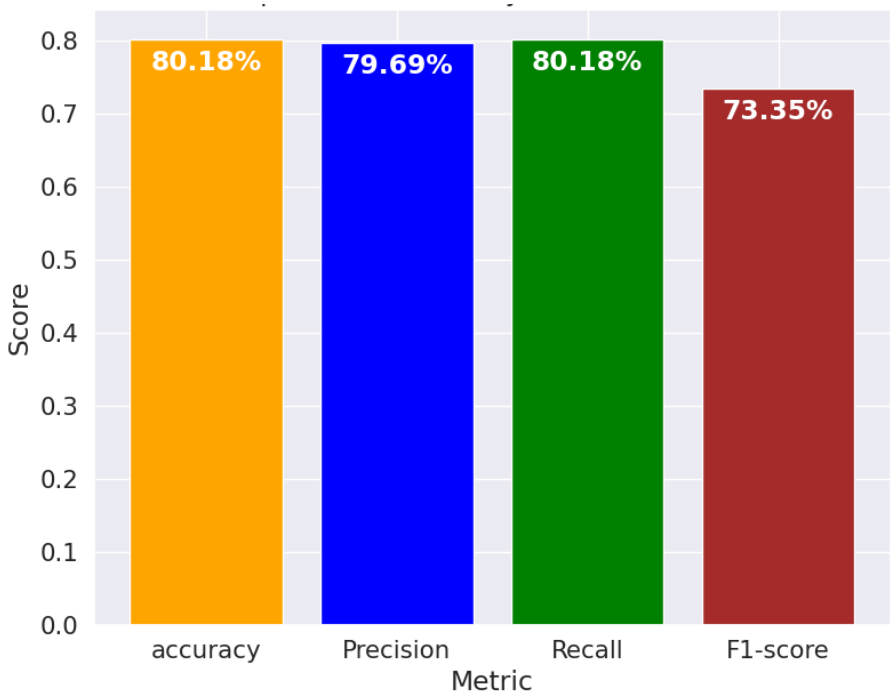


Figure 6-5: Classification Report of SVM Model

6.3.4 Result of Classification Metrics for Ada Boost

A prediction model using Adaboost has been implemented and executed, with its performance evaluated using a classification matrix. The results are presented in Table 6.4. Adaboost achieved an overall accuracy of 80% in predicting the severity of vehicle traffic accidents. For the individual severity categories, the model showed varying performance metrics. In category '0' (fatal accidents), Adaboost achieved a precision of 0.42, a recall of 0.09, and an F1-score of 0.14. For category '1' (less severe accidents), the model exhibited a

precision of 0.80, a recall of 0.98, and an F1-score of 0.88. In category '2' (severe accidents), Adaboost attained a precision of 0.78, a recall of 0.10, and an F1-score of 0.18. The weighted average performance of the Adaboost model resulted in a precision of 0.76, a recall of 0.80, and an F1-score of 0.73. The macro average, which accounts for each category equally, showed a precision of 0.67, a recall of 0.39, and an F1-score of 0.40. These results are depicted in Figure 6-6

Table 6-4: Classification Report for Ada Boost

	precision	recall	f1-score	support
0	0.42	0.09	0.14	223
1	0.80	0.98	0.88	1511
2	0.78	0.10	0.18	173
accuracy			0.80	1907
macro avg	0.67	0.39	0.40	1907
weighted avg	0.76	0.80	0.73	1907

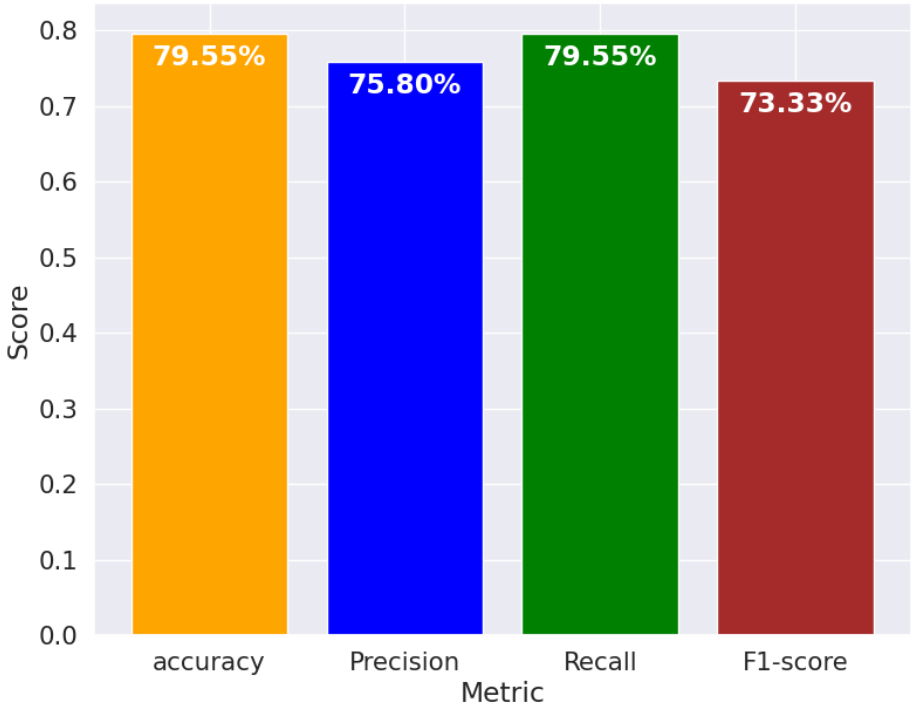


Figure 6-6: Classification Report for Ada Boost

6.3.5 Result of Classification Metrics for Bagging

A prediction model using Bagging with Support Vector Machine (SVM) has been implemented and executed. Its performance was evaluated using a classification matrix and yielded the following results. As shown in Table 6.5, the Bagging SVM model achieved an overall accuracy of 76% in predicting the severity of vehicle traffic accidents. For the individual categories of severity, the model demonstrated varied performance. For category '0', which represents fatal accidents, the precision was 0.31, recall was 0.22, and the F1-score was 0.26, with a support of 223 instances. For category '1', indicating less severe accidents, the model showed a precision of 0.82, recall of 0.89, and an F1-score of 0.86, with 1511 instances. For category '2', representing severe vehicle traffic accidents, the precision was 0.43, recall was 0.27, and the F1-score was 0.33, with a support of 173 instances. The weighted averages for the Bagging SVM model are 0.73 for precision, 0.76 for recall, and 0.74 for the F1-score, as depicted in Figure 6-7. The macro average across all categories is 0.52 for precision, 0.46 for recall, and 0.48 for the F1-score. This indicates that while the model performs well in predicting less severe accidents, there is room for improvement in accurately predicting fatal and severe accidents.

Table 6-5: Classification Report for Bagging

	precision	recall	f1-score	support
0	0.31	0.22	0.26	223
1	0.82	0.89	0.86	1511
2	0.43	0.27	0.33	173
accuracy			0.76	1907
macro avg	0.52	0.46	0.48	1907
weighted avg	0.73	0.76	0.74	1907

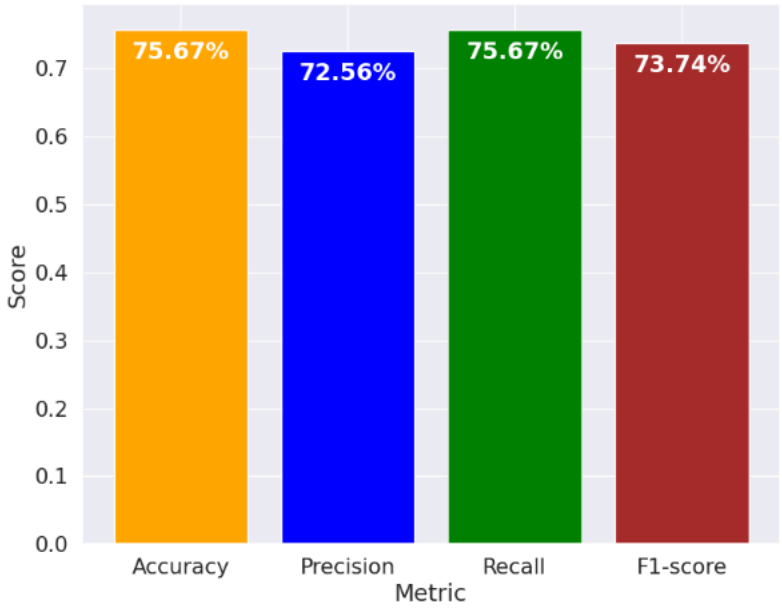


Figure 6-7: Classification Report for Bagging

6.3.6 Result of Classification Metrics for Naïve Bayes

A prediction model using Naïve Bayes has been implemented and executed. Its performance was evaluated using a classification matrix, yielding the following results. As shown in Table 6.6 below, Naïve Bayes achieved an aggregate accuracy of 79% in predicting the severity of vehicle traffic accidents. For the individual categories of severity, the model demonstrated a precision of 0.39, a recall of 0.25, and an F1-score of 0.30 for category '0', which represents fatal accidents. For category '1', representing less severe accidents, the model achieved a precision of 0.82, a recall of 0.95, and an F1-score of 0.88. For category '2', representing severe vehicle traffic accidents, the model showed a precision of 0.95, a recall of 0.10, and an F1-score of 0.19. The weighted average precision, recall, and F1-score for the Naïve Bayes model are 0.78, 0.79, and 0.75, respectively. These results are illustrated in Figure 6-8.

Table 6-6 Classification Report for Naïve Bayes

	precision	recall	f1-score	support
0	0.39	0.25	0.30	223
1	0.82	0.95	0.88	1511
2	0.95	0.10	0.19	173
accuracy			0.79	1907
macro avg	0.72	0.43	0.46	1907
weighted avg	0.78	0.79	0.75	1907

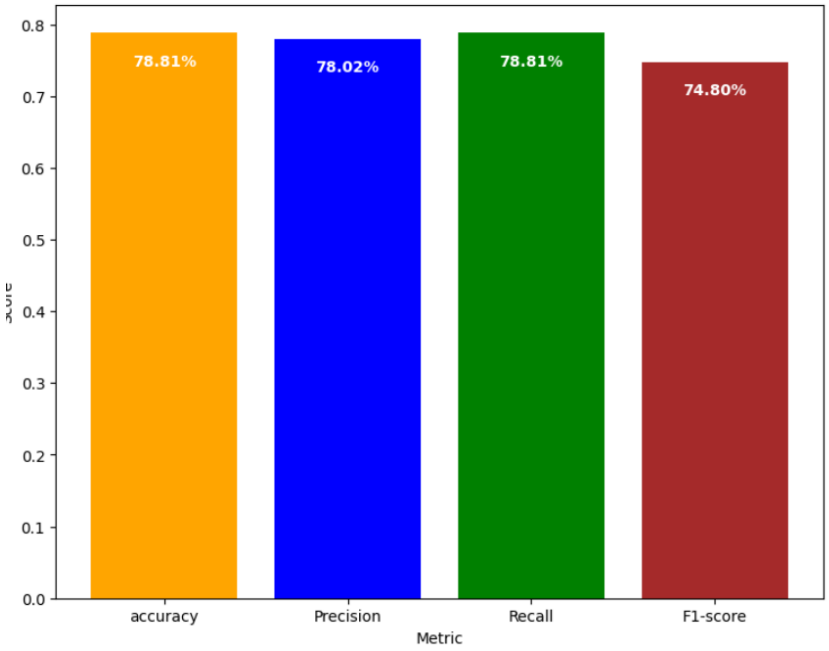


Figure 6-8: Classification Report for Naïve Bayes

6.3.7 Result of Classification Metrics for KNN

The prediction model using K-Nearest Neighbors (KNN) has been implemented and executed. Its performance was evaluated using a classification matrix and yielded the following results. As shown in Table 6.7 below, KNN achieved an overall accuracy of 80% in predicting the severity of vehicle traffic accidents. For the individual categories of severity levels, it demonstrated a precision of 0.52, a recall of 0.13, and an F1-score of 0.21 for category '0', which represents fatal accidents. Similarly, for category '1', representing less severe accidents, it showed a precision of 0.81, a recall of 0.98, and an F1-score of 0.89. For

category '2', which corresponds to severe vehicle traffic accidents, KNN achieved a precision of 0.76, a recall of 0.08, and an F1-score of 0.14.

The weighted average of the KNN model was 77% precision, 80% recall, and 74% F1-score. These results are illustrated in Figure 6-9.

Table 6-7: Classification Report for KNN

	precision	recall	f1-score	support
0	0.52	0.13	0.21	223
1	0.81	0.98	0.89	1511
2	0.76	0.08	0.14	173
accuracy			0.80	1907
macro avg	0.70	0.40	0.41	1907
weighted avg	0.77	0.80	0.74	1907

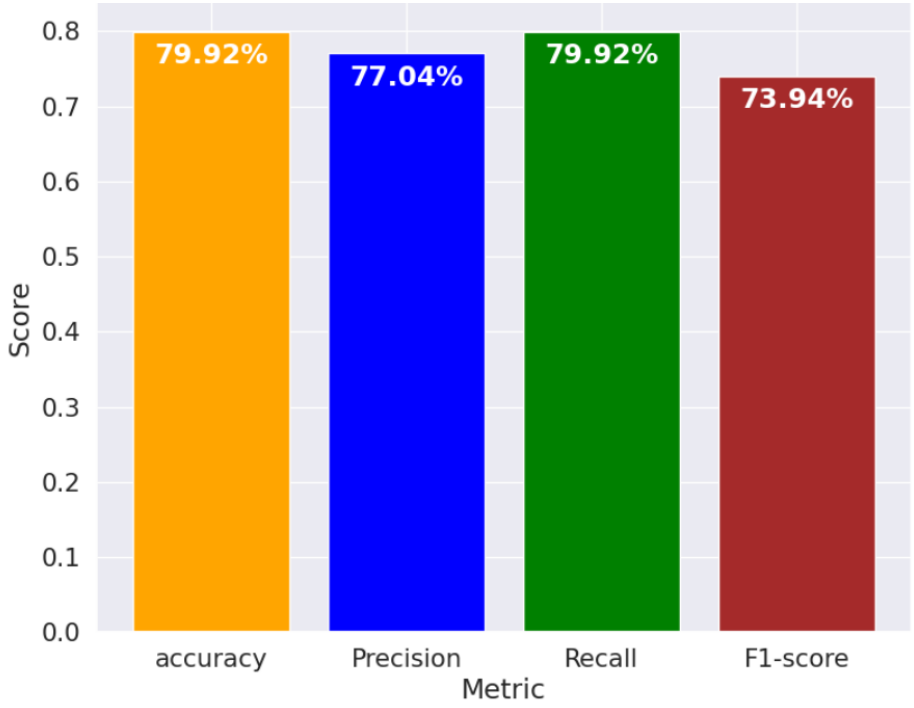


Figure 6-9: Classification Report for KNN

6.3.8 Result of Classification Metrics for Logistic Regression

The prediction model using logistic regression has been implemented and executed. Its performance was evaluated using a classification matrix and yielded the following results. As shown in Table 6.8 below, logistic regression achieved an overall accuracy of 80% in predicting the severity of vehicle traffic accidents.

Table 6-8: Classification Report for Logistic Regression

	precision	recall	f1-score	support
0	0.49	0.11	0.18	223
1	0.81	0.98	0.89	1511
2	0.85	0.13	0.23	173
accuracy			0.80	1907
macro avg	0.72	0.41	0.43	1907
weighted avg	0.78	0.80	0.75	1907

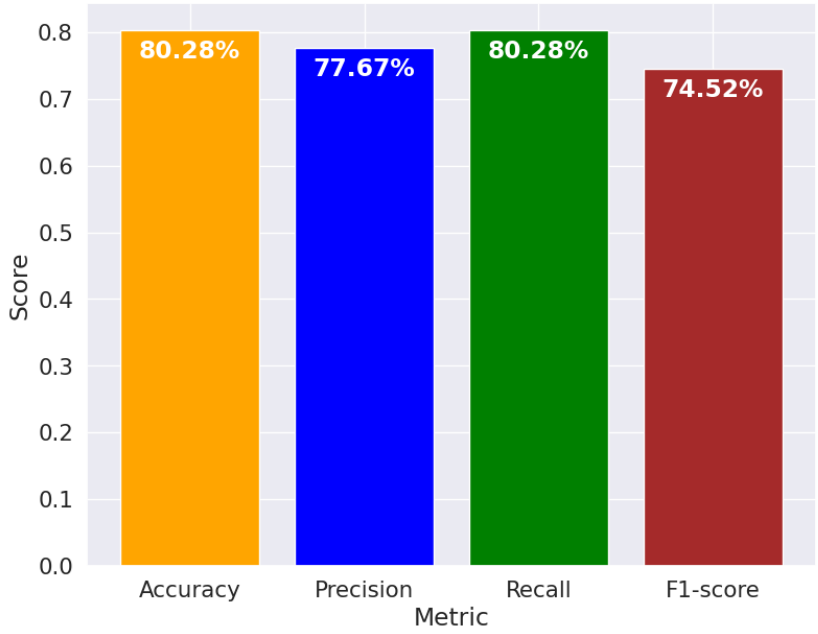


Figure 6-10: Classification Report for Logistic Regression

For the individual categories of severity, the model showed a precision of 0.52, a recall of 0.13, and an F1-score of 0.21 for category '0', which represents fatal accidents. For category '1', which corresponds to less severe accidents, the precision was 0.81, the recall was 0.98, and the F1-score was 0.89. Lastly, for category '2', indicating severe vehicle traffic accidents, the precision was 0.76, the recall was 0.08, and the F1-score was 0.14. The weighted averages for the logistic regression model were 0.77 for precision, 0.80 for recall, and 0.74 for the F1-score. These results are depicted in Figure 6-10.

6.3.9 Result of Classification Metrics for Decision Trees

The prediction model using Decision Trees has been implemented and executed. Its performance was evaluated using a classification matrix and yielded the following results. As shown in Table 6.9 below, the Decision Trees model achieved an overall accuracy of 72% in predicting the severity of vehicle traffic accidents. For the individual categories of severity levels, the model demonstrated a precision of 0.24, a recall of 0.26, and an F1-score of 0.25 for category '0', which corresponds to fatal accidents. For category '1', which represents less severe accidents, the model showed a precision of 0.82, a recall of 0.85, and an F1-score of 0.83. For category '2', representing severe vehicle traffic accidents, the model achieved a precision of 0.40, a recall of 0.25, and an F1-score of 0.31. The weighted averages for the Decision Trees model are 71% precision, 72% recall, and 72% F1-score. This performance of the Decision Trees model is illustrated in Figure 6-11.

Table 6-9: Classification Report for Decision Trees

	precision	recall	f1-score	support
0	0.24	0.26	0.25	223
1	0.82	0.85	0.83	1511
2	0.40	0.25	0.31	173
accuracy			0.72	1907
macro avg	0.49	0.45	0.46	1907
weighted avg	0.71	0.72	0.72	1907

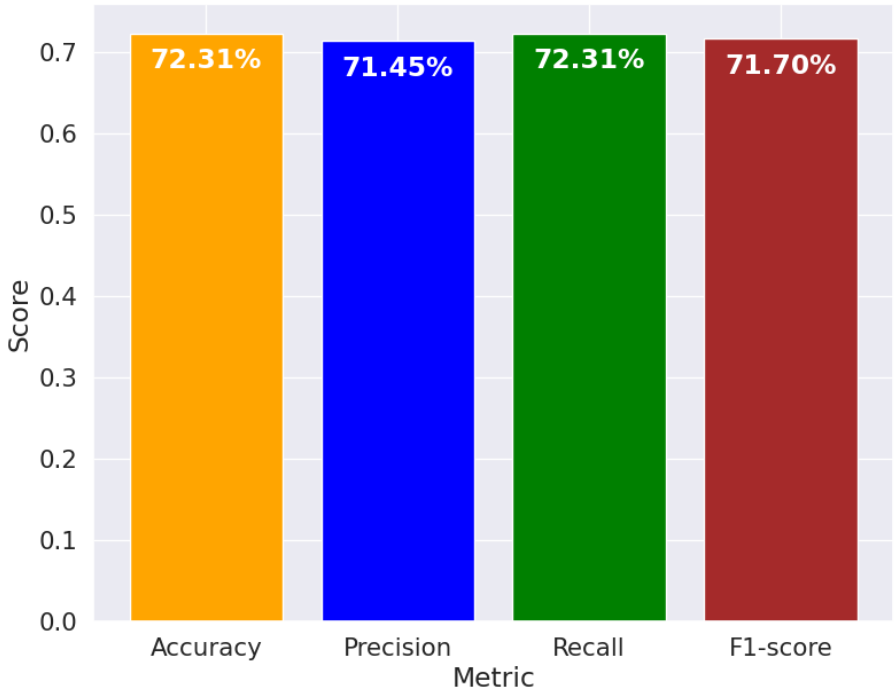


Figure 6-11: Classification Report for Decision Trees

6.3.10 Result of Classification Metrics for RNN

The evaluation metrics from the Recurrent Neural Network (RNN) model showcase varying performance across different accident severity levels. Notably, for Level 0 (less severe accidents), the precision is 0.50, indicating that when the model predicts this category, it is correct 50% of the time. However, the recall is low at 0.05, suggesting that the model fails to capture a significant portion of actual Level 0 accidents. The F1-score of 0.09 for this category underscores the difficulty in accurately identifying less severe accidents. Conversely, for Level 1 (severe accidents), the model performs robustly with a precision of 0.80 and a high recall of 0.99, indicating that it effectively identifies and recalls a large majority of severe accidents. The F1-score of 0.89 for Level 1 underscores the model's strong performance in this category. For Level 2 (fatal accidents), the precision is 0.84, indicating a good ability to correctly identify fatal accidents when predicting this category.

However, the recall is lower at 0.12, suggesting that the model misses a significant number of actual fatal accidents. The F1-score of 0.21 for Level 2 indicates moderate performance in accurately predicting fatal accidents. Overall, the weighted average F1-score of 0.73 across all categories signifies decent overall model performance. However, the macro average F1-score of 0.40 indicates disparities in performance across different severity levels, with the model excelling in identifying severe accidents but struggling with less severe and fatal accidents.

Table 6-10: Classification Report for RNN

	precision	recall	f1-score	support
0	0.50	0.05	0.09	223
1	0.80	0.99	0.89	1511
2	0.84	0.12	0.21	173
accuracy			0.80	1907
macro avg	0.71	0.39	0.40	1907
weighted avg	0.77	0.80	0.73	1907

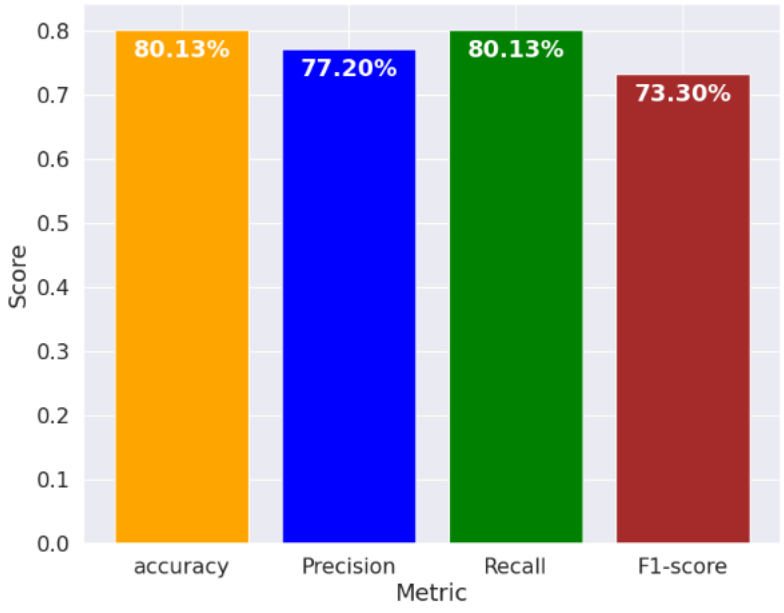


Figure 6-12: Classification Report for RNN

6.3.11 Summary of Classification Result for All Models

In this section, the research presents a comparative analysis of the performance of the three machine learning models utilized in predicting accident severity. Each metric provides a different perspective on the model's performance, offering a comprehensive view of their strengths and weaknesses. The results are summarized in the Table 6.11.

Table 6-11: Model Performance Result

Model	***Evaluation Metrics***			
	Accuracy	***Precision***	***Recall***	***F1 Score***
RF	***79***	***74***	***79***	***74***
XGBoost	***84***	***84***	***84***	***82***
SVM	***80***	***80***	***80***	***73***
Ada Boost	***79***	***78***	***79***	***75***
Bagging with SVM	***76***	***73***	***76***	***74***
Naïve Bayes	***79***	***78***	***78***	***75***
KNN	***80***	***77***	***80***	***74***
Logistic Regression	***80***	***78***	***80***	***75***
Decision Trees	***72***	***71***	***72***	***72***
RNN	***80***	***77***	***80***	***73***

6.4 Evaluation on Confusion Matrix

In this section, the research explores the detailed performance analysis of the models using the confusion matrix. The confusion matrix is a powerful tool that provides comprehensive insights into the classification performance by displaying the true positives, true negatives, false positives, and false negatives. By evaluating these metrics, we can better understand the accuracy, precision, recall, and F1 score of each model, highlighting their strengths and weaknesses in various scenarios. This analysis will guide us in identifying the most reliable model for our application and uncover areas where improvements are necessary.

6.4.1 Evaluation XGBoost on Confusion Matrix

Figure 6-13: shows, the confusion matrix for the XGBoost model shown in the image categorizes the predictions into three classes: Fatal (2), Less Severe (0), and Severe (1). The matrix is divided into three rows and three columns, representing the actual and predicted classifications respectively. The numbers in diagonal line 78, 1477 and 56 shows the correct predication of the mode each time. The numbers off the diagonal line shows the how much time wrong

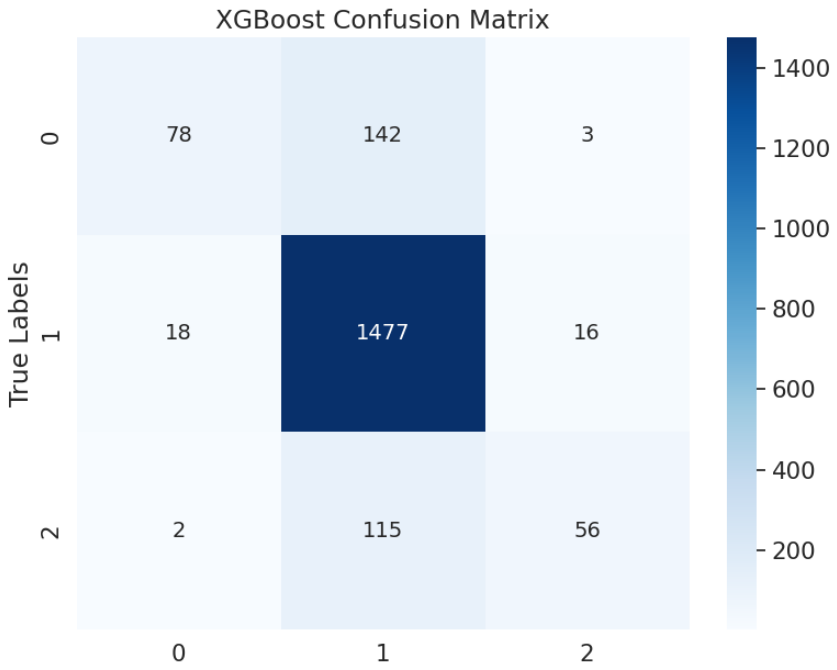


Figure 6-13: Confusion Matrix Result for XGBoost Model

predication is made. In this case the XGBoost model predict sever accident 1477 time out of 1499, less sever accident 78 times out of 223 and fatal accidents 56 times out of 165. This shows the model has good predication power on sever accidents. On the other hand less predication on less sever and fatal accidents.

6.4.2 Evaluation of RF on Confusion Matrix

Figure 6.14: shows, the confusion matrix for the Random Forest model it categorizes the predictions into three classes: Fatal (2), Less Severe (0), and Severe (1).

The matrix is divided into three rows and three columns, representing the actual and predicted classifications respectively. Here's a breakdown of the matrix: The numbers in diagonal line 25, 1449 and 33 shows the correct predication of the model each time. The numbers off the diagonal line shows the how much time wrong predication is made. In this case the RF model predicts sever accident 1449 time out of 1499; less sever accident 25 times out of 223 and fatal accidents 33 times out of 173. This shows the model has good predication power on sever accidents. On the other hand poor predication on less sever and fatal accidents. The overall performance of RF shows even less than XGBoost Model.

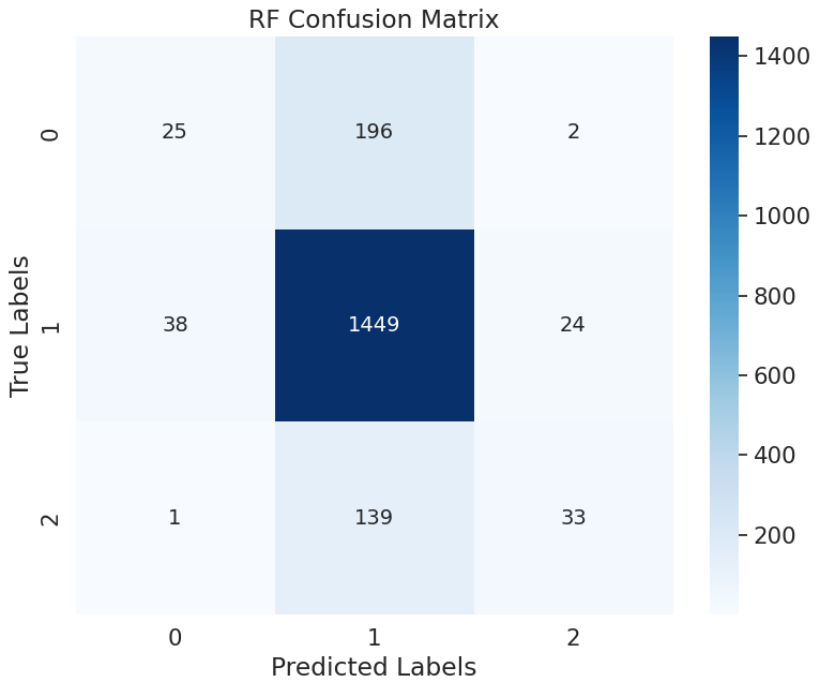


Figure 6-14: Confusion Matrix for RF Model

6.4.3 Evaluation of SVM on Confusion Matrix

Figure 6-15: shows, the confusion matrix for the SVM model shown in the image categorizes the predictions into three classes: Fatal (2), Less Severe (0), and Severe (1).

The matrix is divided into three rows and three columns, representing the actual and predicted classifications respectively. The numbers off the diagonal line shows the how much time wrong predication is made. In this case the SVM model predict sever accident 1497 time out of 1499, less sever accident 20 times out of 223 and fatal accidents 12 times out of 173. This shows the model has good predication power on sever accidents than even XGBoost. On the other hand poor predication on less sever and fatal accidents. The overall performance of SVM shows even mixed behavior predication.

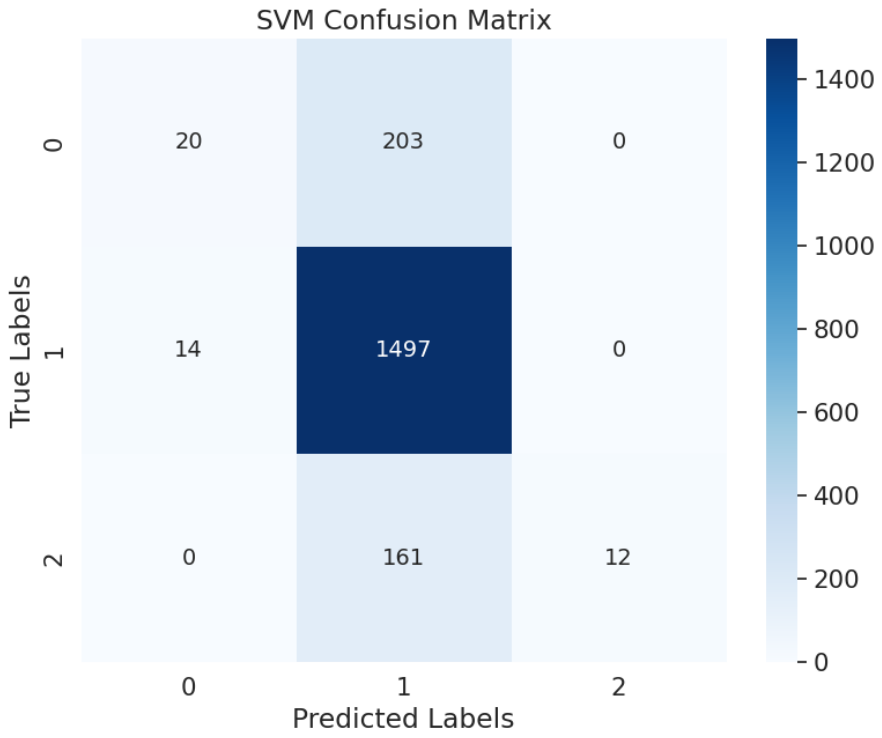


Figure 6-15: Confusion Matrix for SVM Model

6.4.4 Evaluation of Ada Boost on Confusion Matrix

Figure 6-16: shows, the confusion matrix for the Ada Boost model. In this case the Ada Boost model predict sever accident 1430 time out of 1511, less sever accident 55 times out of 223 and fatal accidents 18 times out of 173.

On the other hand it wrongly predicate 168 less sever accidents as sever and 148 fatal accidents as sever. The model has good predication on sever accident.

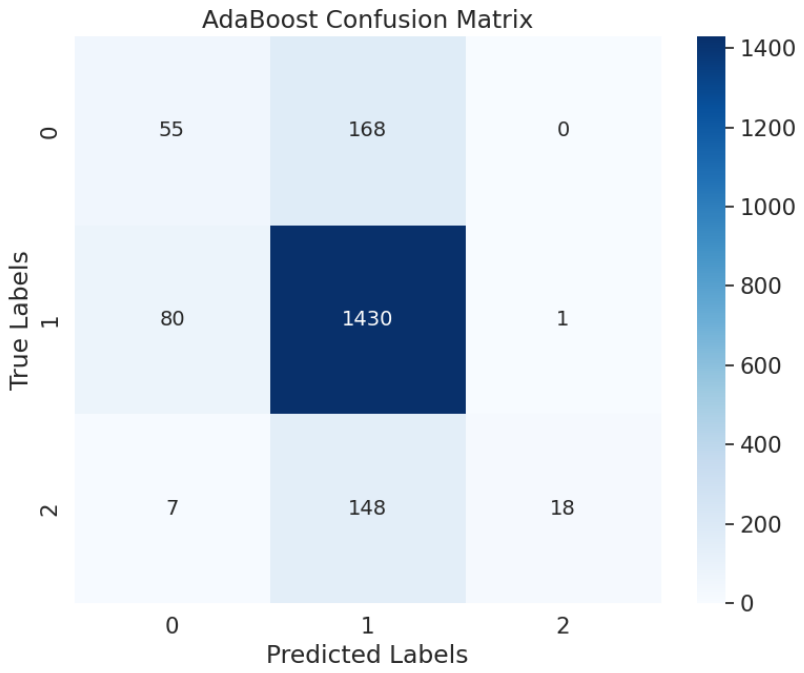


Figure 6-16: Confusion Matrix for Ada Boost Model

6.4.5 Evaluation of Bagging with SVM on Confusion Matrix

Figure 6-17 shows, the confusion matrix for the Bagging with SVM model. In this case the model predicts sever accident 1348 time out of 1511; less sever accident 49 times out of 223 and fatal accidents 46 times out of 173. On the other hand it wrongly predicate 170 less sever accidents as sever and 124 fatal accidents as sever. This model also has relatively good predication on sever accident.

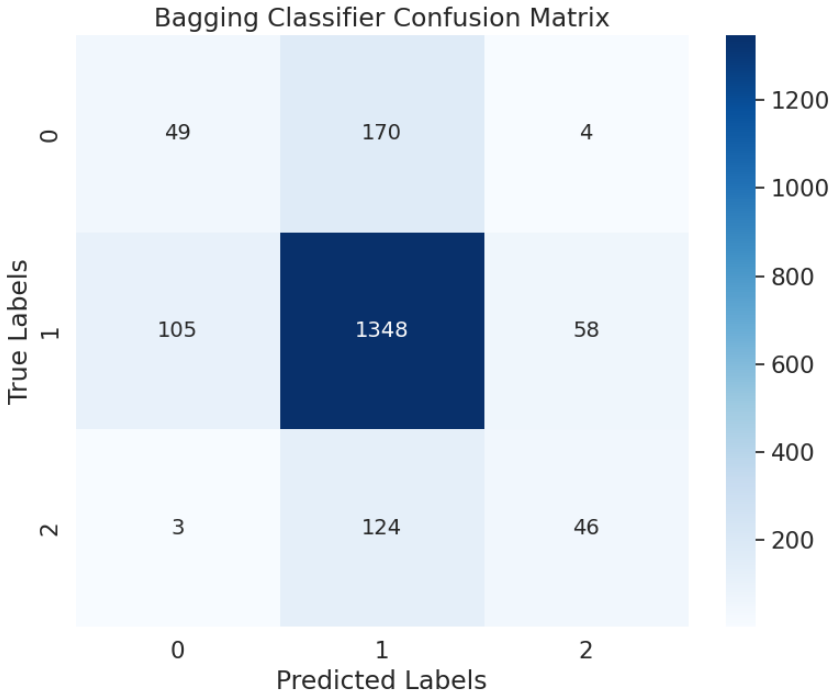


Figure 6-17: Confusion Matrix for Bagging with SVM Model

6.4.6 Evaluation of Naïve Bayes on Confusion Matrix

Figure 6-18: shows, the confusion matrix for Naïve Bayes model. In this case the Naïve Bayes model predict sever accident 1430 time out of 1511, less sever accident 55 times out of 223 and fatal accidents 18 times out of 173. On the other hand it wrongly predicate 168 less sever accidents as sever and 148 fatal accidents as sever.

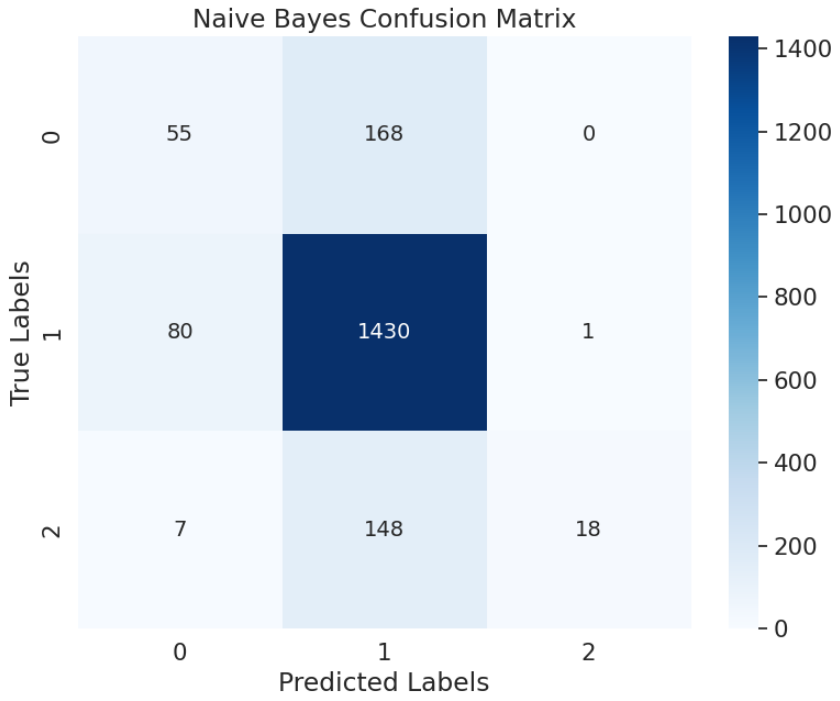


Figure 6-18: Confusion Matrix for Naïve Bayes Model

6.4.7 Evaluation of KNN on Confusion Matrix

Figure 6-19: shows, the confusion matrix for the KNN model. In this case the KNN model predict sever accident 1481 time out of 1511, less sever accident 30 times out of 223 and fatal accidents 13 times out of 173. On the other hand it wrongly predicate 192 less sever accidents as sever and 159 fatal accidents as sever. This model has poor predication on less sever accident and fatal accidents even if the overall performance is 80% accuracy.

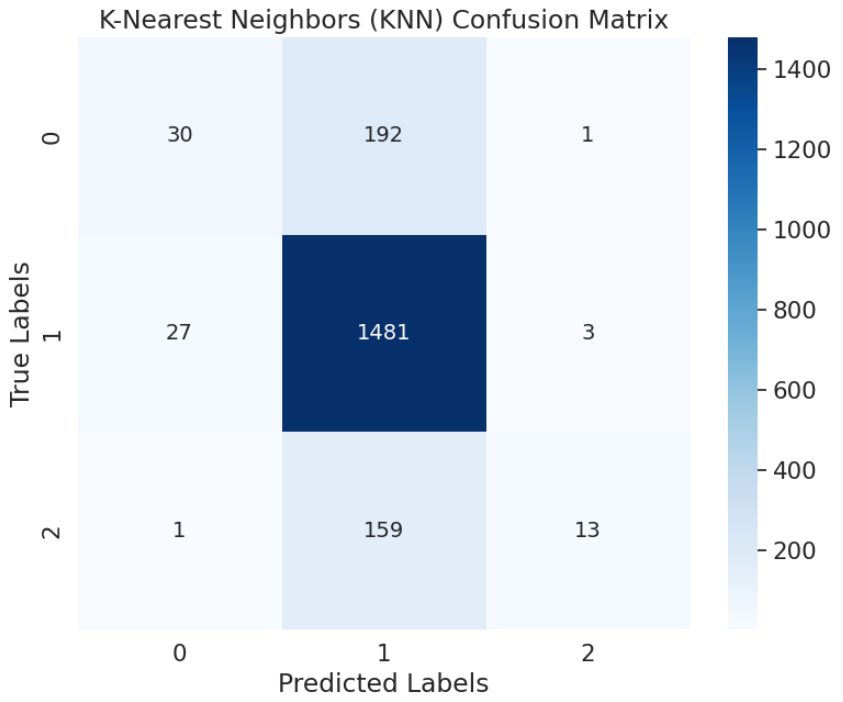


Figure 6-19: Confusion Matrix for KNN Model

6.4.8 Evaluation of Logistic Regression on Confusion Matrix

Figure 6-20 shows, the confusion matrix for the Logistic Regression model. In this case the Logistic Regression model predict sever accident 1484 time out of 1511, less sever accident 24 times out of 223 and fatal accidents 23 times out of 199. On the other hand it wrongly predicate 170 less sever accidents as sever and 148 fatal accidents as sever. This model also has good predication on sever accident

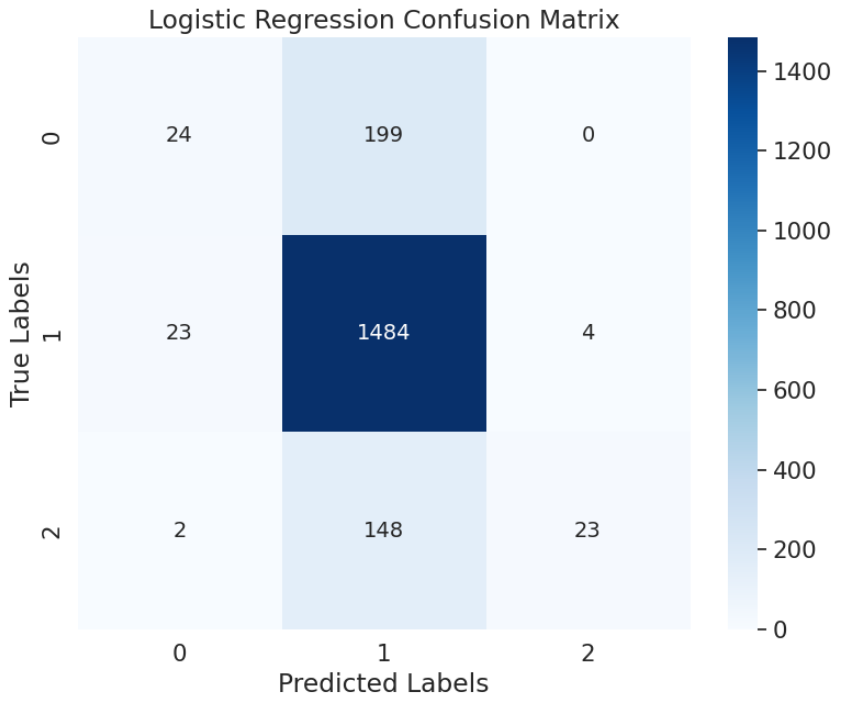


Figure 6-20: Confusion Matrix for Logistic Regression Model

6.4.9 Evaluation of Decision Trees on Confusion Matrix

Figure 6-21 shows, the confusion matrix for the Decision Trees model. In this case the model predicts sever accident 1277 time out of 1511; less sever accident 59 times out of 223 and fatal accidents 43 times out of 173. On the other hand it wrongly predicate 161 less sever accidents as sever and 119 fatal accidents as sever. This model also has poor in predication of sever accidents unlike all other models, however it is similar in predication of less sever and fatal accidents with the other models.

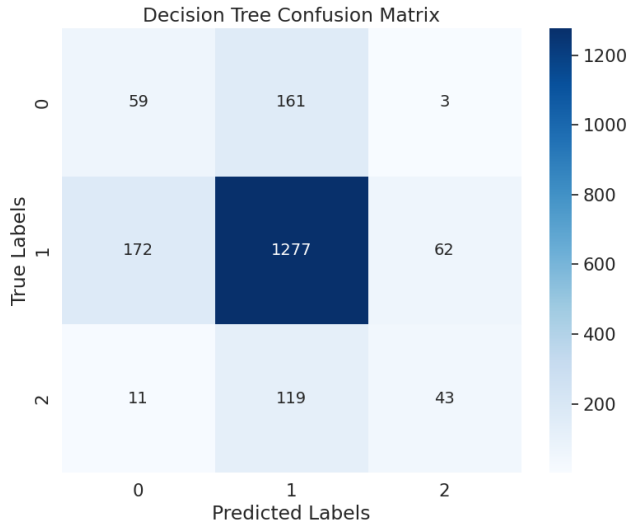


Figure 6-21: Confusion Matrix for Decision Trees Model

6.4.10 Evaluation of RNN on Confusion Matrix

Figure 6-22 shows, the confusion matrix for the RNN model. In this case RNN model predict sever accident 1496 time out of 1511, less sever accident 11 times out of 223 and fatal accidents 21 times out of 173. On the other hand it wrongly predicate 212 less sever accidents as sever and 152 fatal accidents as sever. This model also has very good predication on sever accident. But very poor predication on less sever and fatal accidents unlike all other models.

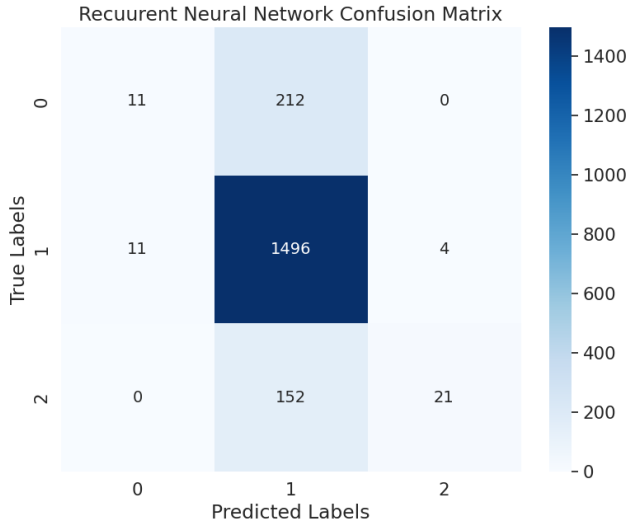


Figure 6-22: Confusion Matrix for RNN Model

6.5 Discussion

Analyzing the performance metrics across various machine learning models reveals valuable insights into their effectiveness for predicting and preventing traffic accidents. Among the ensemble methods, Random Forest (RF) and XGBoost demonstrate the highest overall performance with accuracies of 79% and 84%, respectively. These models excel in balancing accuracy, precision, recall, and F1 score, indicating their robustness in identifying both positive (accident-prone) and negative (non-accident-prone) instances. This capability is crucial for traffic accident prevention systems, where the ability to correctly identify potential risk factors and mitigate them early is vital. Models like AdaBoost and Bagging with SVM, although slightly lower in performance metrics, still show competitive results, emphasizing their utility as supplementary tools in comprehensive accident prediction frameworks.

Beyond machine learning metrics, the integration of these models into real-world traffic accident prevention strategies is essential. For instance, SVM and Naïve Bayes models, while showing good precision and recall, could be particularly useful in scenarios where identifying specific risk factors or anomalies in traffic patterns is critical. Logistic Regression and Decision Trees offer interpretable insights into the factors contributing to accidents, aiding policymakers and urban planners in implementing targeted interventions such as improved signage or traffic flow management. Moreover, the adaptability of models like Recurrent Neural Networks (RNNs) highlights their potential for capturing temporal dependencies in accident data, facilitating dynamic predictive capabilities that adjust to changing traffic conditions and behaviors.

In practical terms, the effectiveness of these models extends beyond their individual metrics. Their collective ability to analyze vast datasets, including historical accident records, weather conditions, and traffic volume, enables a proactive approach to accident prevention. By leveraging machine learning outputs, traffic authorities can prioritize high-risk areas for infrastructure

improvements, allocate resources more efficiently during peak accident periods, and implement targeted awareness campaigns for drivers. Ultimately, the synergy between advanced predictive models and traditional traffic management strategies not only enhances safety measures but also fosters a data-driven approach to reducing accidents and improving overall road safety for communities worldwide.

In addition this section will also address research results with research questions and objective drawn at the beginning of this research specifically in section 1.3 and 1.4 respectively.

RQ1: *How can machine learning techniques be used to identify the most significant contributing factors and their co-occurring patterns associated with traffic accident severity in Addis Ababa?*

Machine learning techniques can play a pivotal role in identifying the significant contributing factors and their co-occurring patterns associated with traffic accident severity in Addis Ababa. By leveraging large datasets, these techniques can uncover hidden relationships that are not immediately obvious through traditional analysis methods. For instance, algorithms like used in this research can be trained on historical traffic accident data to predict the severity of accidents based on various factors like road conditions, weather, pedestrian movement and vehicle types. These models can help identify which variables have the most substantial impact on accident outcomes, thereby providing insights into the primary risk factors.

In the context of traffic accident severity, association rule mining, such as the Apriori algorithm, can be particularly useful for discovering frequent patterns and co-occurring factors in the data. This method identifies rules that explain the likelihood of certain outcomes given specific conditions. For example, it can reveal that accidents are more likely to be severe when occurring at locations without road junctions or during poor lighting conditions. By setting minimum thresholds for support and confidence, the analysis can focus on the

most relevant and impactful rules, helping to prioritize factors that contribute significantly to accident severity.

Combining these approaches, machine learning not only helps in predictive modeling but also in deriving actionable insights from the data. The identification of significant factors and patterns can inform policy-making and strategic interventions aimed at reducing traffic accidents and mitigating their severity. For instance, improving lighting conditions in certain areas or redesigning road layouts could be targeted interventions based on the findings. Ultimately, machine learning techniques provide a data-driven foundation for developing more effective traffic safety measures, leading to safer roads in Addis Ababa.

RQ2: *How can insights gained from machine learning models be used to develop targeted interventions for reducing traffic accidents in Addis Ababa?*

Insights gained from machine learning models can be instrumental in developing targeted interventions for reducing traffic accidents in Addis Ababa. By identifying the most significant contributing factors to traffic accidents, policymakers and urban planners can design specific strategies to mitigate these risks. For instance, if models indicate that poor lighting conditions are strongly associated with fatal accidents; interventions such as improving street lighting in high-risk areas can be prioritized. Similarly, if certain road conditions or layouts are found to contribute to higher accident rates, infrastructure improvements like resurfacing roads or redesigning intersections can be implemented to enhance safety.

Furthermore, machine learning models can help in the allocation of resources and efforts towards the most impactful interventions. For example, if the analysis reveals that accidents are more frequent during certain times of the day or under specific weather conditions, targeted campaigns can be launched to raise awareness among drivers during these periods. Law enforcement can also be strategically deployed to monitor high-risk areas and times, thereby

preventing accidents through increased vigilance and rapid response. By focusing on the identified hotspots and high-risk scenarios, the overall effectiveness of safety measures can be maximized.

Additionally, continuous monitoring and updating of machine learning models with new data allow for the dynamic adjustment of interventions. As traffic patterns and urban landscapes evolve, these models can provide up-to-date insights, ensuring that safety measures remain relevant and effective. For example, if new construction or changes in traffic flow are observed, the models can help predict potential new risk factors, allowing for preemptive actions. This adaptive approach ensures that the strategies for reducing traffic accidents are not only based on historical data but are also responsive to current and emerging trends, leading to sustained improvements in road safety in Addis Ababa.

RQ3: *Which machine learning algorithms are most effective in predicting traffic accident severity in Addis Ababa?*

Based on the evaluation metrics of all the models trained, XGBoost is relatively better across all key performance indicators, achieving the highest accuracy, precision, recall, and F1 score among the evaluated models. With an accuracy of 84%, precision and recall both at 84%, and an F1 score of 82. This suggests that XGBoost's gradient boosting framework, which combines the strengths of multiple weak learners to enhance predictive accuracy, is particularly well-suited for handling the complexities of traffic accident data according to this research.

Support Vector Machine (SVM) and Logistic Regression also show good performance, making them effective alternatives for this prediction task. SVM achieves an accuracy, precision, and recall of 80%, with an F1 score of 73, indicating it is also usable in classifying accident severity. Logistic Regression, with similar metrics (80% accuracy, 78% precision, 80% recall, and an F1 score of 75), highlights its effectiveness as a linear model for binary classification

tasks. Both SVM and Logistic Regression offer high recall rates, which are crucial for ensuring that severe accidents are correctly identified, thereby enhancing the overall safety measures based on these predictions.

Other models like RNN, Ada Boost, Naïve Bayes, and KNN also show commendable performance, with accuracies around 79-80% and balanced precision and recall metrics. While Random Forest provides reliable results, Decision Trees fall short with the lowest accuracy and F1 score, indicating that more sophisticated ensemble methods like XGBoost and Ada Boost are better suited for this dataset.

6.6 Conclusion, Recommendations and Future Work

6.6.1 Conclusion

This research has explored the application of machine learning techniques to address the critical issue of traffic accidents in Addis Ababa, motivated by the alarming global and local statistics of traffic-related fatalities and injuries. Through comprehensive data collection, preprocessing, and analysis guided by specific research questions and objectives, this study has yielded several important insights. By leveraging machine learning algorithms, notably through the collection, preprocessing, and analysis of historical accident data, this research has demonstrated the potential for predicting the level of accidents with relatively high degree of accuracy and reliability.

Among the evaluated models, XGBoost relatively better achieving an accuracy of 84% in predicting accident severity. This model's relatively greater performance underscores its potential for improving traffic safety in Addis Ababa. Support Vector Machine (SVM) and Logistic Regression also showed strong predictive capabilities, with both models achieving an accuracy of 80%. These findings highlight the robustness of these algorithms in handling traffic accident data and their utility in developing reliable predictive systems for accident severity.

The significance of this study lies in its potential to inform targeted interventions by stakeholders such as the Addis Ababa Traffic Management Agency, Addis Ababa Police Commission, and the general public. By providing actionable insights into the factors contributing to accidents and the effectiveness of preventive measures, this research can guide resource allocation, policy formulation, and public awareness campaigns aimed at improving road safety. For instance, the model can help the Traffic Management Agency identify high-risk factors and locations, guiding targeted interventions like stricter enforcement, improved driver education, or infrastructure modifications. The Police Commission can use the model's risk predictions to inform resource allocation, allowing for more effective deployment of traffic police and CCTV cameras. Increased public awareness about high-risk factors can encourage safer driving practices, potentially reducing accident rates. Ultimately, this research paves the way for continued exploration and implementation of data-driven solutions, emphasizing the importance of interdisciplinary collaboration and ethical decision-making in addressing complex challenges, leading towards a future with fewer traffic accidents and their devastating consequences.

6.6.2 Recommendations

It is essential to establish a robust system for continuous and comprehensive data collection on traffic accidents. Collaboration between the Addis Ababa Traffic Management Agency (TMA) and Addis Ababa Police Commission (AAPC) should be strengthened to ensure the timely and accurate recording of accident data.

Continued efforts should be made to optimize the hyper parameters of machine learning models to achieve the best possible performance. It is highly recommended to develop and deploy real-time traffic management systems that utilize the predictive models to generate instant accident risk alerts. These systems can assist in proactive traffic management and timely intervention.

Integrating these real-time systems with existing traffic management infrastructure, such as traffic signals and electronic signage, can dynamically adjust traffic flow and reduce accident risks.

The insights gained from the predictive models can be used to design and implement targeted public awareness campaigns focused on the most significant risk factors identified, such as speeding and pedestrian safety. It is advisable to monitor and address any biases in the predictive models to ensure fair and equitable treatment of all demographic groups. Implement techniques such as fairness-aware learning algorithms to mitigate bias. Stringent data privacy standards should be maintained to protect personal information by anonymizing data where necessary and ensuring compliance with ethical guidelines and legal requirements. The research findings should be utilized to inform policy formulation aimed at improving road safety. Policies should address key factors contributing to accidents and prioritize interventions based on data-driven insights. Additionally, investing in infrastructure improvements such as better road signage, pedestrian crossings, and traffic calming measures in high-risk areas identified by the study should be prioritized.

Improving road lighting in poorly lit areas could significantly reduce the incidence of fatal accidents. Re-evaluating the design of roads and junctions to minimize the likelihood of severe accidents is crucial. Driver awareness programs should be implemented to educate drivers about the risks associated with specific conditions, such as driving in poorly lit areas or on roads without junctions. By implementing these measures, it is possible to mitigate the severity of traffic accidents and enhance overall road safety in Addis Ababa.

6.6.3 Future Work

While this research has made significant contributions to understanding and predicting traffic accidents in Addis Ababa using machine learning, there are

several areas where future research can build on these findings to further enhance road safety and traffic management.

One potential area of exploration is the extension of data analysis to incorporate a longer historical period, which could provide a more comprehensive understanding of traffic accident trends and improve the robustness of predictive models. Additionally, integrating a wider range of data sources, such as real-time traffic flow, weather conditions, and socio-economic factors, can enhance model accuracy and provide deeper insights into the complex interplay of variables contributing to accidents.

Cross-regional studies comparing traffic accident patterns and model performance in different cities or regions would be valuable in validating the generalizability of the models developed and adapting them to diverse contexts.

Finally, exploring the practical implementation of real-time accident risk alert systems and their integration with existing traffic management infrastructure could provide tangible benefits in reducing accident rates and enhancing road safety. These future directions underscore the importance of continuous innovation and interdisciplinary collaboration in the mission to create safer and more efficient transportation systems.

REFERENCES

Abdul, S. M. (2021). Accident Prediction Model Using Machine Learning: Accuracy of Predicted Model (Master's thesis). UiT The Arctic University of Norway, Faculty of Science and Technology.

Aboulola, O. I. (2024). Improving traffic accident severity prediction using MobileNet transfer learning model and SHAP XAI technique. PLoS ONE, 19(4), e0300640. https://doi.org/10.1371/journal.pone.0300640

Aggarwal, C. C. (2018). Neural networks and deep learning. Switzerland: Springer International Publishing. pp. 1-5.

Ayana Abdi, T., Hailu, B. H., Van Gelder, P. H. A. J. M., & Hagenzieker, M. P. (2017). Road Crashes in Addis Ababa, Ethiopia: Empirical Findings between the Years 2010 and 2014. African Research Review, 11(2). https://doi.org/10.4314/afrrev.v11i2.1

Bell, M. G. H., Bonsall, P. W., Leake, G. R., May, A. D., Nash, C. A., & O'Flaherty, C. A. (1997). Transport Planning and Traffic Engineering. Butterworth-Heinemann.

Beshah, T., Ejigu, D., Abraham, A., Snášel, V., & Krömer, P. (2012, January). Knowledge discovery from road traffic accident data in Ethiopia: Data quality, ensembling and trend analysis for improving road safety. Addis Ababa University, Addis Ababa, Ethiopia. Retrieved from ResearchGate.

Beyene Gerbi, H. (2020). Statistical analysis of road traffic accident in Addis Ababa: Application of SARIMA and SETAR model. Master's thesis, Jimma University.

Bloomberg Philantrpophs.Intiative for Road Saftey(2021 and 2023). Addis Ababa Road Safety Report (2023). Addis Ababa: Bloomburg Philantrophy Initiative.

Brownlee, J. (2018). Machine Learning Algorithms from Scratch (v1.8). Jason Brownlee.

Burkov, A. (2019). The Hundred-Page Machine Learning Book. Andriy Burkov.

David Levinson, Henry Liu, William Garrison, Adam Danczyk, Michael Corbett (2009). *Fundamentals of Transportation (Version 1.2).* USA Free Software Foundation, Inc.

Debela Deme Jima (2019) *Road Traffic Accident in Ethiopia from 2007/08-2017/18*. University of Gondar Ethiopia: American Center of Science and Education, USA

Deme, D. (2019). Road traffic accident in Ethiopia from 2007/08-2017/18. American International Journal of Sciences and Engineering Research, 2(2).

Deneke, S. S., & Endalkachew, A. A. (2020). Predicting factors contributing to road traffic accident and implying driver's driving behavior in Addis Ababa City. In The 6th Annual ACIST Proceedings. Kennesaw State University,

Dominic Zaal,(1994). *Traffic Law Enforcement: A Review of the Literature (Report No. 53).* Netherlands*:* Institute for Road Safety Research

Elias, A., & [Second Author]. (2020). Predicting Factors of Vehicle Traffic Accidents Using Machine Learning Algorithms: In the Case of Wolaita Zone. International Journal of Scientific Research in Computer Science Engineering and Information Technology.

Elias, A., & Sundado, T. S. (2020). Predicting Factors of Vehicle Traffic Accidents Using Machine Learning Algorithms: In the Case of Wolaita Zone. International Journal of Scientific Research in Computer Science Engineering and Information Technology, 8(4), 105-115. https://doi.org/10.21275/ART20204243

Emenike, *G. &* Akpu, *D., (*2017*). Assessment of Road Traffic Violations in Port Hacourt* Metropolis, Nigeria: International Journal of New Technology and Research (IJNTR)

Endalie, D., & Abebe, W. T. (2023). Analysis and detection of road traffic accident severity via data mining techniques: Case study Addis Ababa, Ethiopia [Article 6536768]. Mathematical Problems in Engineering, 2023.

Federal Transport Authority, *(*2017). *National Traffic Offences Statistics.* Addis Ababa: Federal Transport Authority.

International Traffic Safety Data and Analytics Group. (2023). Road Safety Annual Report 2023: International Transport Forum

International Transport Forum (2020). *Road Safety Annual Report*. Paris: International Transport Forum

J Peden, M., Scurfield, R., Sleet, D., Mohan, D., Hyder, A. A., Jarawan, E., & Mathers, C. (Eds.). (2004). World report on road traffic injury prevention. World Health Organization: Geneva.ung, A. (2022). Machine Learning: The Basics. Singapore: Springer.

Jarvinen, P. and Siltanen, P. (2021). Data Analytics and Machine Learning. VTT Technical Research Centre of Finland Ltd., Espoo, Finland. Kirschenbaum (Ed.), Institute for Applied Informatics (InfAI), University of Leipzig, Germany (Ed.), pp. 132 - 139

Kelleher, J. D., Mac Namee, B., & D'Arcy, A. (2015). Fundamentals of machine learning for predictive data analytics: Algorithms, worked examples, and case studies. The MIT Press: Cambridge, Massachusetts; London, England. pp. 1-50

Levinson, D., Liu, H., Garrison, W., Danczyk, A., & Corbett, M. (2009). Fundamentals of Transportation. Retrieved from http://code.pediapress.com/

M.G.H. Bell, P.W. Bonsall, G.R. Leake, A.D. May, C.A. Nash and C.A. O'Flaherty (1997). *Transport Planning and Traffic Engineering* (6ht ed.). : Elsevier Ltd

Michael D. Meyer (2016). *Transportation Planning Handbook* (Fourth Edition). Institute of Transportation Engineers: John Wiley & Sons, Inc., Hoboken, New Jersey.

Mohammed, M., Khan, M. B., & Bashier, E. B. M. (2017). Machine learning: Algorithms and applications. CRC Press.

Mosisa, G. L. (2018, October). Analysis of road traffic violations in Addis Ababa City (The case of Arada Sub-City. Retrieved from ResearchGate

Nicola Christie, RCPH, UK; Linda Drupsteen, TNO, The Netherlands; Jakko van Kampen, TNO, The Netherlands; Lottie Kuijt-Evers, TNO, The Netherlands; Ellen Schmitz-Felten, KOOP, Germany Marthe Verjans, Prevent, Belgium (2010). *A review of accidents and injuries*. Sarah Copsey European Agency for Safety and Health at Work (EU-OSHA)

Pourroostaei Ardakani, S., Liang, X., Mengistu, K. T., So, R. S., Wei, X., He, B., & Cheshmehzangi, A. (2023). Road Car Accident Prediction Using a Machine-Learning-Enabled Data Analysis. Switzerland. https://doi.org/10.3390/su15075939

Russell, R. (2018). Machine Learning: Step-by-Step Guide to Implement Machine Learning Algorithms with Python. CreateSpace Independent Publishing Platform.

Sakashita, C. Fleiter, J.J, Cliff, D., Flieger, M., Harman, B. & Lilley, M (2021). *A Guide to the Use of Penalties to Improve Road Safety. Global Road Safety Partnership, Geneva, Switzerland.*

Santos, D., Saias, J., Quaresma, P., & Nogueira, V. B. (2021). Machine Learning Approaches to Traffic Accident Analysis and Hotspot Prediction. Computers, 10, 157.

Smola, A. J., & Vishwanathan, S. V. N. (2008). Introduction to Machine Learning. Cambridge, United Kingdom: Cambridge University Press.

Theobald, O. (2017). Machine Learning For Absolute Beginners (2nd ed.). pp. 8 -115

Vital Strategies, Bloomberg Philanthropies Initiative for Global Road Safety. (2021). Addis Ababa Annual Road Safety Report, 2020-2021.

Weerasekera, K. (2009). An Introduction to Traffic Engineering. The Open University of Sri Lanka.

World Health Organization. (2006). Road traffic injury prevention: Training manual (D. Mohan, G. Tiwari, M. Khayesi, & F. M. Nafukho, Authors). World Health Organization. ISBN: 92 4 154675 1.

APPENDICES

Appendix A: Code for RF Model

```
# Importing all the neccessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.preprocessing import LabelEncoder
from sklearn.utils import resample
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.metrics import classification_report
from sklearn.feature_selection import SelectKBest, chi2
import seaborn as sns
# Loading the dataset
df = pd.read_csv("/content/Accident_Dataset.csv")
#creating two different dataframe of majority and minority classes
df_majority = df[(df['Level_of_Accident']==1)]
df_minority1 = df[(df['Level_of_Accident']==0)]
df_minority2 = df[(df['Level_of_Accident']==2)]
# upsampling minority classes individually
df_minority_upsampled1 = resample(df_minority1,
                                  replace=True,
                                  n_samples= 7493,
                                  random_state=42)
df_minority_upsampled2 = resample(df_minority2,
                                  replace=True,
                                  n_samples= 7493,
                                  random_state=42)
# Combining majority class with upsampled minority classes
df_upsampled = pd.concat([df_minority_upsampled1, df_minority_upsampled2, df_majority])
# Categorical features to encode using one hot encoding
features =["Sub_City","Gender","Road_Condition","Age", "Date_Accident_Occur",
                       "Road_Junction_Type","Road_Pavement_Type","Road_Layout","Road_Division",
                       "Light_Condition","Type_Of_Accident","Weather_Condition","Pedstrain_Movment",
                       "Vehiecle_Type", "Vehicle_Plate_Number",
                       "Vehicle_Movment","Driver_Experiance","Offense_Type"]
```

```
# setting input features X and target y
X = df[features]
y = df['Level_of_Accident']
# using pandas get_dummies method for on-hot encoding
encoded_df = pd.get_dummies(X, drop_first=True)
encoded_df.shape
# creating labelencoder object
lb = LabelEncoder()
lb.fit(y)
y_encoded = lb.transform(y)
print("Encoded labels:",lb.classes_)
y_en = pd.Series(y_encoded)
# feature seleciton method using chi2 for categorical output, categorical input
fs = SelectKBest(chi2, k=50)
X_new = fs.fit_transform(encoded_df, y_en)
# Take the selected features
cols = fs.get_feature_names_out()
# convert selected features into dataframe
fs_df = pd.DataFrame(X_new, columns=cols)
fs_df.shape, y_en
```

-1-

```
# train and test split and building baseline model to predict target features
X_trn, X_tst, y_trn, y_tst = train_test_split(fs_df, y_en, test_size=0.2, random_state=42)
# modelling using random forest baseline
rf = RandomForestClassifier(n_estimators=800, max_depth=20, random_state=42)
rf.fit(X_trn, y_trn)
# predicting on test data
predics = rf.predict(X_tst)
accuracy = accuracy_score(y_tst, predics)
print("Accuracy in Decimal:", accuracy)
print("Accuracy in Percentage: %.2f%%" % (accuracy * 100.0))
# Generate the classification report
report = classification_report(y_tst, predics, output_dict=True)
print("Classification Report:\n", report)
# Extract precision, recall, and F1-score for overall performance (weighted average)
accuracy =  accuracy_score(y_tst, predics)
precision = report['weighted avg']['precision']
recall = report['weighted avg']['recall']
f1_score = report['weighted avg']['f1-score']
# Plot precision, recall, and F1-score
report = ['accuracy','Precision', 'Recall', 'F1-score']
values = [accuracy, precision, recall, f1_score]
colors = ['orange','blue', 'green', 'brown']
plt.figure(figsize=(9, 7))
bars = plt.bar(report, values, color=colors)
plt.xlabel('Metric')
plt.ylabel('Score')
plt.title('Classification Report for RF: Accuracy, Precision, Recall, and F1-score')
# Show each result on the bar graph
for bar, value in zip(bars, values):
    percentage = value * 100
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() - 0.05,
             f'{percentage:.2f}%', ha='center', va='bottom', color='white', fontweight='bold')
plt.savefig("RF_Classififcation_comparison.png")
plt.show()
# Generate the confusion matrix
matrix = confusion_matrix(y_tst, predics)
print("Confusion Matrix:\n", matrix)
classes = sorted(y.unique())
annot_kws={'fontsize':14}
# Create a color-coded confusion matrix
sns.set(font_scale=1.4)
plt.figure(figsize=(9,7))
sns.heatmap(matrix,annot=True,cmap='Blues', fmt="d", annot_kws=annot_kws,
            xticklabels=classes, yticklabels=classes)
```

```
# cbar=0, cmap="YlGnBu",linewidths=2, ax=ax0,vmax=3000, vmin=0
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("RF Confusion Matrix")
plt.savefig("RF_heatmap.png")
plt.show()
```

Appendix B: Code for XGBoost Model

```
# Importing all the neccessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.utils import resample
import xgboost as xgb
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.metrics import classification_report
from sklearn.feature_selection import SelectKBest, chi2
import seaborn as sns
# Loading the dataset
df = pd.read_csv("/content/Accident_Dataset.csv")
#creating two different dataframe of majority and minority classes
df_majority = df[(df['Level_of_Accident']==1)]
df_minority1 = df[(df['Level_of_Accident']==0)]
df_minority2 = df[(df['Level_of_Accident']==2)]
# upsampling minority classes individually
df_minority_upsampled1 = resample(df_minority1,
                                  replace=True,
                                  n_samples= 7493,
                                  random_state=42)
df_minority_upsampled2 = resample(df_minority2,
                                  replace=True,
                                  n_samples= 7493,
                                  random_state=42)
# Combining majority class with upsampled minority classes
df_upsampled = pd.concat([df_minority_upsampled1, df_minority_upsampled2, df_majority])
# Categorical features to encode using one hot encoding
features =["Sub_City","Gender","Road_Condition","Age", "Date_Accident_Occur",
                        "Road_Junction_Type","Road_Pavement_Type","Road_Layout","Road_Division",
                        "Light_Condition","Type_Of_Accident","Weather_Condition","Pedstrain_Movment",
                         "Vehiecle_Type", "Vehicle_Plate_Number",
                         "Vehicle_Movment","Driver_Experiance","Offense_Type"]

# setting input features X and target y
X = df[features]
y = df['Level_of_Accident']
# using pandas get_dummies method for on-hot encoding
encoded_df = pd.get_dummies(X, drop_first=True)
encoded_df.shape
# create labelencoder object
lb = LabelEncoder()
lb.fit(y)
y_encoded = lb.transform(y)
print("Encoded labels:",lb.classes_)
y_en = pd.Series(y_encoded)
# feature seleciton method using chi2 for categorical output, categorical input
fs = SelectKBest(chi2, k=50)
X_new = fs.fit_transform(encoded_df, y_en)
# Taking the selected features
cols = fs.get_feature_names_out()
# converting selected features into dataframe
fs_df = pd.DataFrame(X_new, columns=cols)
fs_df.shape, y_en
# split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(fs_df, y_en, test_size = 0.20,random_state=42)
```

```
# modelling using XGBoos Classifier
model = XGBClassifier()
model.fit(fs_df, y_en)
# make predictions for test data
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
print("Accuracy in Decimal:", accuracy)
print("Accuracy in Percentage: %.2f%%" % (accuracy * 100.0))
# Generate the classification report
report = classification_report(y_test, predictions, output_dict=True)
#report = classification_report(y_test, predictions)
print("Classification Report:\n", report)
# Extract precision, recall, and F1-score for overall performance (weighted average)
accuracy =  metrics.accuracy_score(y_test, y_pred)
precision = report['weighted avg']['precision']
recall = report['weighted avg']['recall']
f1_score = report['weighted avg']['f1-score']
# Plot precision, recall, and F1-score
report = ['accuracy','Precision', 'Recall', 'F1-score']
values = [accuracy, precision, recall, f1_score]
colors = ['orange','blue', 'green', 'brown']
plt.figure(figsize=(9, 7))
bars = plt.bar(report, values, color=colors)
plt.xlabel('Metric')
plt.ylabel('Score')
plt.title('Classification Report XGBoost: Accuracy, Precision, Recall, and F1-score')
# Show each result on the bar graph
for bar, value in zip(bars, values):
    percentage = value * 100
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() - 0.05,
             f'{percentage:.2f}%', ha='center', va='bottom', color='white', fontweight='bold')
plt.savefig("XGBoost_Classififcation_comparison.png")
plt.show()
# Generate the confusion matrix
matrix = confusion_matrix(y_test, predictions)
print("Confusion Matrix:\n", matrix)
classes = sorted(y.unique())
annot_kws={'fontsize':14}
# Create a color-coded confusion matrix
sns.set(font_scale=1.4)
plt.figure(figsize=(9,7))
sns.heatmap(matrix,annot=True,cmap='Blues', fmt="d", annot_kws=annot_kws,xticklabels=classes, yticklabels=classes)
# cbar=0, cmap="YlGnBu",linewidths=2, ax=ax0,vmax=3000, vmin=0
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("XGBoost Confusion Matrix")
plt.savefig("XGBoost_heatmap.png")
plt.show()
```

Appendix C: Code for SVM Model

```
# Importing all the neccessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import folium
from sklearn import svm
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import precision_score, recall_score, f1_score
import seaborn as sns
from sklearn.metrics import classification_report
from sklearn import metrics
from sklearn.feature_selection import SelectKBest, chi2
# Loading traffic accident data set prepared
df = pd.read_csv("/content/Accident_Dataset.csv")
# upsampling minority classes individually
df_minority_upsampled1 = resample(df_minority1,
                                  replace=True,
                                  n_samples= 7493,
                                  random_state=42)
df_minority_upsampled2 = resample(df_minority2,
                                  replace=True,
                                  n_samples= 7493,
                                  random_state=42)
# Combining majority class with upsampled minority classes
df_upsampled = pd.concat([df_minority_upsampled1, df_minority_upsampled2, df_majority])
# Categorical features to encode using one hot encoding
features =["Sub_City","Gender","Road_Condition","Age", "Date_Accident_Occur",
                        "Road_Junction_Type","Road_Pavement_Type","Road_Layout","Road_Division",
                        "Light_Condition","Type_Of_Accident","Weather_Condition","Pedstrain_Movment",
                       "Vehiecle_Type", "Vehicle_Plate_Number",
                        "Vehicle_Movment","Driver_Experiance","Offense_Type"]
# setting input features X and target y
X = df[features]
y = df['Level_of_Accident']
# using pandas get_dummies method for on-hot encoding
encoded_df = pd.get_dummies(X, drop_first=True)
encoded_df.shape
```

```
# create labelencoder object
lb = LabelEncoder()
lb.fit(y)
y_encoded = lb.transform(y)
print("Encoded labels:",lb.classes_)
y_en = pd.Series(y_encoded)
# feature seleciton method using chi2 for categorical output, categorical input
from sklearn.feature_selection import SelectKBest, chi2
fs = SelectKBest(chi2, k=18)
X_new = fs.fit_transform(encoded_df, y_en)
# Take the selected features
cols = fs.get_feature_names_out()
# convert selected features into dataframe
fs_df = pd.DataFrame(X_new, columns=cols)
fs_df.shape, y_en
X_train, X_test, y_train, y_test = train_test_split(fs_df, y_en, test_size = 0.20,random_state=42)
X_test_copy=X_test.copy()
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
clf = svm.SVC(kernel='poly', degree=2, C=1.0, gamma='scale', class_weight=None)
clf.fit(X_train, y_train)
# Assuming y_test and y_pred are defined
y_pred = clf.predict(X_test)
accuracy = metrics.accuracy_score(y_test, y_pred)
print("Accuracy in decimal:", accuracy)
print("Accuracy in percentage: %.2f%%" % (accuracy * 100.0))
report = classification_report(y_test, y_pred, output_dict=True)
print("Classification Report:\n", report)
```

```
# Generate the confusion matrix
matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", matrix)
classes = sorted(y.unique())
annot_kws={'fontsize':14}
# Extract precision, recall, and F1-score for overall performance (weighted average)
accuracy =  metrics.accuracy_score(y_test, y_pred)
precision = report['weighted avg']['precision']
recall = report['weighted avg']['recall']
f1_score = report['weighted avg']['f1-score']
# Plot precision, recall, and F1-score
report = ['accuracy','Precision', 'Recall', 'F1-score']
values = [accuracy, precision, recall, f1_score]
colors = ['orange','blue', 'green', 'brown']
plt.figure(figsize=(9, 7))
        bars = plt.bar(report, values, color=colors)
plt.xlabel('Metric')
plt.ylabel('Score')
plt.title('Classification Report SVM: Accuracy, Precision, Recall, and F1-score')
# Show each result on the bar graph
for bar, value in zip(bars, values):
    percentage = value * 100
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() - 0.05,
             f'{percentage:.2f}%', ha='center', va='bottom', color='white', fontweight='bold')
plt.savefig("SVM_Classififcation_comparison.png")
plt.show()
# Create a color-coded confusion matrix
sns.set(font_scale=1.4)
plt.figure(figsize=(9,7))
sns.heatmap(matrix,annot=True,cmap='Blues', fmt="d", annot_kws=annot_kws,xticklabels=classes, yticklabels=classes)
# cbar=0, cmap="YlGnBu",linewidths=2, ax=ax0,vmax=3000, vmin=0
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("SVM Confusion Matrix")
plt.savefig("SVM_heatmap.png")
plt.show()
```

Appendix D: Code for Ada Boost Model

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score
from sklearn.utils import resample
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.pipeline import Pipeline
# Loading the dataset
df = pd.read_csv("/content/Accident_Dataset.csv")

# Categorical features to encode using one hot encoding
features = ["Sub_City", "Gender", "Road_Condition", "Age", "Date_Accident_Occur",
            "Road_Junction_Type", "Road_Pavement_Type", "Road_Layout", "Road_Division",
            "Light_Condition", "Type_Of_Accident", "Weather_Condition", "Pedstrain_Movment",
            "Vehiecle_Type", "Vehicle_Plate_Number",
            "Vehicle_Movment", "Driver_Experiance", "Offense_Type"]
# Setting input features X and target y
X = df[features]
y = df['Level_of_Accident']
# Using pandas get_dummies method for one-hot encoding
encoded_df = pd.get_dummies(X, drop_first=True)
# Creating labelencoder object
lb = LabelEncoder()
y_encoded = lb.fit_transform(y)
print("Encoded labels:", lb.classes_)
y_en = pd.Series(y_encoded)
# Feature selection method using chi2 for categorical output, categorical input
fs = SelectKBest(chi2, k=50)
X_new = fs.fit_transform(encoded_df, y_en)
# Take the selected features
cols = fs.get_support(indices=True)
selected_features = encoded_df.columns[cols]
fs_df = pd.DataFrame(X_new, columns=selected_features)
# Train and test split
X_train, X_test, y_train, y_test = train_test_split(fs_df, y_en, test_size=0.2, random_state=42)
```

```
# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Build an AdaBoost classifier with Decision Trees as base estimators
base_estimator = DecisionTreeClassifier(max_depth=1)  # Weak learner
ada_boost_model = AdaBoostClassifier(base_estimator=base_estimator, n_estimators=50, random_state=42)
# Fit the AdaBoost model
ada_boost_model.fit(X_train, y_train)
# Evaluate the model
y_pred = ada_boost_model.predict(X_test)
# Generate classification report and confusion matrix
report = classification_report(y_test, y_pred)
print(classification_report(y_test, y_pred))
matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", matrix)
# Plot confusion matrix
sns.set(font_scale=1.4)
plt.figure(figsize=(9, 7))
sns.heatmap(matrix, annot=True, cmap='Blues', fmt="d", annot_kws={"fontsize": 14})
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("AdaBoost Confusion Matrix")
plt.savefig("AdaBoost_Confusion_Matrix.png")
plt.show()
```

```python
# Plot accuracy, precision, recall, and F1-score
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
report_labels = ['accuracy', 'Precision', 'Recall', 'F1-score']
values = [accuracy, precision, recall, f1]
colors = ['orange', 'blue', 'green', 'brown']
plt.figure(figsize=(9, 7))
bars = plt.bar(report_labels, values, color=colors)
plt.xlabel('Metric')
plt.ylabel('Score')
plt.title('Classification Report for AdaBoost: Accuracy, Precision, Recall, and F1-score')
# Show each result on the bar graph
for bar, value in zip(bars, values):
    percentage = value * 100
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() - 0.05,
             f'{percentage:.2f}%', ha='center', va='bottom', color='white', fontweight='bold')
plt.savefig("AdaBoost_Classification_Report.png")
plt.show()
```

Appendix E: Code for Bagging with SVM Model

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
# Loading the dataset
df = pd.read_csv("/content/Accident_Dataset.csv")
# Categorical features to encode using one hot encoding
features = ["Sub_City", "Gender", "Road_Condition", "Age", "Date_Accident_Occur",
            "Road_Junction_Type", "Road_Pavement_Type", "Road_Layout", "Road_Division",
            "Light_Condition", "Type_Of_Accident", "Weather_Condition", "Pedstrain_Movment",
            "Vehiecle_Type", "Vehicle_Plate_Number",
            "Vehicle_Movment", "Driver_Experiance", "Offense_Type"]
# Setting input features X and target y
X = df[features]
y = df['Level_of_Accident']
# Using pandas get_dummies method for one-hot encoding
encoded_df = pd.get_dummies(X, drop_first=True)
# Creating labelencoder object
lb = LabelEncoder()
y_encoded = lb.fit_transform(y)
print("Encoded labels:", lb.classes_)
y_en = pd.Series(y_encoded)
# Feature selection method using chi2 for categorical output, categorical input
fs = SelectKBest(chi2, k=50)
X_new = fs.fit_transform(encoded_df, y_en)
# Take the selected features
cols = fs.get_support(indices=True)
selected_features = encoded_df.columns[cols]
fs_df = pd.DataFrame(X_new, columns=selected_features)
# Train and test split
X_train, X_test, y_train, y_test = train_test_split(fs_df, y_en, test_size=0.2, random_state=42)
# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Show each result on the bar graph
for bar, value in zip(bars, values):
    percentage = value * 100
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() - 0.05,
             f'{percentage:.2f}%', ha='center', va='bottom', color='white', fontweight='bold')
plt.savefig("Bagging_Classifier_Classification_comparison.png")
plt.show()
# Plot confusion matrix
sns.set(font_scale=1.4)
plt.figure(figsize=(9, 7))
sns.heatmap(matrix, annot=True, cmap='Blues', fmt="d", annot_kws={"fontsize": 14})
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Bagging Classifier Confusion Matrix")
plt.savefig("Bagging_Classifier_Confusion_Matrix.png")
plt.show()
```

```
# Build Bagging ensemble model using Decision Tree as base estimator
base_estimator = DecisionTreeClassifier()
model = BaggingClassifier(base_estimator=base_estimator, n_estimators=50, random_state=42)
# Train the model
model.fit(X_train, y_train)
# Predict on test data
y_pred = model.predict(X_test)
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy on test set: {accuracy * 100:.2f}%")
# Generate classification report and confusion matrix
report = classification_report(y_test, y_pred, output_dict=True)
print(classification_report(y_test, y_pred))
matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", matrix)
# Extract precision, recall, and F1-score for overall performance (weighted average)
precision = report['weighted avg']['precision']
recall = report['weighted avg']['recall']
f1_score = report['weighted avg']['f1-score']
# Plot accuracy and classification report metrics
report_labels = ['Accuracy', 'Precision', 'Recall', 'F1-score']
values = [accuracy, precision, recall, f1_score]
colors = ['orange', 'blue', 'green', 'brown']
plt.figure(figsize=(9, 7))
bars = plt.bar(report_labels, values, color=colors)
plt.xlabel('Metric')
plt.ylabel('Score')
plt.title('Classification Report for Bagging Classifier: Accuracy, Precision, Recall, and F1-score')
```

Appendix F: Code for Naïve Bayes Model

Note: Evaluation part of the code is same as other models.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.naive_bayes import MultinomialNB
from sklearn.utils import resample
# Loading the dataset
df = pd.read_csv("/content/Accident_Dataset.csv")
# Categorical features to encode using one hot encoding
features = ["Sub_City", "Gender", "Road_Condition", "Age", "Date_Accident_Occur",
            "Road_Junction_Type", "Road_Pavement_Type", "Road_Layout", "Road_Division",
            "Light_Condition", "Type_Of_Accident", "Weather_Condition", "Pedstrain_Movment",
            "Vehiecle_Type", "Vehicle_Plate_Number","Vehicle_Movment", "Driver_Experiance", "Offense_Type"]
# Setting input features X and target y
X = df[features]
y = df['Level_of_Accident']
# Using pandas get_dummies method for one-hot encoding
encoded_df = pd.get_dummies(X, drop_first=True)
# Creating labelencoder object
lb = LabelEncoder()
y_encoded = lb.fit_transform(y)
print("Encoded labels:", lb.classes_)
y_en = pd.Series(y_encoded)
# Feature selection method using chi2 for categorical output, categorical input
fs = SelectKBest(chi2, k=50)
X_new = fs.fit_transform(encoded_df, y_en)
# Take the selected features
cols = fs.get_support(indices=True)
selected_features = encoded_df.columns[cols]
fs_df = pd.DataFrame(X_new, columns=selected_features)
# Train and test split
X_train, X_test, y_train, y_test = train_test_split(fs_df, y_en, test_size=0.2, random_state=42)
# Naive Bayes classifier
model = MultinomialNB()
# Fit the model
model.fit(X_train, y_train)
# Predict on test data
y_pred = model.predict(X_test)
```

Appendix G: Code for KNN Model

Note: Evaluation part of the code is same as other models.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.neighbors import KNeighborsClassifier
# Loading the dataset
df = pd.read_csv("/content/Accident_Dataset.csv")
# Categorical features to encode using one hot encoding
features = ["Sub_City", "Gender", "Road_Condition", "Age", "Date_Accident_Occur",
            "Road_Junction_Type", "Road_Pavement_Type", "Road_Layout", "Road_Division",
            "Light_Condition", "Type_Of_Accident", "Weather_Condition", "Pedstrain_Movment",
            "Vehiecle_Type", "Vehicle_Plate_Number",
            "Vehicle_Movment", "Driver_Experiance", "Offense_Type"]
# Setting input features X and target y
X = df[features]
y = df['Level_of_Accident']
# Using pandas get_dummies method for one-hot encoding
encoded_df = pd.get_dummies(X, drop_first=True)
# Creating labelencoder object
lb = LabelEncoder()
y_encoded = lb.fit_transform(y)
print("Encoded labels:", lb.classes_)
y_en = pd.Series(y_encoded)
# Feature selection method using chi2 for categorical output, categorical input
fs = SelectKBest(chi2, k=50)
X_new = fs.fit_transform(encoded_df, y_en)
# Take the selected features
cols = fs.get_support(indices=True)
selected_features = encoded_df.columns[cols]
fs_df = pd.DataFrame(X_new, columns=selected_features)
# Train and test split
X_train, X_test, y_train, y_test = train_test_split(fs_df, y_en, test_size=0.2, random_state=42)
# Standardize features for KNN (important for distance-based algorithms)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Build K-Nearest Neighbors (KNN) model
knn_model = KNeighborsClassifier(n_neighbors=20)  # Specify the number of neighbors
knn_model.fit(X_train, y_train)
```

Appendix H: Code for Logistic Regression Model

Note: Evaluation part of the code is same as other models.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt, import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.linear_model import LogisticRegression
# Loading the dataset
df = pd.read_csv("/content/Accident_Dataset.csv")
# Categorical features to encode using one hot encoding
features = ["Sub_City", "Gender", "Road_Condition", "Age", "Date_Accident_Occur",
            "Road_Junction_Type", "Road_Pavement_Type", "Road_Layout", "Road_Division",
            "Light_Condition", "Type_Of_Accident", "Weather_Condition", "Pedstrain_Movment",
            "Vehiecle_Type", "Vehicle_Plate_Number","Vehicle_Movment", "Driver_Experiance", "Offense_Type"]
# Setting input features X and target y
X = df[features]
y = df['Level_of_Accident']
# Using pandas get_dummies method for one-hot encoding
encoded_df = pd.get_dummies(X, drop_first=True)
# Creating labelencoder object
lb = LabelEncoder()
y_encoded = lb.fit_transform(y)
print("Encoded labels:", lb.classes_)
y_en = pd.Series(y_encoded)
# Feature selection method using chi2 for categorical output, categorical input
fs = SelectKBest(chi2, k=50)
X_new = fs.fit_transform(encoded_df, y_en)
# Take the selected features
cols = fs.get_support(indices=True)
selected_features = encoded_df.columns[cols]
fs_df = pd.DataFrame(X_new, columns=selected_features)
# Train and test split
X_train, X_test, y_train, y_test = train_test_split(fs_df, y_en, test_size=0.2, random_state=42)
# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Build Logistic Regression model
model = LogisticRegression(max_iter=1000, random_state=42)
# Train the model
model.fit(X_train, y_train)
# Predict on test data
y_pred = model.predict(X_test)
```

Appendix I: Code for Decision Trees Model

Note: Evaluation part of the code is same as other models.

```
import pandas as pd, import numpy as np
import matplotlib.pyplot as plt, import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score
from sklearn.feature_selection import SelectKBest, chi2,from sklearn.tree import DecisionTreeClassifier
# Loading the dataset
df = pd.read_csv("/content/Accident_Dataset.csv")
# Categorical features to encode using one hot encoding
features = ["Sub_City", "Gender", "Road_Condition", "Age", "Date_Accident_Occur",
            "Road_Junction_Type", "Road_Pavement_Type", "Road_Layout", "Road_Division",
            "Light_Condition", "Type_Of_Accident", "Weather_Condition", "Pedstrain_Movment",
            "Vehiecle_Type", "Vehicle_Plate_Number",
            "Vehicle_Movment", "Driver_Experiance", "Offense_Type"]
# Setting input features X and target y
X = df[features]
y = df['Level_of_Accident']
# Using pandas get_dummies method for one-hot encoding
encoded_df = pd.get_dummies(X, drop_first=True)
# Creating labelencoder object
lb = LabelEncoder()
y_encoded = lb.fit_transform(y)
print("Encoded labels:", lb.classes_)
y_en = pd.Series(y_encoded)
# Feature selection method using chi2 for categorical output, categorical input
fs = SelectKBest(chi2, k=50)
X_new = fs.fit_transform(encoded_df, y_en)
# Take the selected features
cols = fs.get_support(indices=True)
selected_features = encoded_df.columns[cols]
fs_df = pd.DataFrame(X_new, columns=selected_features)
# Train and test split
X_train, X_test, y_train, y_test = train_test_split(fs_df, y_en, test_size=0.2, random_state=42)
# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Build Decision Tree model
model = DecisionTreeClassifier(random_state=42)
# Train the model
model.fit(X_train, y_train)
# Predict on test data
y_pred = model.predict(X_test)
```

Appendix J: Code for RNN Model

Note: Evaluation part of the code is same as other models.

```
import pandas as pd, import numpy as np
import matplotlib.pyplot as plt, import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.utils import resample, from sklearn.feature_selection import SelectKBest, chi2
from tensorflow.keras.models import Sequential, from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
# Loading the dataset
df = pd.read_csv("/content/Accident_Dataset.csv")
# Categorical features to encode using one hot encoding
features = ["Sub_City", "Gender", "Road_Condition", "Age", "Date_Accident_Occur",
            "Road_Junction_Type", "Road_Pavement_Type", "Road_Layout", "Road_Division",
            "Light_Condition", "Type_Of_Accident", "Weather_Condition", "Pedstrain_Movment",
            "Vehiecle_Type", "Vehicle_Plate_Number",
            "Vehicle_Movment", "Driver_Experiance", "Offense_Type"]
# Setting input features X and target y
X = df[features]
y = df['Level_of_Accident']
# Using pandas get_dummies method for one-hot encoding
encoded_df = pd.get_dummies(X, drop_first=True)
# Creating labelencoder object
lb = LabelEncoder()
y_encoded = lb.fit_transform(y)
print("Encoded labels:", lb.classes_)
y_en = pd.Series(y_encoded)
# Feature selection method using chi2 for categorical output, categorical input
fs = SelectKBest(chi2, k=50)
X_new = fs.fit_transform(encoded_df, y_en)
# Take the selected features
cols = fs.get_support(indices=True)
selected_features = encoded_df.columns[cols]
fs_df = pd.DataFrame(X_new, columns=selected_features)
# Train and test split
X_train, X_test, y_train, y_test = train_test_split(fs_df, y_en, test_size=0.2, random_state=42)
# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Build Reccurent Neural Network model
model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(3, activation='softmax')  # 3 output classes (0, 1, 2) for Level_of_Accident
])
# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
# Early stopping to prevent overfitting
early_stop = EarlyStopping(monitor='val_loss', patience=50, restore_best_weights=True)
# Train the model
history = model.fit(X_train, y_train, epochs=25, batch_size=32,
                    validation_data=(X_test, y_test), callbacks=[early_stop])
```