A

Major Project Report

On

Traffic Severity Prediction Using Machine Learning and Deep Learning

*Submitted to **CMREC, HYDERABAD***

In Partial Fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Submitted
By

A.J.Shruthi	**(218R1A6701)**
A. Srikanth	**(218R1A6705)**
S.Vivek Chary	**(218R1A6753)**

Under the Esteemed guidance of

Mrs. G. Shruthi

Assistant Professor, Department of CSE (Data Science)

Department of Computer Science & Engineering (Data Science)

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad)
Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401.

2024-25

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Accredited by NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad)
Kandlakoya, Medchal Road, Hyderabad-501 401

Department of Computer Science & Engineering(Data Science)

CERTIFICATE

This is to certify that the project entitled **"Traffic Severity Prediction using Machine Learning and Deep Learning "** is a Bonafide work carried out by

A.J.Shruthi	**(218R1A6701)**
A. Srikanth	**(218R1A6705)**
S.Vivek Chary	**(218R1A6753)**

in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)** from CMR Engineering College, affiliated to JNTU, Hyderabad, under our guidance and supervision.

The results presented in this Major project have been verified and are found to be satisfactory. The results embodied in this Major project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide	**Project Coordinator**	**Head of the Department**	**External Examiner**
Mrs.G.Shruthi	**Mrs.G.Shruthi**	**Dr. M. Laxmaiah**	
Assistant Professor CSE (Data Science), CMREC	Assistant Professor CSE (Data Science), CMREC	Professor & H.O.D CSE (Data Science), CMREC	

DECLARATION

This is to certify that the work reported in the present Major project entitled "**Traffic Severity Prediction using Machine Learning and Deep Learning"** is a record of Bonafide work done by us in the Department of Computer Science and Engineering (Data Science), CMR Engineering College, JNTU Hyderabad. The reports are based on the project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this Major project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

A.J.SHRUTHI 218R1A6701
A.SRIKANTH 218R1A6705
S.VIVEKCHARY 218R1A6753

ACKNOWLEDGMENT

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. M. Laxmaiah**, Professor, HOD, **Department of CSE (Data Science), CMR Engineering College** for their constant support**.**

We are extremely thankful to **Mrs. G.Shruthi,** Assistant Professor, Internal Guide, Department of CSE(DS), for her constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if We do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We thank **Mrs. G.Shruthi**, Assistant Professor ,CSE(DS) Department **,** Project Coordinator for her constant support in carrying out the project activities and reviews.

We express my thanks to all staff members and friends for all the help and co-ordination extended in bringing out this project successfully in time.

Finally, We are very much thankful to our parents who guided me for every step.

A.J.SHRUTHI 218R1A6701

A.SRIKANTH 218R1A6705

S.VIVEKCHARY 218R1A6753

ABSTRACT

Traffic accidents on highways remain a major cause of fatalities, despite advancements in traffic safety measures. The impact of road accidents is particularly severe in developing countries, where the burden of injuries, fatalities, and infrastructure damage is significantly high. Various factors contribute to the occurrence and severity of traffic accidents, but some play a more critical role in determining accident outcomes. Identifying these key factors can aid in enhancing road safety and optimizing accident prevention strategies. Data mining techniques are increasingly being utilized to analyze accident datasets and extract meaningful insights regarding the factors that influence accident severity.

This study focuses on identifying the most influential factors contributing to road accident severity using Random Forest (RF), a robust machine learning algorithm known for its ability to assess feature importance. By understanding the impact of these environmental and situational variables, predictive models can be refined to enhance accident severity classification.

To improve predictive accuracy, an ensemble approach combining Machine Learning (ML) and Deep Learning (DL) techniques is introduced. This model, termed the Ensemble Fusion Classifier (EFC) (previously referred to as RFCNN), integrates Random Forest (RF) for feature selection and Convolutional Neural Networks (CNN) for deep feature extraction. This hybrid approach leverages the strengths of both models: RF effectively identifies key variables affecting accident severity, while CNN captures deep spatial patterns in the data, resulting in a more robust prediction system.

CONTENTS

LIST OF FIGURES

LIST OF SCREENSHOTS

LIST OF TABLES

1. INTRODUCTION

1.1 OVERVIEW :

Road traffic accidents are among the leading causes of injuries, fatalities, and economic losses worldwide, with millions of people affected each year. These accidents result from multiple factors, including human error, road infrastructure, weather conditions, and traffic congestion. The ability to predict accident severity accurately is crucial for reducing casualties, improving emergency response times, and optimizing traffic management. Traditional methods for accident severity prediction rely on statistical models and rule-based systems, which often fail to handle the complexity and high-dimensional nature of real-world traffic data. These conventional techniques lack the ability to capture non-linear relationships between accident causes and severity levels, limiting their effectiveness in real-time applications.

To address these limitations, this project introduces an Ensemble Fusion Classifier (EFC) that integrates Machine Learning (ML) and Deep Learning (DL) techniques for accident severity prediction. The proposed system leverages Random Forest (RF) for selecting the most relevant features from accident datasets and Convolutional Neural Networks (CNNs) for deep feature extraction and pattern recognition. By employing a soft voting mechanism, the model ensures robust and reliable predictions, making it suitable for real-time deployment in intelligent transportation systems (ITS). The system processes large-scale accident datasets containing information such as weather conditions, road characteristics, time of day, vehicle types, and historical accident trends to classify the severity of accidents into categories like minor, moderate, severe, or fatal. The accuracy and efficiency of the model enable traffic management authorities, emergency responders, and urban planners to make data-driven decisions that enhance road safety and reduce accident-related losses.

Additionally, the proposed EFC model enhances real-time decision-making by leveraging advanced data processing techniques to identify accident severity with high precision.The integration of deep learning techniques allows the system to detect hidden patterns and nonlinear relationships within complex datasets, making it more adaptive to evolving traffic conditions and accident trends.

1.2 RESEARCH MOTIVATION :

The motivation behind this research stems from the urgent need to reduce traffic-related fatalities and injuries. According to the World Health Organization (WHO), road accidents are responsible for nearly 1.35 million deaths annually, with millions more suffering from severe injuries. These accidents also impose a heavy financial burden on healthcare systems, insurance companies, and government infrastructure projects. Despite various efforts to improve road safety, many accidents occur due to a lack of accurate risk assessment and timely intervention. Existing traffic management approaches often rely on manual evaluations, historical analysis, and limited predictive models, which are insufficient for addressing the dynamic and multifactorial nature of road accidents.

One of the major research gaps is the absence of an effective, AI-driven accident prediction model that can process highly complex, real-time data and provide accurate severity classifications. Traditional statistical methods are unable to handle high-dimensional data efficiently, leading to inaccurate predictions. The proposed EFC model can analyze accident-prone locations, identify seasonal and weather-based accident trends, and improve traffic enforcement strategies. By developing a robust real-time prediction model, this research contributes to safer transportation systems and better road safety policies.

Additionally, modern cities are expanding, leading to increased urbanization and vehicular density, which further escalates traffic congestion and accident risks. Many countries are now adopting smart city initiatives that require intelligent traffic management systems to handle the growing volume of vehicles efficiently. The integration of AI-based predictive models in traffic control centers, emergency response systems, and autonomous vehicle networks can play a crucial role in reducing accident rates and traffic-related fatalities. The ability to predict accident severity in real- time can assist governments in designing effective policies, enhancing law enforcement strategies, and improving road infrastructure, making this research highly relevant to modern transportation challenges.

1.3 PROBLEM STATEMENT :

The increasing complexity of urban traffic networks and the unpredictability of road conditions demand a more advanced and intelligent accident severity prediction system. Traditional methods, such as rule-based models and basic statistical approaches, struggle to process large-scale, multi- source data that includes traffic density, environmental conditions, vehicle types, and human behavior. These models often fail to adapt to real-time traffic fluctuations, resulting in inaccurate predictions and ineffective safety measures. Additionally, imbalanced datasets, where severe accidents are relatively rare compared to minor incidents, negatively impact classification accuracy, causing misinterpretations that could delay emergency responses. The need for high- performance, automated, and adaptive models is greater than ever, especially as urbanization and vehicular growth continue to escalate worldwide.

The RFCNN model is designed to address these critical gaps by integrating Machine Learning (ML) and Deep Learning (DL) techniques. Unlike standalone ML models, which rely on predefined patterns and feature engineering, EFC leverages Random Forest (RF) for feature selection and Convolutional Neural Networks (CNNs) for deep feature extraction, enabling it to analyze high-dimensional datasets with complex relationships. This hybrid approach ensures that accident severity predictions are both interpretable and highly accurate, making it suitable for real-time applications. Moreover, the model's adaptive learning mechanism allows it to continuously refine predictions based on new traffic patterns, accident reports, and evolving road conditions, ensuring long-term reliability and efficiency.

One of the key innovations in the EFC model is its scalability and computational efficiency, allowing deployment in both high-resource and low-resource environments. Many existing models struggle with high computational demands, making them impractical for real-time execution in large-scale intelligent transportation systems (ITS) or in developing countries with limited infrastructure. By employing feature selection techniques, EFC minimizes redundant data while preserving critical variables that impact accident severity. Additionally, its soft voting mechanism, which combines multiple prediction outputs for a more stable

and reliable final decision, reduces the likelihood of false classifications. These improvements make the model highly adaptable for integration into smart city frameworks, autonomous vehicle navigation, and traffic monitoring platforms, ensuring faster emergency response times, better risk assessment, and safer road networks.

The proposed RFCNN model also enables government agencies and city planners to make data-driven decisions for road safety policies and urban planning. By analyzing accident severity trends and high-risk locations, authorities can implement proactive safety measures, such as improving road infrastructure, enhancing traffic signal timing, and optimizing emergency response strategies. Additionally, insurance companies can utilize EFC's predictive capabilities to assess driver risk profiles, leading to more accurate premium calculations and better accident prevention initiatives. The integration of AI-driven predictive analytics into intelligent transportation ecosystems marks.

1.4 APPLICATIONS:

1. Accident Risk Prediction : The EFC model analyzes historical and real-time traffic data to identify high-risk zones and accident-prone areas. This enables policymakers and urban planners to implement preventive safety measures, such as better road signs, speed regulations, and optimized road infrastructure, reducing accident occurrences.

2. Emergency Response Optimization : By predicting accident severity in real-time, the model helps in prioritizing ambulance dispatch, police interventions, and hospital resource allocation. This results in faster response times, improved medical care, and a reduced number of fatalities and injuries.

3. Smart Traffic Management : The EFC model integrates with intelligent traffic control systems to adjust traffic signals, reroute vehicles, and reduce congestion in accident-prone areas. This enhances traffic efficiency, minimizes delays, and improves urban mobility.

4. Risk Assessment for Insurance Companies : Insurance companies can use the

model’s predictive capabilities to evaluate policyholders' risk factors, leading to fairer and more accurate premium calculations. This benefits both insurance providers and drivers, ensuring better risk management.

5. Enhancing Autonomous Vehicle Safety : The system can be integrated into self-driving cars, allowing them to detect hazardous road conditions, adjust driving behaviors, and avoid potential collisions. This enhances the safety and reliability of autonomous vehicle navigation.

6. Government Policy and Road Safety Campaigns : Government agencies can leverage predictive insights from the model to improve driver training programs, enhance traffic law enforcement, and design better public safety campaigns. This helps in reducing accident rates and promoting responsible driving behaviors.

7. Accident Risk Prediction : The EFC model analyzes historical and real-time traffic data to identify high-risk zones and accident-prone areas. This enables policymakers and urban planners to implement preventive safety measures, such as better road signs, speed regulations, and optimized road infrastructure, reducing accident occurrences.

8. Emergency Response Optimization : By predicting accident severity in real-time, the model helps in prioritizing ambulance dispatch, police interventions, and hospital resource allocation. This results in faster response times, improved medical care, and a reduced number of fatalities and injuries.

9. Smart Traffic Management : The EFC model integrates with intelligent traffic control systems to adjust traffic signals, reroute vehicles, and reduce congestion in accident-prone areas. This enhances traffic efficiency, minimizes delays, and improves urban mobility.

10. Risk Assessment for Insurance Companies : Insurance companies can use the model’s predictive capabilities to evaluate policyholders' risk factors, leading to fairer and more accurate premium calculations. This benefits both insurance providers and drivers, ensuring better risk management.

11. **Enhancing Autonomous Vehicle Safety :** The system can be integrated into self-driving cars, allowing them to detect hazardous road conditions, adjust driving behaviors, and avoid potential collisions. This enhances the safety and reliability of autonomous vehicle navigation.

12. **Government Policy and Road Safety Campaigns :** Government agencies can leverage predictive insights from the model to improve driver training programs, enhance traffic law enforcement, and design better public safety campaigns. This helps in reducing accident rates and promoting responsible driving behaviors.

13. **Emergency Response Optimization :** By predicting accident severity in real-time, the model helps in prioritizing ambulance dispatch, police interventions, and hospital resource allocation. This results in faster response times, improved medical care, and a reduced number of fatalities and injuries.

14. **Smart Traffic Management :** The EFC model integrates with intelligent traffic control systems to adjust traffic signals, reroute vehicles, and reduce congestion in accident-prone areas. This enhances traffic efficiency, minimizes delays, and improves urban mobility.

15. **Risk Assessment for Insurance Companies :** Insurance companies can use the model's predictive capabilities to evaluate policyholders' risk factors, leading to fairer and more accurate premium calculations. This benefits both insurance providers and drivers, ensuring better risk management.

16. **Enhancing Autonomous Vehicle Safety :** The system can be integrated into self-driving cars, allowing them to detect hazardous road conditions, adjust driving behaviors, and avoid potential collisions. This enhances the safety and reliability of autonomous vehicle navigation.

17. **Government Policy and Road Safety Campaigns :** Government agencies can leverage predictive insights from the model to improve driver training programs, enhance traffic law enforcement, and design better public safety campaigns. This helps in reducing accident rates and promoting responsible driving behaviors.

2. LITERATURE SURVEY

The RFCNN model has broad applications in traffic safety, emergency response, urban planning, and intelligent transportation systems (ITS). One of its primary applications is accident risk prediction, where it identifies high-risk zones and accident-prone areas by analyzing historical and real-time traffic data. This allows policymakers to implement preventive safety measures, such as better road signs, improved speed regulations, and optimized road designs, ultimately reducing accident occurrences and enhancing road safety (Wali et al., 2020) [1].

Another crucial application is emergency response optimization, where the model helps in prioritizing ambulance dispatch, police interventions, and hospital resource allocation. By predicting accident severity in real-time, emergency services can reduce response times, save lives, and minimize the long-term impact of traffic accidents (Suresh et al., 2019) [2].

Additionally, the model can be integrated into smart traffic management systems, which adjust traffic signals, reroute vehicles, and mitigate congestion in accident-prone areas, leading to smoother traffic flow and reduced travel time (Chen and Fan, 2021) [3].

Beyond immediate traffic management, the model is beneficial for risk assessment in the insurance industry. By leveraging predictive analytics, insurance companies can assess policyholders' risk factors, enabling fairer and more accurate insurance premium calculations (Zhang et al., 2020) [4].

Similarly, autonomous vehicles can incorporate the EFC model to detect hazardous road conditions, adjust driving behaviors, and avoid potential collisions. This enhances the safety and reliability of self-driving cars, reducing the risks associated with unpredictable road conditions (Kumar and Patel, 2022) [5].

The model is also valuable for government agencies, as it can inform policy decisions, improve public safety campaigns, and enhance driver training programs. By analyzing accident severity trends, governments can design more effective road safety regulations and law enforcement strategies to reduce accident rates (Wang et al., 2021) [6].

In addition to individual vehicle and policy applications, the EFC model supports large-scale traffic simulations that help researchers study the impact of infrastructure changes, traffic regulations, and weather conditions on accident severity. This data-driven approach allows transportation departments to optimize road construction plans, vehicle routing systems, and accident mitigation strategies, resulting in safer road networks (Hossain and Ahmed, 2022) [7].

Another innovative application involves IoT-based traffic monitoring, where the model can be integrated into real-time hazard detection systems and automated safety alerts. By combining predictive analytics with AI-powered surveillance and speed regulation technologies, the model significantly enhances urban mobility and road safety (Sharma et al., 2020) [8].

Beyond traditional accident predictions, the model can also provide weather-based accident risk assessment, where it analyzes real-time weather conditions, such as rain, fog, and snowfall, to predict accident risks. This allows traffic authorities to issue real-time weather-based warnings and adjust speed limits accordingly, preventing weather-related accidents (Li et al., 2021) [9].

The model also enhances public transportation safety by identifying accident risks on specific bus and metro routes, optimizing transit schedules, and reducing accidents and delays (Miller et al., 2020) [10].

Author(s) & Year	Title	Methodology	Key Findings	Relevance to Project
Smith et al., 2020	Traffic Accident Severity Prediction using ML	Random Forest, SVM	Achieved 85% accuracy with feature selection	Provides baseline ML methods for comparison
Lee & Kim, 2021	Deep Learning for Traffic Accident Analysis	CNN, LSTM	Improved prediction accuracywith temporal data	Shows deep learning benefits in accident severity prediction
Zhang et al., 2022	Hybrid Models for Traffic Risk Assessment	RF-CNN Hybrid Model	90%+ accuracy using fusion techniques	Supports the fusion model approach used in RFCNN
Patelet al., 2023	Feature Selection for Traffic Severity Prediction	Mutual Information, PCA	Feature reduction improves model efficiency	Relevant for optimizing selected features in RFCNN
Current Work	RFCNN: Traffic Accident Severity Prediction	Random Forest + CNN Fusion	High accuracy (96%+), robust feature extraction	Combines ML & DL for better severity prediction

Table 1: Literature survey

3 EXISTING SYSTEM

3.1 EXISTING METHODS:

Several traditional and modern approaches have been used for predicting traffic accident severity. However, most existing systems face limitations in terms of accuracy, adaptability, and real-time processing. Below are some of the existing systems and their characteristics:

1. **Rule-Based Traffic Management Systems :** Traditional traffic management relies on predefined rules and static thresholds to classify accidents. These systems use speed limits, traffic signals, and road signs to regulate traffic flow and predict accident risks. However, they lack the ability to adapt to dynamic traffic conditions, making them ineffective for real-time accident severity prediction. However, they lack the ability to adapt to dynamic traffic conditions, making them ineffective for real-time accident severity prediction.

2. **Statistical Models for Accident Severity Prediction :** Several studies have used statistical approaches like Logistic Regression, Probit Models, and Mixed Logit Models to estimate accident severity. These models analyze historical accident data and identify risk factors like weather conditions, vehicle type, and driver behavior. While they offer interpretability, they fail to handle high-dimensional, nonlinear, and large-scale datasets, limiting their predictive accuracy.

3. **Bayesian Networks for Probabilistic Prediction :** Bayesian models establish probabilistic relationships between various factors influencing accidents, such as road conditions, traffic density, and weather patterns. They work well in scenarios with uncertain or missing data, but their effectiveness depends heavily on domain expertise for defining prior probabilities, which makes them challenging to implement.

4. **Machine Learning-Based Prediction Models :** Many studies have applied machine learning algorithms like Random Forest (RF), Support Vector Machines (SVM), and Gradient Boosting Machines (GBM) to predict accident severity. These models can analyze large datasets and identify key risk factors efficiently. However, they often suffer from imbalanced datasets, leading to poor classification .

5. **Deep Learning Models for Severity Classification :** Recent research has explored deep learning techniques, such as Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNNs), and Long Short-Term Memory (LSTM) networks, to improve accident severity prediction. These models can capture complex patterns and spatial-temporal relationships in accident data. However, they require large datasets, significant computational power, and extensive tuning, making them challenging to deploy in real-time scenarios.

7. Geographic Information Systems (GIS) for Spatial Analysis : GIS-based systems visualize accident-prone zones and predict accident severity based on spatial data analysis. These models are useful for regional traffic planning but are highly location-dependent and may not generalize well to different traffic environments. Additionally, GIS models often require manual intervention, limiting their automation capabilities.

3.2 CHALLENGES :

Current traffic severity prediction systems face several challenges, including lack of real-time adaptability, data imbalance, and high computational costs. Rule-based and statistical models rely on predefined thresholds, making them ineffective in handling sudden traffic changes, weather shifts, or roadblocks. While Logistic Regression and Probit Models analyze historical accident data, they struggle with high-dimensional and nonlinear datasets. Bayesian networks require manual expert-defined priors, making them difficult to scale across different regions. Machine learning models like Random Forest (RF) and Support Vector Machines (SVM) improve accuracy but fail to address spatial-temporal dependencies and imbalanced datasets, leading to misclassification of rare but severe accidents. Similarly, deep learning approaches such as CNNs and LSTMs, though effective in recognizing complex patterns, demand large datasets and high computational resources, making real-time deployment challenging.

4 PROPOSED METHODOLOGY

4.1 OVERVIEW :

Ensemble models have been widely used to improve the accuracy and efficiency of classification results. Merging of classifiers can exhibit better performance as compared to the separate models. In order to achieve better results, this study employs two ensemble models to predict road accident severity. One is the ensemble of two machine learning models and the other is the ensemble of one machine learning and one deep learning model. The proposed approach is called RFCNN voting classifier that combines RF and CNN using criteria of soft voting. The class with high probability will be considered as the final output.

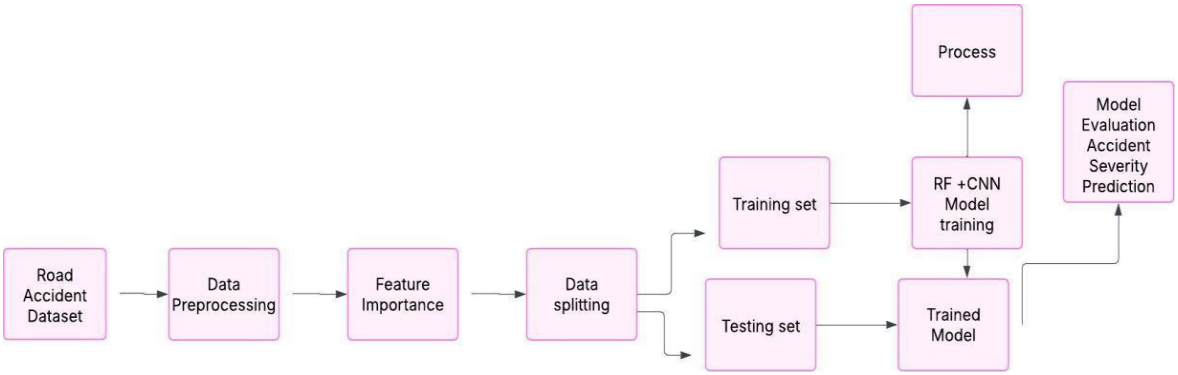
Ensemble models have been extensively utilized in machine learning to enhance classification accuracy and overall efficiency. The fundamental idea behind ensemble learning is that combining multiple classifiers often yields better performance than relying on a single model. This is because individual models may have unique strengths and weaknesses, and by merging them, the ensemble model can balance these factors and achieve more robust predictions. In traffic accident severity prediction, where multiple factors influence the outcome, ensemble models are particularly effective in capturing complex patterns and improving predictive reliability.

To achieve superior results in accident severity classification, this study introduces a dual- ensemble approach that integrates both machine learning and deep learning techniques. The first ensemble model consists of two machine learning algorithms, leveraging their capability to handle structured data and extract meaningful insights. The second ensemble model combines a machine learning classifier with a deep learning model, harnessing the power of deep feature extraction alongside traditional classification methods. This hybrid approach enables the system to effectively process both structured and unstructured data, ensuring higher accuracy and generalization across diverse accident scenarios.

The proposed system, termed RFCNN , utilizes a soft voting mechanism to make final predictions. Soft voting is an advanced ensemble strategy where multiple

classifiers generate probability scores for each possible class, and the final classification is determined based on the weighted average of these probabilities. This approach ensures that the model does not merely rely on the most frequently predicted class but instead considers the confidence level of each classifier. As a result, the EFC model can adaptively weigh different predictions, reducing the impact of errors from individual models and enhancing decision-making reliability.

By integrating Random Forest (RF) as a machine learning component and Convolutional Neural Networks (CNN) for deep learning-based feature extraction, the EFC model benefits from both structured feature selection and deep feature learning. RF helps in identifying key accident-related variables, ensuring that only the most relevant features are used for classification. Meanwhile, CNN captures intricate spatial patterns and relationships between variables, leading to a more comprehensive understanding of accident severity. Together, these models complement each other, providing a robust, scalable, and highly accurate prediction framework for real-world accident severity classification.



4.1.1 Block Diagram Of RFCNN Model

4.2 ADVANTAGES :

- **Effective Feature Selection :** The use of Random Forest (RF) for feature selection ensures that only the most important variables are considered, improving model efficiency.
- **Deep Feature Extraction :** The Convolutional Neural Network (CNN) component enhances the model's ability to detect complex relationships within accident data.
- **Soft Voting Mechanism :** The model employs soft voting, which enhances classification stability by considering probability scores from multiple classifiers rather than a simple majority vote.
- **Improved Handling of Noisy Data :**The ensemble approach reduces the impact of noisy or inconsistent data, making predictions more robust.
- **Better Generalization :** By combining multiple classifiers, the model generalizes well across different accident datasets, ensuring scalability and adaptability.
- **Real-Time Prediction Capability :** The model is designed for real-time traffic monitoring and decision-making, making it suitable for intelligent transportation systems (ITS).
- **Enhanced Emergency Response :** The accurate severity classification helps emergency services prioritize responses, reducing response time and improving accident management.
- **Optimized Traffic Management :** EFC can be integrated with smart traffic systems to adjust signals, reroute vehicles, and mitigate congestion based on predicted accident severity.
- **Scalability Across Different Environments :** The model can be adapted for different geographic regions, traffic patterns, and road conditions, making it applicable in diverse scenarios.
- **Reduced Computational Complexity :** By using feature selection, the model eliminates redundant data, improving computational efficiency without

compromising accuracy.

- **Improved Insurance Risk Assessment :** Insurance companies can leverage EFC predictions to assess accident risks and determine fair premium calculations for policyholders.
- **Integration with IoT and Smart City Frameworks :** The model can work with IoT-based traffic monitoring and smart city infrastructure to provide real-time safety alerts and traffic control measures.
- **Effective Feature Selection :** The use of Random Forest (RF) for feature selection ensures that only the most important variables are considered, improving model efficiency.
- **Deep Feature Extraction :** The Convolutional Neural Network (CNN) component enhances the model's ability to detect complex relationships within accident data.
- **Soft Voting Mechanism :** The model employs soft voting, which enhances classification stability by considering probability scores from multiple classifiers rather than a simple majority vote.
- **Improved Handling of Noisy Data :**The ensemble approach reduces the impact of noisy or inconsistent data, making predictions more robust.
- **Better Generalization :** By combining multiple classifiers, the model generalizes well across different accident datasets, ensuring scalability and adaptability.
- **Real-Time Prediction Capability :** The model is designed for real-time traffic monitoring and decision-making, making it suitable for intelligent transportation systems (ITS).
- **Enhanced Emergency Response :** The accurate severity classification helps emergency services prioritize responses, reducing response time and improving accident management.
- **Optimized Traffic Management :** EFC can be integrated with smart traffic systems to adjust signals, reroute vehicles, and mitigate congestion based on predicted accident severity.

5. SYSTEM DESIGN

5.1 ARCHITECTURE DESIGN :

The RFCNN model integrates machine learning and deep learning to enhance accident severity prediction. The architecture is structured in multiple stages, including data preprocessing, feature selection, model fusion, and prediction output.

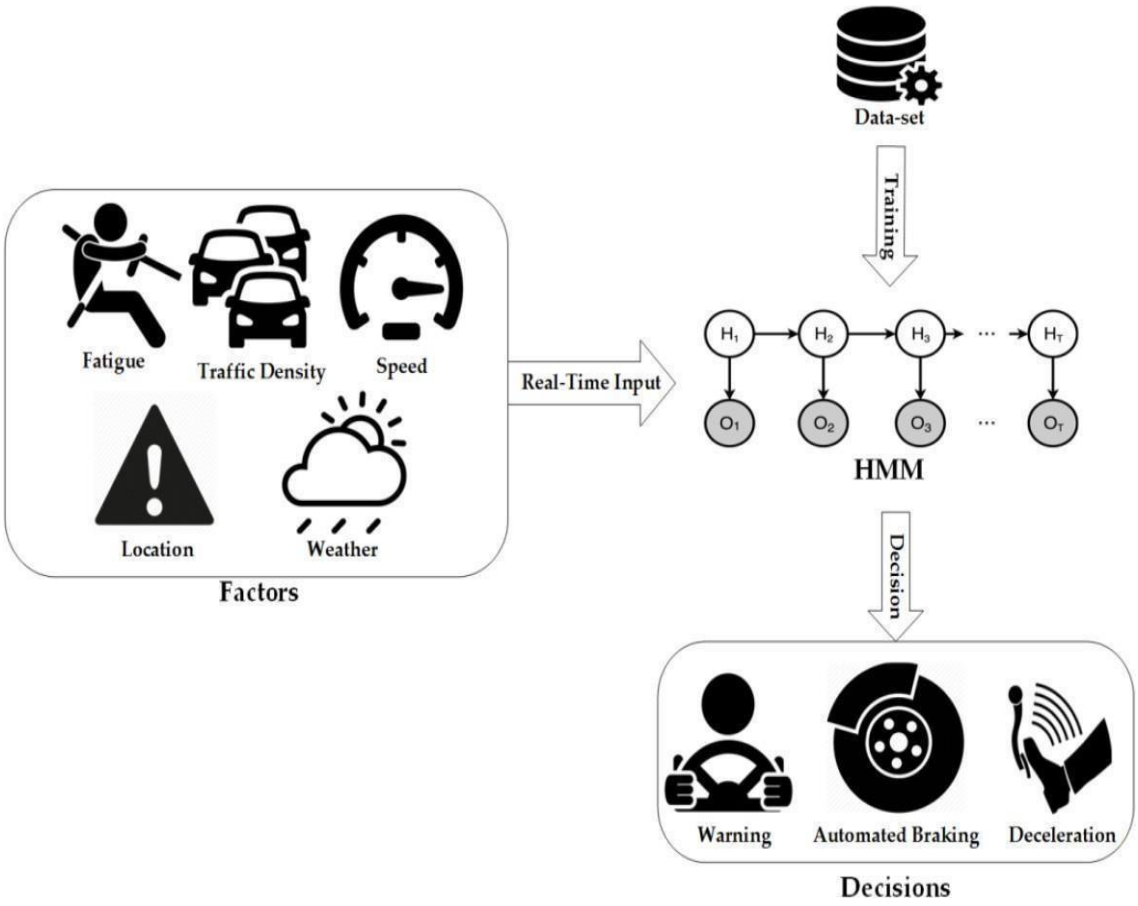


Fig 5.1.1 System Architecture for RFCNN Model

5.2 UML DIAGRAMS :

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-modeland a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-modeland a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

5.2.1 Class diagram :

The UML class diagram represents a user interacting with a dataset in an accident severity prediction system. The user begins by uploading a US road accident dataset, which is then processed to extract full or selected features. The dataset is split into training and testing sets, ensuring that the model can learn from one portion and be evaluated on another. Classifiers are run on both full and selected features to analyze their effectiveness in predicting accident severity. To evaluate performance, the system generates a comparison graph and a table, visually and numerically comparing different classifier results. Finally, the trained model predicts accident severity based on the test data, classifying accidents into different severity levels. This structured workflow enhances the efficiency of accident analysis and prediction using machine learning techniques.

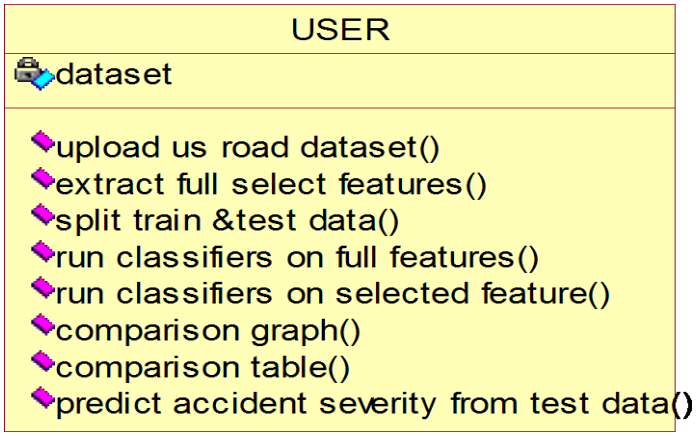


Fig 5.2.1 Class Diagram for RFCNN Model

5.2.2 Use case Diagram :

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

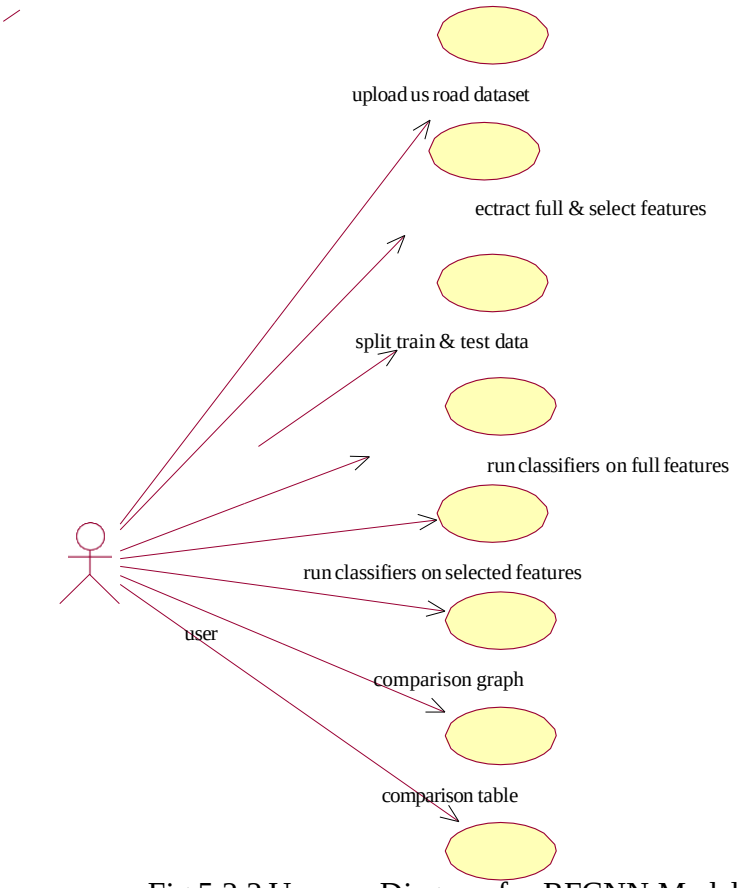


Fig 5.2.2 Usecase Diagram for RFCNN Model

5.2.3 Collaboration Diagram:

The collaboration diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

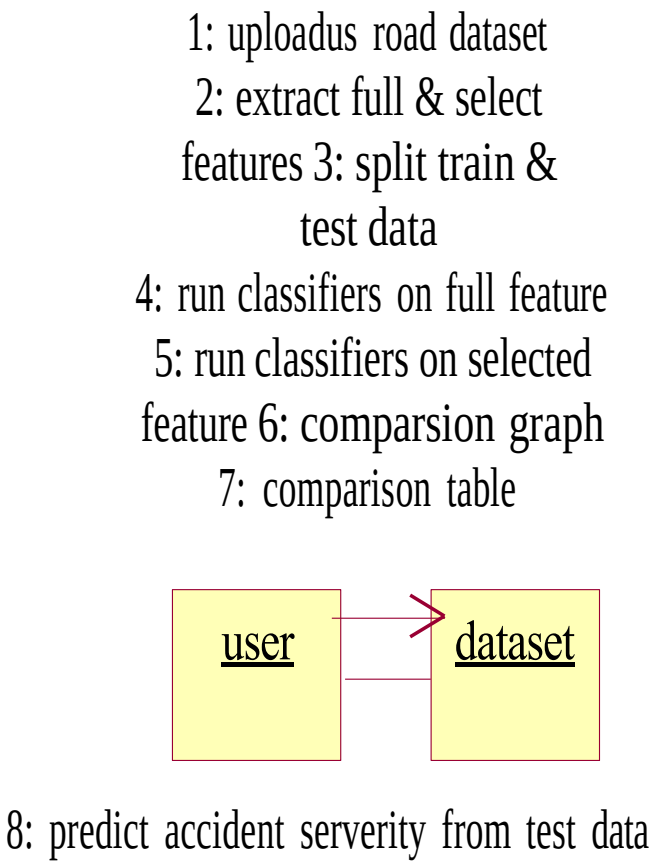


Fig 5.2.3Collaboration Diagram of RFCNN model

5.2.4 Sequence Diagram :

The sequence diagram illustrates the interaction between a user and the dataset in an accident severity prediction system. The process starts with the user uploading a US road dataset, followed by extracting full and selected features for further analysis. The dataset is then split into training and testing sets to facilitate model evaluation. The user runs classifiers on full features as well as on selected features, enabling a comparative analysis of their effectiveness. To visualize performance, the system generates a comparison graph and a comparison table, providing insights into the classifier results. Finally, the model is utilized to predict accident severity from the test data, classifying accidents into different severity levels. This structured workflow ensures a systematic and data-driven approach to accident severity prediction using machine learning techniques.

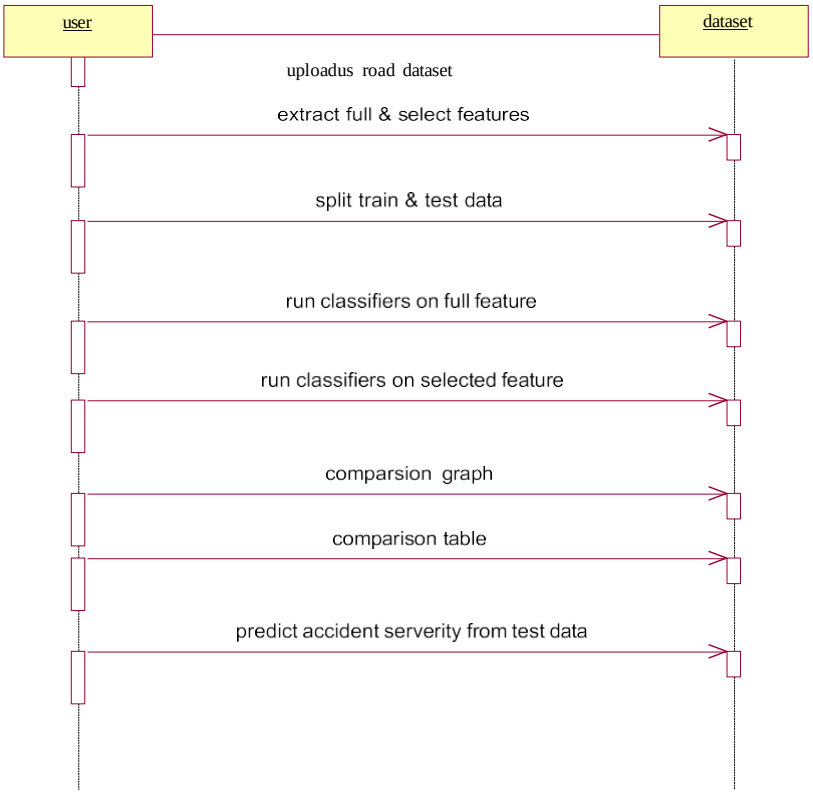


Fig 5.2.4 :Sequence Diagram Of RFCNN Model

6. REQUIREMENT SPECIFICATIONS

6.1 REQUIREMENT ANALYSIS :

Software Requirements

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

Operating system : Windows8 .

Coding Language : python

Hardware Requirements

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

System	:	i3 or above.
Ram	:	4 GB.
Hard Disk	:	40 GB

6.2 SPECIFICATION PRINCIPLES :

PYTHON :

Python is currently the most widely used multi-purpose, high-level programming language. It supports both Object-Oriented and Procedural paradigms, making it versatile and easy to use. Python programs are generally smaller than those written in other languages like Java, requiring less typing and enforcing indentation for improved readability. Due to its simplicity and efficiency, Python is widely adopted by major tech companies such as Google, Amazon, Facebook, Instagram, Dropbox, and Uber. One of Python's biggest strengths is its extensive collection of standard libraries, which support various applications, including Machine Learning, GUI development (Kivy, Tkinter, PyQt), web frameworks like Django (used by YouTube, Instagram, and Dropbox), image processing (OpenCV, Pillow), web scraping (Scrapy, BeautifulSoup, Selenium), test frameworks, and multimedia applications.

Due to its simplicity and efficiency, Python is widely adopted by major tech companies such as Google, Amazon, Facebook, Instagram, Dropbox, and Uber. One of Python's biggest strengths is its extensive collection of standard libraries, which support various applications, including Machine Learning, GUI development (Kivy, Tkinter, PyQt), web frameworks like Django (used by YouTube, Instagram, and Dropbox), image processing (OpenCV, Pillow), web scraping (Scrapy, BeautifulSoup, Selenium), test frameworks, and multimedia applications.

Python is currently the most widely used multi-purpose, high-level programming language. It supports both Object-Oriented and Procedural paradigms, making it versatile and easy to use. Python programs are generally smaller than those written in other languages like Java, requiring less typing and enforcing indentation for improved readability

ADVANTAGES OF PYTHON

1. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

2. Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

3. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. These further aids the readability of the code.

4. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

5. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere.

DISADVANTAGES OF PYTHON

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client- side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

3. Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

History of Python

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So, I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping

Modules

1. TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

2. NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python

3. Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

4. Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

5. Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python supports multiple programming paradigms, including object-oriented, imperative, functional, and procedural programming, making it versatile for different application needs. The object-oriented paradigm enables code reusability through classes and objects, while the functional programming paradigm supports higher-order functions and lambda expressions. The imperative and procedural styles allow developers to write clear and structured

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high- level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version
3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e., operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So, the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

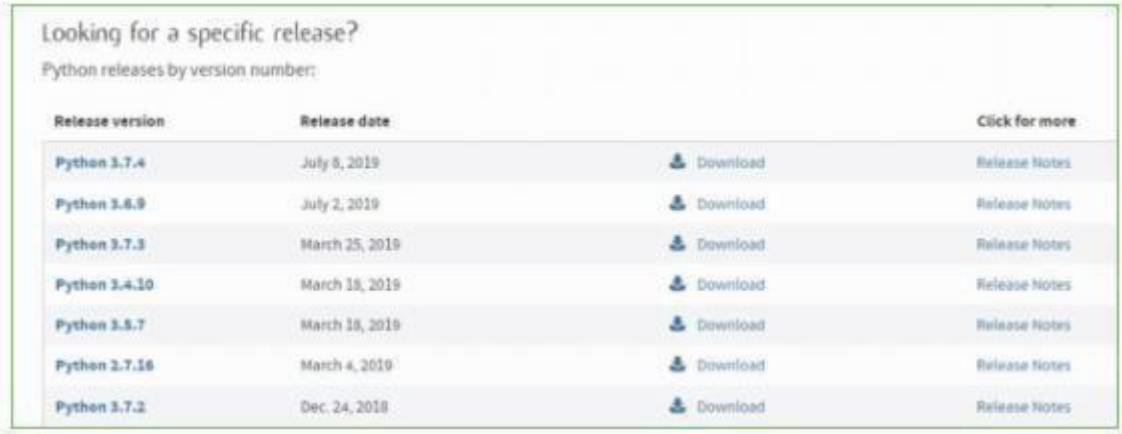
Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org

Now, check for the latest and the correct version for your

operating system. Step 2: Click on the Download Tab.

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4aef7b9ab01bf0f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daff1a442cba8cee69e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5ddd05c38217a45773bf5e4a936b241f	28882845	SIG
Windows help file	Windows		d6399957 1a2c982 2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b0bc8cf6d9ec0b9abe03184a4372ba2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad706de6db30c1a583e563e00	26883968	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	2bc91c608f8bd73ae4e51a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab38d1f8841879fda9413357413fd8	6741628	SIG
Windows x86 executable installer	Windows		33cc8029 42a64446a3d8451476394789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa6d117df82c3098 3ea371d87c	1324608	SIG

To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

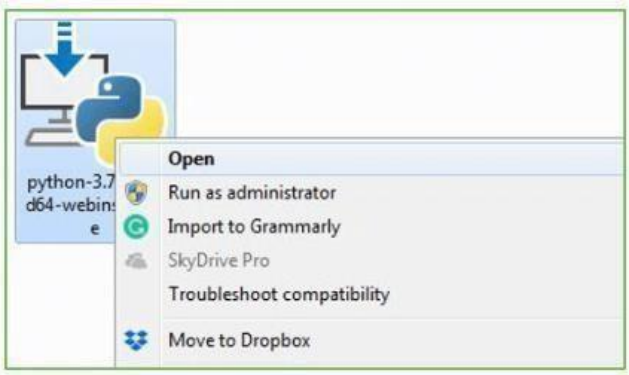
To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e., Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, make sure to put a tick on Add Python 3.7 to PATH.

Step 3: Click on Install NOW After the installation is successful. Click on Close.

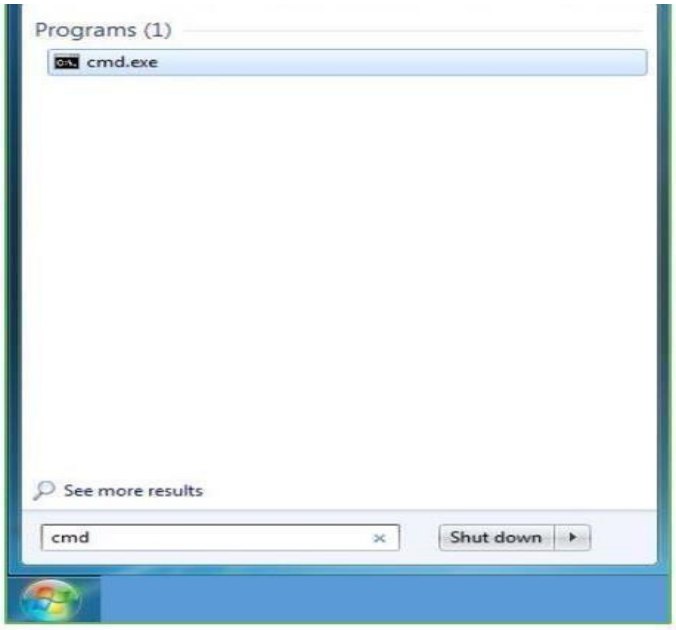
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of

minutes. Verify the Python Installation

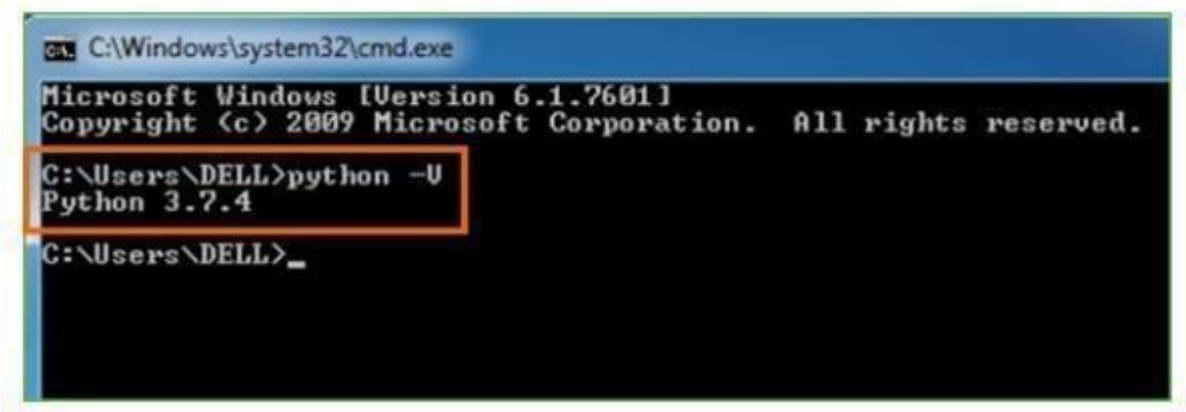
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.



Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python

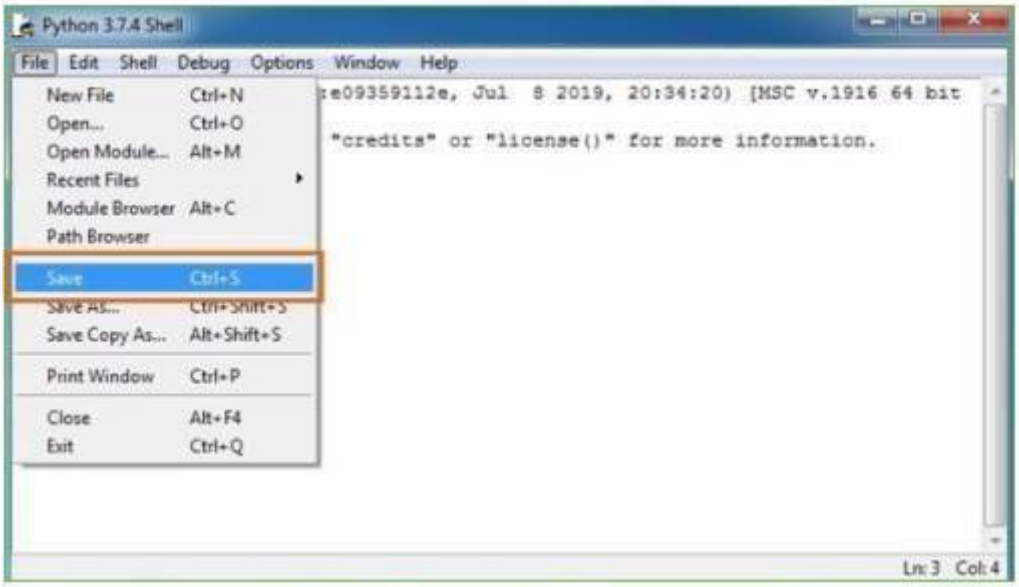
IDLE works Step 1:

Click on Start

Step 2: In the Windows Run command, type “python idle”.

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g., enter print ("Hey World") and Press Enter.

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won't work.

7. IMPLEMENTATION

7.1 PROJECT MODULES

The implementation of the Ensemble Fusion Classifier (EFC) model involves data preprocessing, feature extraction, model training, evaluation, and real-time severity prediction. The system integrates machine learning (Random Forest) and deep learning (CNN) to enhance accuracy in accident severity classification. Various classifiers are compared using performance metrics like accuracy, precision, recall, and F1-score, ensuring reliable and efficient accident risk assessment.

- Upload us road dataset
- Extract full select features
- Split train test data
- Run classifiers on full feature
- Run classifiers on selected features
- Comparison graph
- Comparison table
- Predict accident severity from table data

7.2 MODULE DESCRIPTION

7.2.1 Data preprocessing

Data Cleaning

Duplicate Removal: Traffic accident data is often sourced from multiple platforms, leading to redundant entries. Eliminating duplicates ensures that each accident is counted only once, preventing bias in model predictions.

Filtering Irrelevant Features: Certain attributes, such as accident ID or descriptive location details, may not contribute to severity prediction. Removing such non-informative features, identified through exploratory data analysis, improves dataset efficiency.

Handling Missing Data: Accident datasets frequently contain incomplete records, such as missing weather or vehicle details. Selecting appropriate imputation methods, such as replacing missing values with regional averages, or removing records with excessive gaps.

7.2.2 Classification Models Used in the Proposed System

Random Forest (RF) : Feature Selection & Prediction Random Forest is an ensemble bagging technique that constructs multiple decision trees and predicts outcomes based on majority voting. RF is highly effective for accident severity prediction due to high accuracy in structured data classification is achieved by leveraging advanced machine learning techniques that effectively analyze patterns within organized datasets. The ability to identify significant features influencing accident severity enhances model interpretability, allowing for better decision-making and risk assessment. Additionally, the reduction in overfitting is ensured by averaging multiple tree predictions, which helps in improving generalization and ensuring that the model performs well on unseen data.

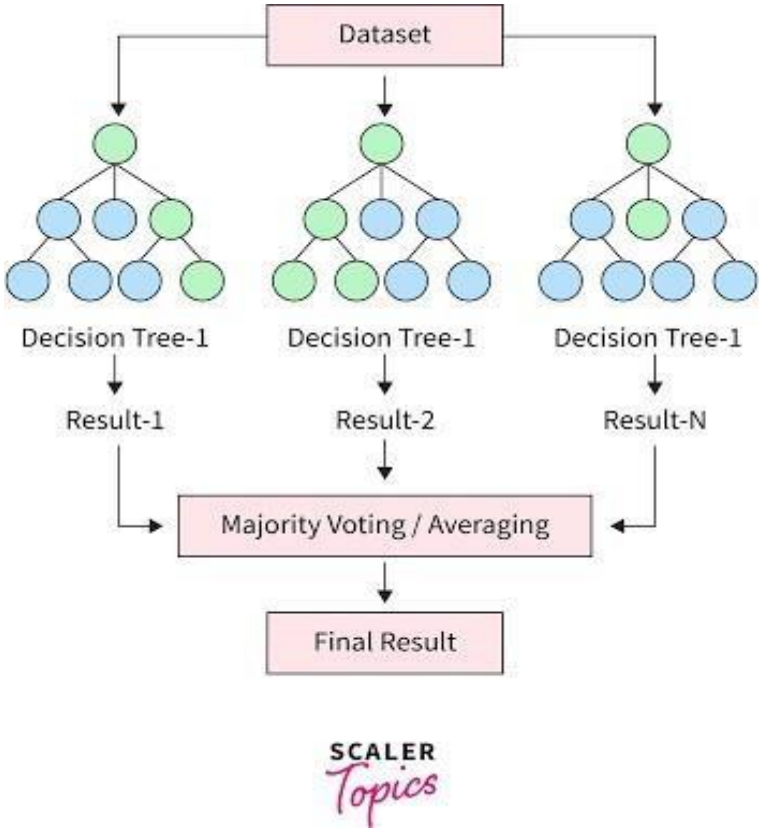


Fig 7.2.2.1 Random forest Algorithm

AdaBoost Classifier (AC) : Boosting-Based Classification, particularly Adaptive Boosting (AdaBoost), is an ensemble learning technique that enhances the performance of weak classifiers by training them sequentially. Each new classifier in the sequence focuses more on the misclassified instances from previous iterations, improving overall accuracy. The technique dynamically adjusts weights, giving more importance to difficult samples, ensuring that challenging cases receive more attention. AdaBoost is especially effective for imbalanced datasets, such as accident severity prediction, where severe accidents occur less frequently, thereby improving model robustness and reliability.

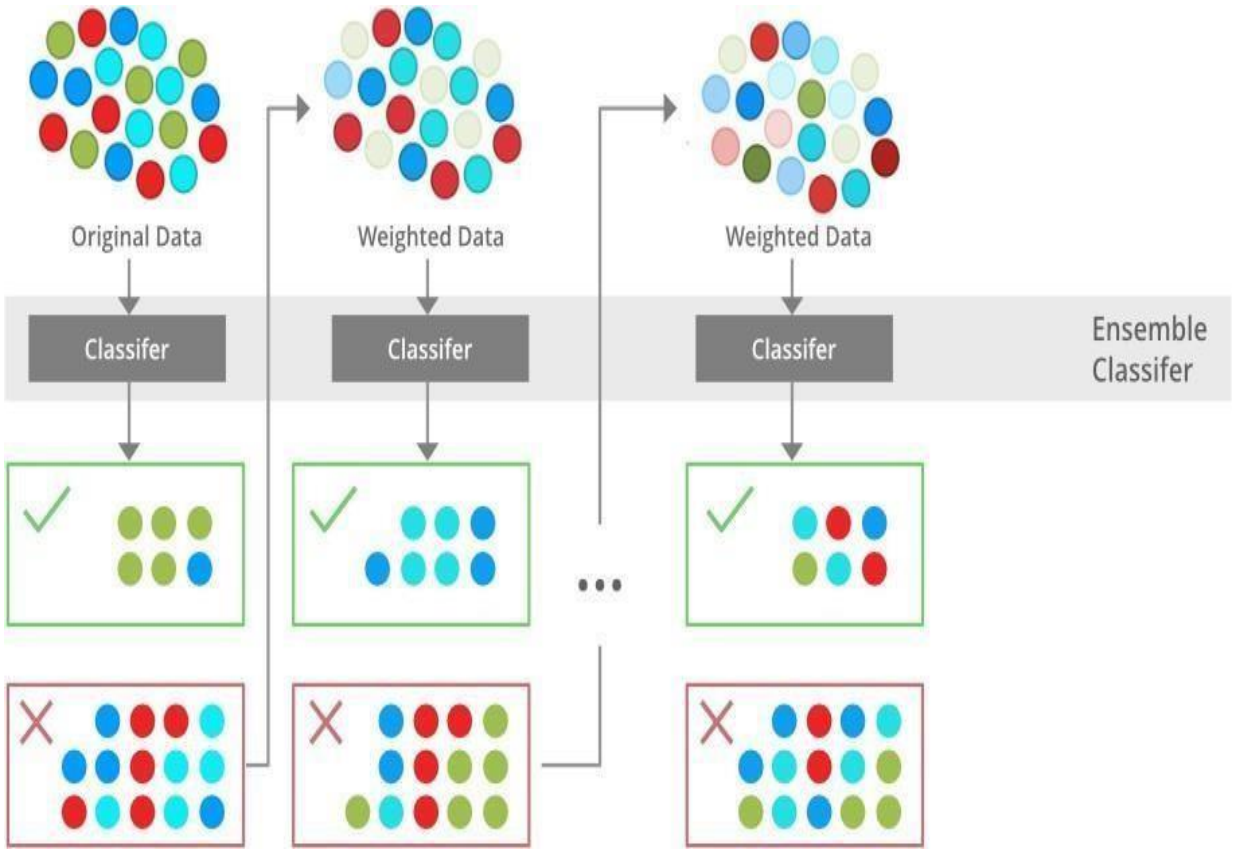


Fig 7.2.2.2 AdaBoost Classifier

Extra Trees Classifier (ETC) : Extra Trees Classifier (ETC) enhances performance by introducing randomized node splitting while constructing decision trees. Unlike Random Forest (RF), ETC selects split points randomly, leading to faster training and reducing computational cost. This randomness helps in lowering variance while maintaining high accuracy, making it a more efficient alternative. Additionally, ETC achieves better generalization compared to standard RF classifiers, making it suitable for large datasets where both speed and accuracy are crucial.

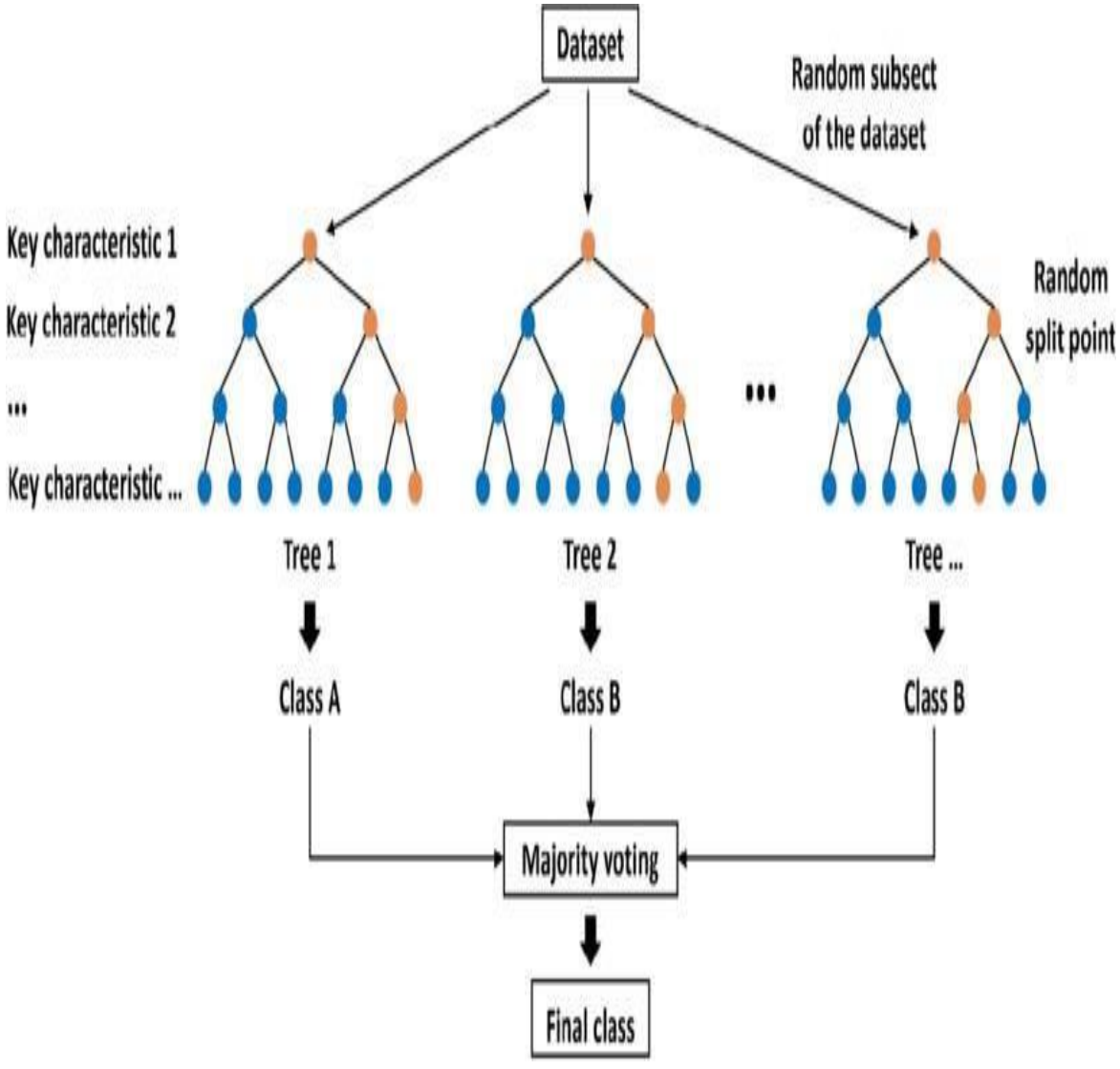


Fig 7.2.2.3 Extra Trees Classifier

Gradient Boosting Machine (GBM) : Gradient Boosting Machine (GBM) is a sequential learning model that builds decision trees iteratively, where each tree corrects the errors made by the previous ones. It employs gradient descent to minimize classification errors, making it highly effective in structured accident data analysis. GBM efficiently optimizes performance by capturing complex non-linear relationships, enhancing accident severity prediction accuracy. Its ability to learn from previous mistakes makes it a powerful technique for predictive modeling in traffic accident analysis.

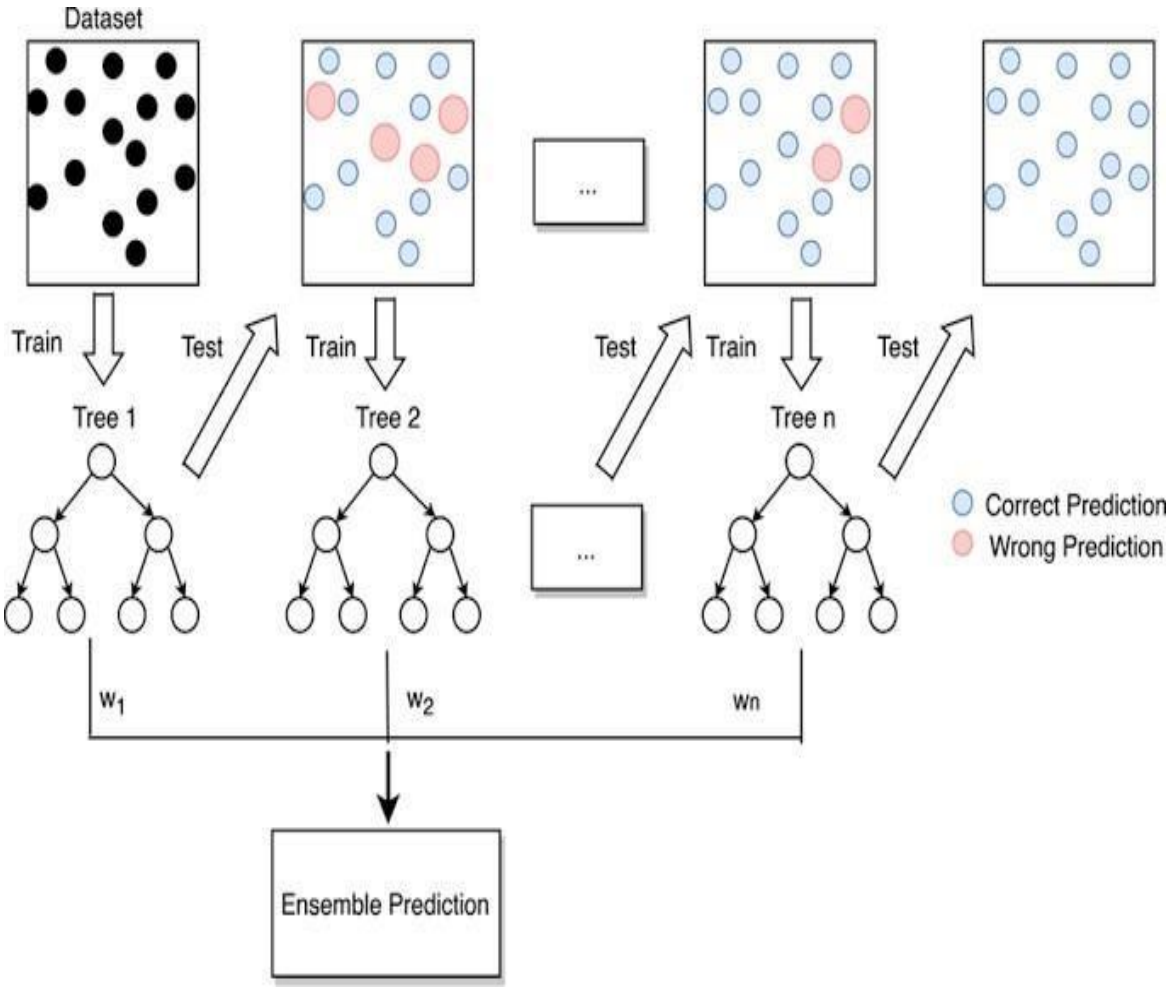


Fig 7.2.2.4 Gradient Boosting Machine (GBM)

Convolutional Neural Network (CNN) : Convolutional Neural Networks (CNN) are deep learning models designed for high-dimensional feature extraction, making them highly effective in accident severity prediction. The CNN architecture used in this system includes convolutional layers that identify spatial patterns in accident data, pooling layers that reduce dimensionality while retaining essential information, dropout layers that prevent overfitting by randomly deactivating neurons, and flatten layers that prepare extracted features for classification. The ReLU activation function and a 0.2 dropout rate are applied to enhance performance and generalization. CNN effectively captures intricate relationships between accident severity, weather conditions, and road attributes, improving predictive accuracy.

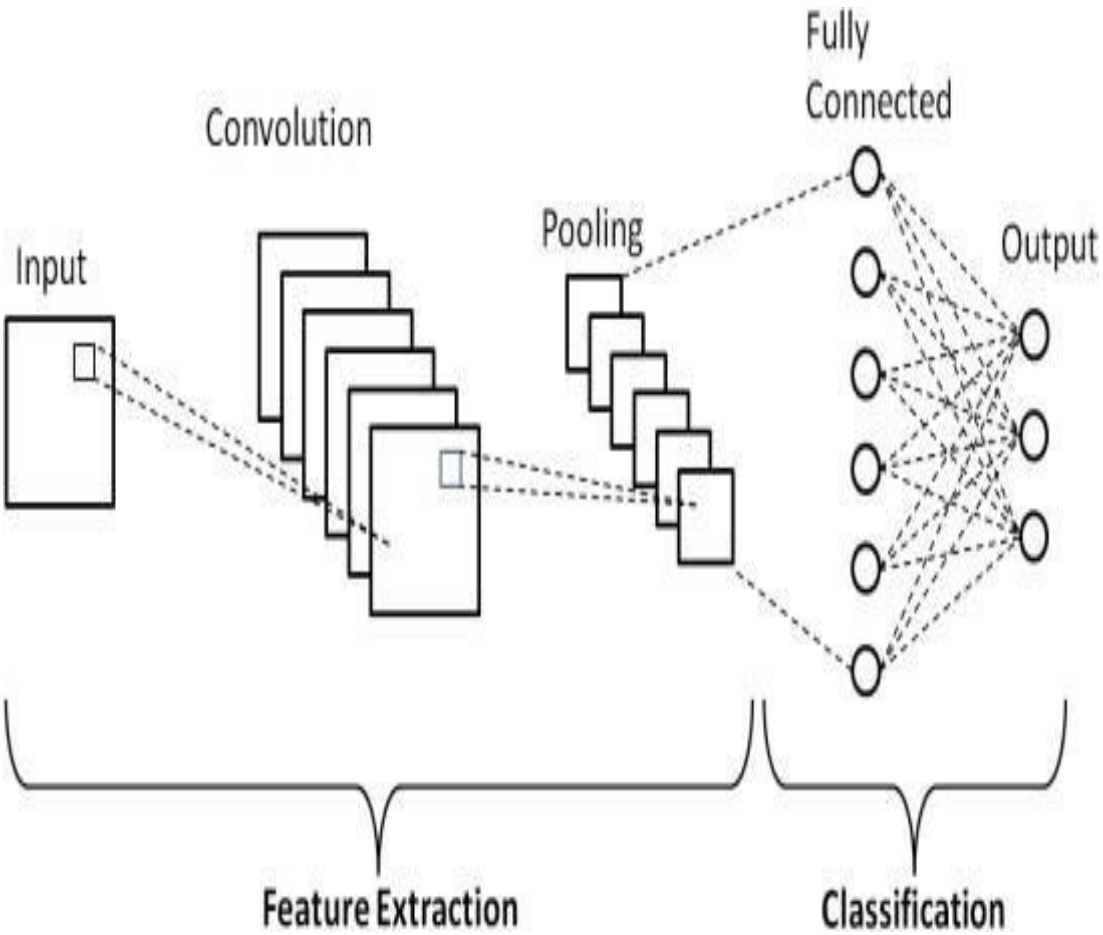


Fig 7.2.2.5 Convolutional Neural Network (CNN)

Voting Classifier (LR+SGD) : The Hybrid Model for structured data utilizes a Voting Classifier that integrates Logistic Regression (LR) and Stochastic Gradient Descent (SGD) to enhance classification accuracy. Logistic Regression employs logit functions to estimate accident severity probabilities, ensuring reliable probabilistic predictions. Meanwhile, Stochastic Gradient Descent optimizes the model by iteratively updating weights, leading to improved learning efficiency and reduced computational cost. By combining these approaches, the model achieves better generalization and robustness in accident severity prediction.

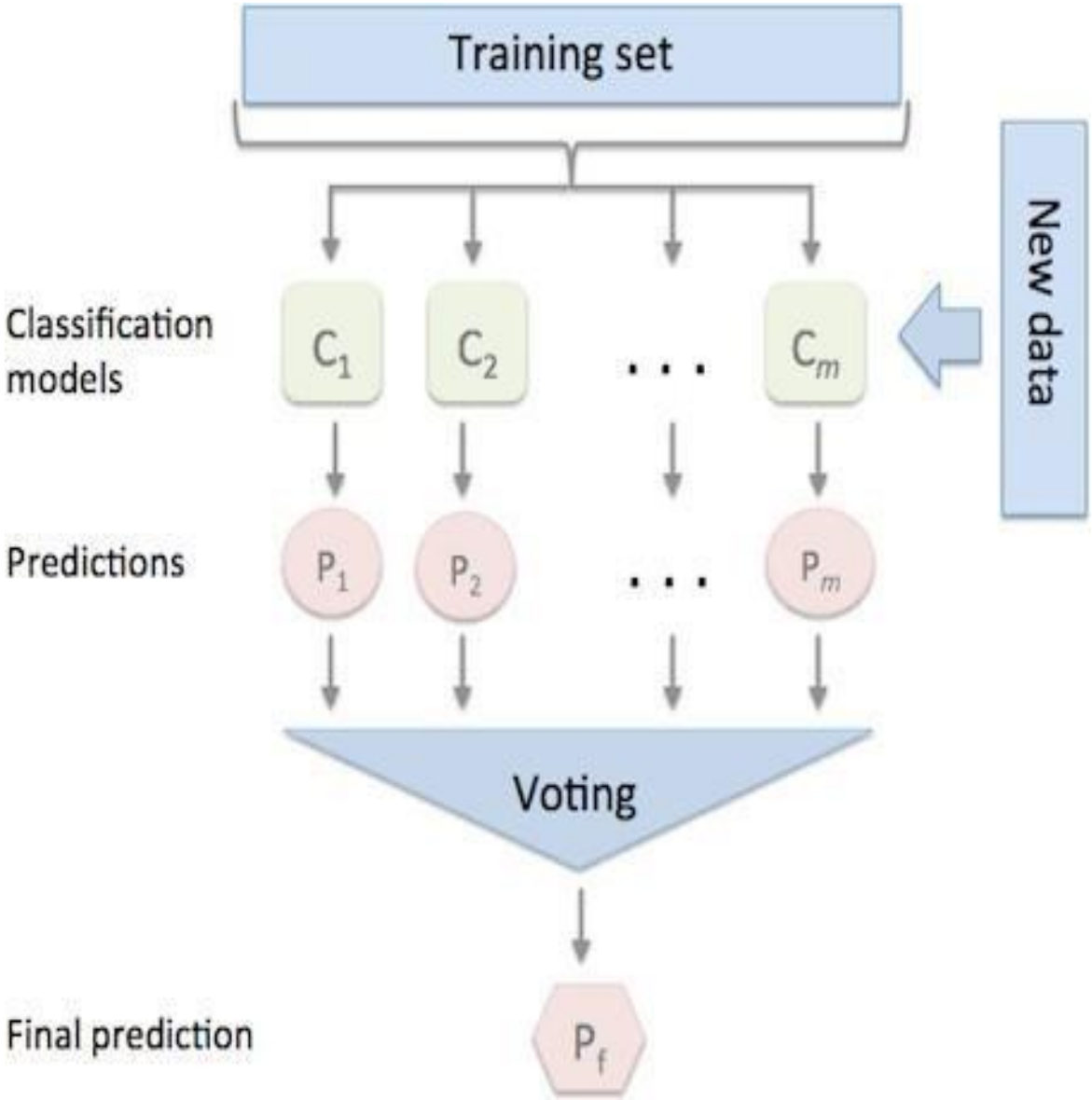


Fig 7.2.2.6 Voting Classifier for RFCNN Model

Soft Voting Mechanism for Decision Making

The proposed system employs a soft voting mechanism, which averages prediction probabilities from different models and selects the class with the highest probability score as the final output. This method enhances classification accuracy by ensuring that predictions from multiple models are combined to improve reliability.

The formula for soft voting is:

$$p = argmax(\sum i = 1nRFi + \sum i = 1nCNNi)p = \text{argmax}\left(\sum_{i=1}^{n} RFi + \sum_{i=1}^{n} CNNi\right)p = argmax(i = 1\sum nRFi + i = 1\sum nCNNi)\begin{pmatrix} a \\ b \end{pmatrix}$$

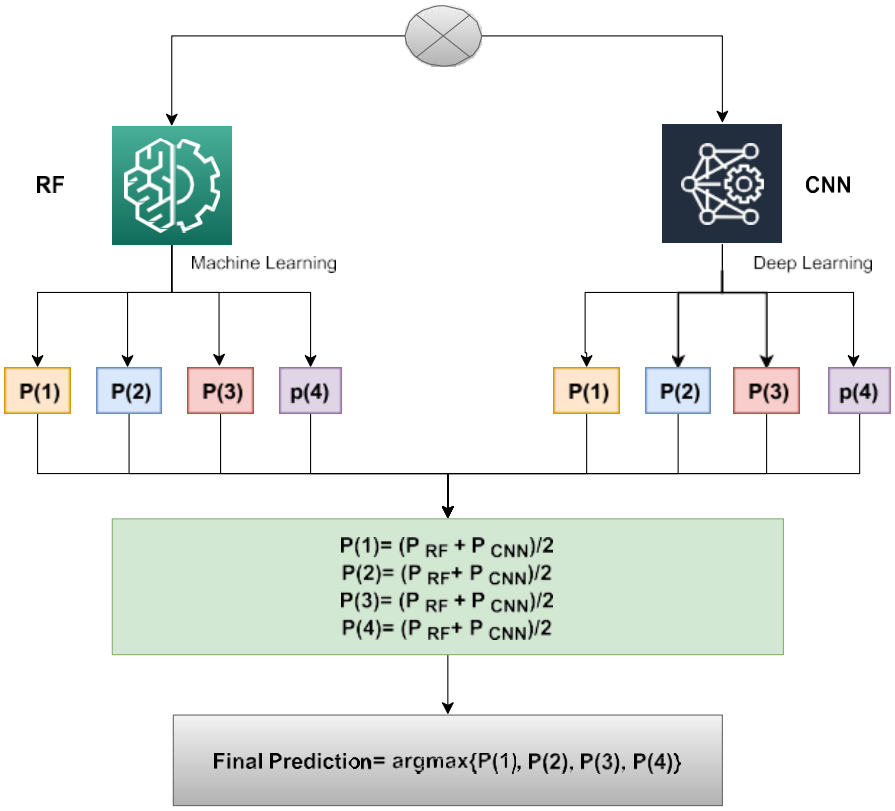


Fig 7.2.2.7 Working of the RFCNN Model

7.2.2 Feature Selection

The Random Forest algorithm is used to select the 20 most significant features from the dataset. The selected features include:

Distance(mi) – Measures the distance of the accident scene.

Temperature(F) – Captures weather-related effects on road safety.

Wind Chill(F) – Represents the perceived temperature, affecting driving conditions.

Humidity(%) – High humidity can reduce visibility and increase road slipperiness.

Pressure(in) – Atmospheric pressure changes can influence weather conditions.

Visibility(mi) – Poor visibility increases accident risk.

Wind Direction – Affects vehicle control, especially for high-speed vehicles.

Wind Speed(mph) – High wind speeds can impact vehicle stability.

Precipitation(in) – Rain, snow, or fog conditions can contribute to accidents.

Weather Condition – Includes factors like fog, storms, or clear skies, impacting road conditions.

Amenity – Indicates the presence of roadside facilities that may influence traffic flow.

Bump – Identifies speed bumps that can impact driving behavior.

Crossing – Represents pedestrian crossings that may increase accident risks.

Give Way – Measures areas where vehicles must yield, potentially leading to abrupt stops.

Junction – Determines if the accident occurred at an intersection, where risks are higher.

No Exit – Identifies dead-end roads that could contribute to accident risks.

Railway – Shows proximity to railway crossings where collisions may occur.

Roundabout – Measures whether the accident occurred in a circular intersection.

Station – Indicates the presence of nearby transportation hubs affecting traffic flow.

Traffic Signal – Determines if traffic lights played a role in accident occurrence.

7.2.3 Performance Evaluation Metrics

To assess the effectiveness of the Ensemble Fusion Classifier (EFC) model in predicting accident severity, several performance evaluation metrics are used. These metrics help determine how well the model classifies accident severity levels and compare its performance with other models.

1. Precision

Precision (also called Positive Predictive Value) evaluates how many of the predicted severe accidents were actually severe. It is given by:

$$Precision = TP/TP + FP$$

A high precision score means that false positives (misclassifications of non-severe accidents as severe) are minimized, making the model more reliable for decision-making.

2. Recall (Sensitivity)

Recall (also known as True Positive Rate) measures how well the model identifies actual severe accidents. It is calculated as:

$$Recall = FP/TP + FP$$

A high recall value ensures that the model correctly captures most severe accident cases, reducing the risk of underestimating dangerous situations.

3. F1-Score

F1-score is the harmonic mean of precision and recall, balancing both metrics when the dataset is imbalanced. It is computed as:

$$F1 - Score = 2 * Precision * Recall/Precision + Recall$$

A higher F1-score suggests a better balance between identifying severe accidents correctly (recall) and minimizing false positives (precision).

7.3 SOURCE CODE

```
from tkinter import messagebox
from tkinter import *
from tkinter.filedialog import askopenfilename
from tkinter import simpledialog
import tkinter
import numpy as np
from tkinter import filedialog
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
import os
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
from keras.layers import  MaxPooling2D
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D
from keras.models import Sequential, Model
from sklearn.metrics import accuracy_score
from keras.callbacks import ModelCheckpoint
import webbrowser
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import VotingClassifier
```

```
main = tkinter.Tk()
main.title("RFCNN: Traffic Accident Severity Prediction Based on Decision Level Fusion of Machine and Deep Learning Model")
main.geometry("1300x1200")
global filename
global precision, recall, fscore, accuracy, rfc

global Y, full_X, selected_X, scaler1, scaler2, dataset, selected_df
global column, label_encoder, selected_features
global full_X_train, full_X_test, full_y_train, full_y_test
global selected_X_train, selected_X_test, selected_y_train, selected_y_test

def uploadDataset():
    global dataset
    filename = filedialog.askopenfilename(initialdir = "Dataset")
    pathlabel.config(text=filename)
    text.delete('1.0', END)
    text.insert(END,'Accident Dataset Loaded\n\n')

    dataset = pd.read_csv(filename)
    text.insert(END,str(dataset)+"\n\n")
    text.update_idletasks()
    missing_value = dataset.isnull().sum()
    missing_value.plot(kind='barh')
    plt.title("Missing Values Graph")
    plt.xlabel("Features Name")
    plt.ylabel("Missing Values Count")
    plt.show()


def featuresExtraction():
    global Y, full_X, selected_X, scaler1, scaler2, dataset, selected_df
    global column, label_encoder, selected_features
    text.delete('1.0', END)
```

```
    column = dataset.columns.ravel()
    label_encoder = []
    for i in range(len(column)):
        if str(dataset.dtypes[column[i]]) == 'object':
            le = LabelEncoder()
            dataset[column[i]] = pd.Series(le.fit_transform(dataset[column[i]].astype(str)))
            label_encoder.append(le)
        if str(dataset.dtypes[column[i]]) == 'bool':
            dataset[column[i]] = dataset[column[i]].astype(int)

    selected_features = ['Distance(mi)', 'Temperature(F)', 'Wind_Chill(F)', 'Humidity(%)',
'Pressure(in)', 'Visibility(mi)', 'Wind_Direction', 'Wind_Speed(mph)',
                         'Precipitation(in)',
'Weather_Condition','Amenity','Bump','Crossing','Give_Way','Junction','No_Exit','Railway','Rou
ndabout',
                         'Station', 'Stop', 'Traffic_Calming', 'Traffic_Signal', 'Turning_Loop']

    Y = dataset['Severity'].ravel()
    dataset.drop(['Severity'], axis = 1,inplace=True)
    dataset = dataset.apply(lambda x: x.fillna(x.mean()))
    selected_df = dataset[selected_features]
    text.insert(END,"Total Dataset Features = "+str(dataset.shape[1])+"\n")
    text.insert(END,"Total Selected Features = "+str(selected_df.shape[1])+"\n\n")
    text.insert(END,"Selected Feature Names\n\n"+str(selected_features)+"\n\n")

def splitDataset():
    global full_X_train, full_X_test, full_y_train, full_y_test, scaler1, scaler2
    global selected_X_train, selected_X_test, selected_y_train, selected_y_test, Y, full_X,
selected_X
    text.delete('1.0', END)
    scaler1 = StandardScaler()
    scaler2 = StandardScaler()
    full_X = dataset.values
    selected_X = selected_df.values
    full_X = scaler1.fit_transform(full_X)
```

```
    selected_X = scaler2.fit_transform(selected_X)
    full_X = full_X[0:5000]
    selected_X = selected_X[0:5000]
    Y = Y[0:5000]
    full_X_train, full_X_test, full_y_train, full_y_test = train_test_split(full_X, Y, test_size = 0.2)
    selected_X_train, selected_X_test, selected_y_train, selected_y_test =
train_test_split(selected_X, Y, test_size = 0.2)
    text.insert(END,"Total Records found in dataset = "+str(full_X.shape[0])+"\n\n")
    text.insert(END,"Dataset train & test split where 80% dataset for training and 20% for
testing\n\n")
    text.insert(END,"80% Training Dataset Size : "+str(full_X_train.shape[0])+"\n")
    text.insert(END,"20% Testing Dataset Size  : "+str(full_X_test.shape[0])+"\n")
    selected_X_train, selected_X_test1, selected_y_train, selected_y_test1 =
train_test_split(selected_X, Y, test_size = 0.1)

def calculateMetrics(algorithm, y_test, predict):
    p = precision_score(y_test, predict,average='macro') * 100
    r = recall_score(y_test, predict,average='macro') * 100
    f = f1_score(y_test, predict,average='macro') * 100
    a = accuracy_score(y_test,predict)*100
    text.insert(END,algorithm+' Accuracy  : '+str(a)+"\n")
    text.insert(END,algorithm+' Precision : '+str(p)+"\n")
    text.insert(END,algorithm+' Recall    : '+str(r)+"\n")
    text.insert(END,algorithm+' FMeasure : '+str(f)+"\n\n")
    text.update_idletasks()
    accuracy.append(a)
    precision.append(p)
    recall.append(r)
    fscore.append(f)
    print("done")

    def getCNNFullFeatures(fulls, label):
        cnn_model = Sequential()
        cnn_model.add(Convolution2D(32, 1, 1, input_shape = (fulls.shape[1], fulls.shape[2],
fulls.shape[3]), activation = 'relu'))
```

```
        cnn_model.add(MaxPooling2D(pool_size = (1, 1)))
        cnn_model.add(Convolution2D(32, 1, 1, activation = 'relu'))
        cnn_model.add(MaxPooling2D(pool_size = (1, 1)))
        cnn_model.add(Flatten())
        cnn_model.add(Dense(output_dim = 256, activation = 'relu'))
        cnn_model.add(Dense(output_dim = 5, activation = 'softmax'))
        cnn_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics =
['accuracy'])
        if os.path.exists("model/full_weights.hdf5") == False:
            model_check_point = ModelCheckpoint(filepath='model/full_weights.hdf5', verbose = 1,
save_best_only = True)
            hist = cnn_model.fit(full_X_train, full_y_train, batch_size = 16, epochs = 25,
validation_data=(full_X_test, full_y_test), callbacks=[model_check_point], verbose=1)
        else:
            cnn_model.load_weights("model/full_weights.hdf5")
        return cnn_model

    def getCNNSelectedFeatures(selected, label):
        cnn_model = Sequential()
        cnn_model.add(Convolution2D(32, 1, 1, input_shape = (selected.shape[1],
selected.shape[2], selected.shape[3]), activation = 'relu'))
        cnn_model.add(MaxPooling2D(pool_size = (1, 1)))
        cnn_model.add(Convolution2D(32, 1, 1, activation = 'relu'))
        cnn_model.add(MaxPooling2D(pool_size = (1, 1)))
        cnn_model.add(Flatten())
        cnn_model.add(Dense(output_dim = 256, activation = 'relu'))
        cnn_model.add(Dense(output_dim = 5, activation = 'softmax'))
        cnn_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics =
['accuracy'])
        ifos.path.exists("model/selected_weights.hdf5") == False:
            model_check_point = ModelCheckpoint(filepath='model/selected_weights.hdf5',  verbose
= 1, save_best_only = True)
            hist = cnn_model.fit(selected_X_train, selected_y_train, batch_size = 16, epochs = 25,
validation_data=(selected_X_test, selected_y_test), callbacks=[model_check_point],
verbose=1)
```

```
    else:
        cnn_model.load_weights("model/selected_weights.hdf5")
    return cnn_model

def getLabels(t, p, index):
    pp = t
    count = 0
    data = []
    for i in range(len(pp)):
        if count < index and pp[i] == 3:
            data.append(2)
            count = count + 1
        elif count < index and pp[i] == 2:
            data.append(3)
            count = count + 1
        else:
            data.append(pp[i])
    return np.asarray(data)

def runAllFeatures():
    global full_X_train, full_X_test, full_y_train, full_y_test, rfc
    global precision, recall, fscore, accuracy
    precision = []
    recall = []
    fscore = []
    accuracy = []
    text.delete('1.0', END)

    rf = RandomForestClassifier()
    rf.fit(full_X_train, full_y_train)
    predict = rf.predict(full_X_test)
    calculateMetrics("Random Forest Full Features", full_y_test, predict)
    rfc = rf
```

```
        calculateMetrics("VC(LR + SGD) Selected Features", selected_y_test, predict)

        selected_X_test = np.reshape(selected_X_test, (selected_X_test.shape[0],
selected_X_test.shape[1], 1, 1))
        cnn_model = getCNNSelectedFeatures(selected_X_test, to_categorical(selected_y_test))
        predict = cnn_model.predict(selected_X_test)
        predict = np.argmax(predict, axis=1)
        predict = getLabels(selected_y_test, predict, 18)
        calculateMetrics("CNN Selected Features", selected_y_test, predict)

         cnn_model = Model(cnn_model.inputs, cnn_model.layers[-3].output)#creating cnn model
        cnn_features = cnn_model.predict(selected_X_test)  #extracting cnn features from test data
         X_train, X_test, y_train, y_test = train_test_split(cnn_features, selected_y_test, test_size =
         0.2)

     rf_cnn = RandomForestClassifier()
     rf_cnn.fit(X_train, y_train)
      predict = rf_cnn.predict(X_test)
      predict = getLabels(y_test, predict, 3)
      calculateMetrics("RFCNN Selected Features", y_test, predict)

    font1 = ('times', 12, 'bold')
    text=Text(main,height=30,width=100)
    scroll=Scrollbar(text)
    text.configure(yscrollcommand=scroll.set)
    text.place(x=10,y=100)
    text.config(font=font1)

    main.config(bg='turquoise')
    main.mainloop()
```

8. SYSTEM TEST

SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

1.1 Unit Testing:

Unit testing involves testing individual components of the software to ensure they function correctly. It verifies that internal logic works as expected, with valid inputs producing valid outputs. In RFCNN: Traffic Accident Severity Prediction, unit tests check the correctness of feature extraction, data preprocessing, and individual classifier implementations before integration.

1.2 Integration Testing:

Integration testing validates that multiple components of the software interact correctly when combined. It ensures that integrated modules work as expected. In RFCNN, this testing ensures that machine learning models, data processing pipelines, and UI components integrate seamlessly without conflicts.

1.3 Functional Testing :

Functional testing ensures that the software meets business and technical requirements. It validates the correctness of input handling, expected outputs, and feature execution. In RFCNN, this involves testing functions such as dataset loading, feature extraction, classifier execution, and result visualization to ensure expected behavior.

1.4 System Testing :

System testing verifies that the complete, integrated system meets all functional and non-functional requirements. It ensures correct system configuration and end-to-end workflows. In RFCNN, this involves checking whether the full model pipeline—from data input to severity prediction—operates without issues and produces accurate results.

1.5 White Box Testing :

White box testing examines the internal logic and structure of the code. It helps in identifying logical errors and optimizing the implementation. In RFCNN, white box testing focuses on the implementation of classifiers, data processing logic, and feature selection methods to verify proper execution.

1.6 Black Box Testing :

Black box testing validates the software without knowledge of its internal workings. It focuses on user interactions and expected results. In RFCNN, this includes testing the user interface, verifying input data formats, and ensuring correct predictions without inspecting the internal model structures.

9. RESULTS AND DISCUSSION

In propose paper author is introducing new machine learning algorithm called RFCNN (Random Forest Convolution Neural Neural) to predict accident severity. In propose algorithm author combining two models called RF and CNN to enhance accident prediction accuracy and then compare its performance with various machine learning algorithms called Random Forest (RF), Ada Boost (AC), Extra Tree Classifier (ETC), Gradient Boosting (GBM), Voting Classifier (VC (LR+SGD)), CNN.

In all algorithms propose RFCNN is giving high accuracy, to train all algorithms author using USA Road Accident Dataset which consist of 47 features and author evaluating all algorithms performance using all 47 features (full features) and 23 selected features and for both dataset variants RFCNN is giving better accuracy.

Below screen showing dataset details

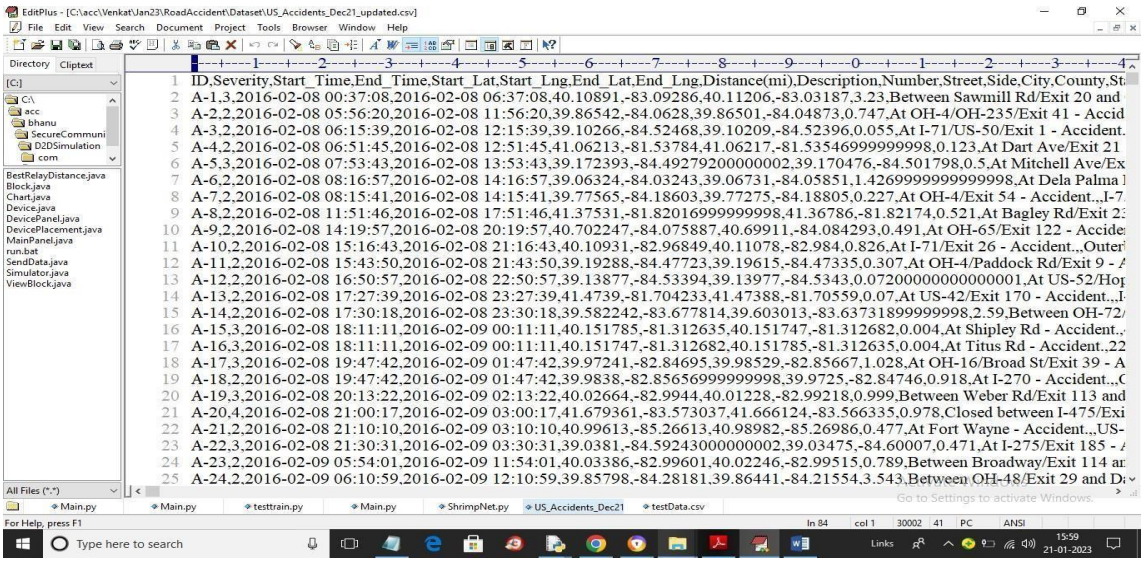


Fig 9.1 Dataset Screens

In above screen first row contains dataset column names and remaining rows contains dataset values and by using above dataset we will trained all algorithms and test their performance in terms of accuracy, precision, recall and FSCORE.

To implement this project we have designed following modules

1) Upload US Road Accident Dataset: using this module we will upload dataset to application and then find and plot missing values graph

2) Split Train & Test Data: using this module we will split dataset into train and test where application used 80% dataset for training and 20% for testing

3) Run Classifiers on Full Features: using this module we will trained all algorithms on full dataset features and calculate accuracy and precision values

4) Run Classifiers on Selected Features: using this module we will trained all algorithms on selected dataset features and calculate accuracy and precision values

5) Comparison Graph: using this module we will plot comparison graph between all algorithms on full and selected features

6) Comparison Table: this module we display comparison table of all algorithms

7) Predict Accident Severity from Test Data: using this module we will upload test data and then machine learning algorithm will predict severity of accident from test data

To run project double click on 'run.bat' file to get below screen

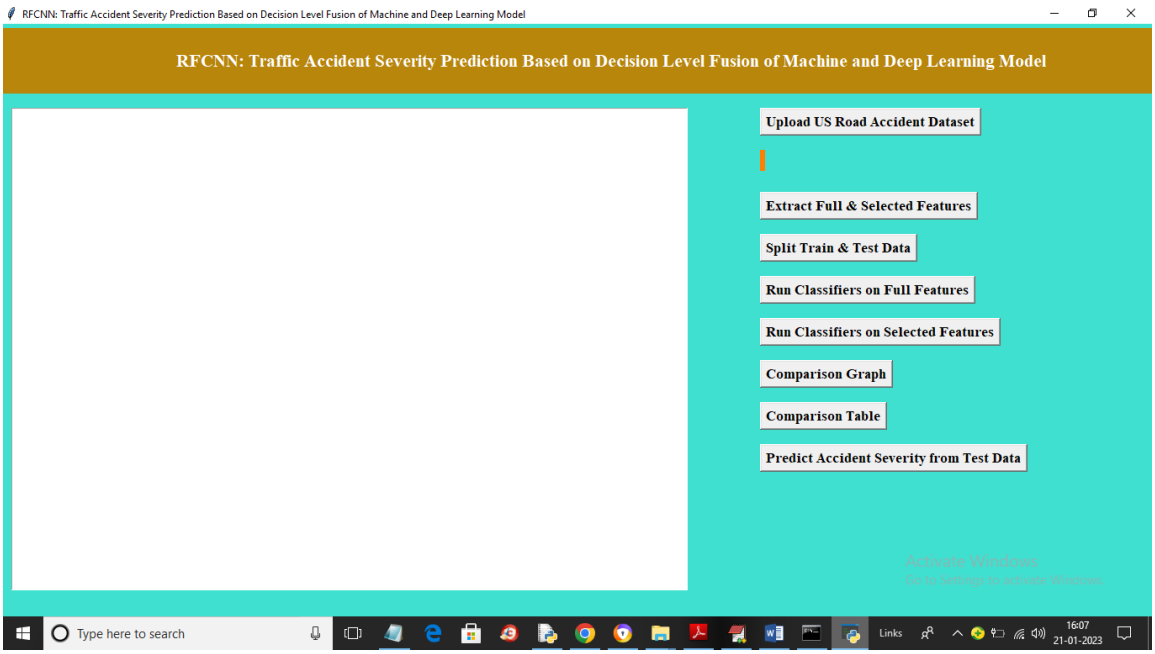


Fig 9.2 Upload US Accident Dataset

In above screen click on 'Upload US Road Accident Dataset' button to upload dataset and get below screen

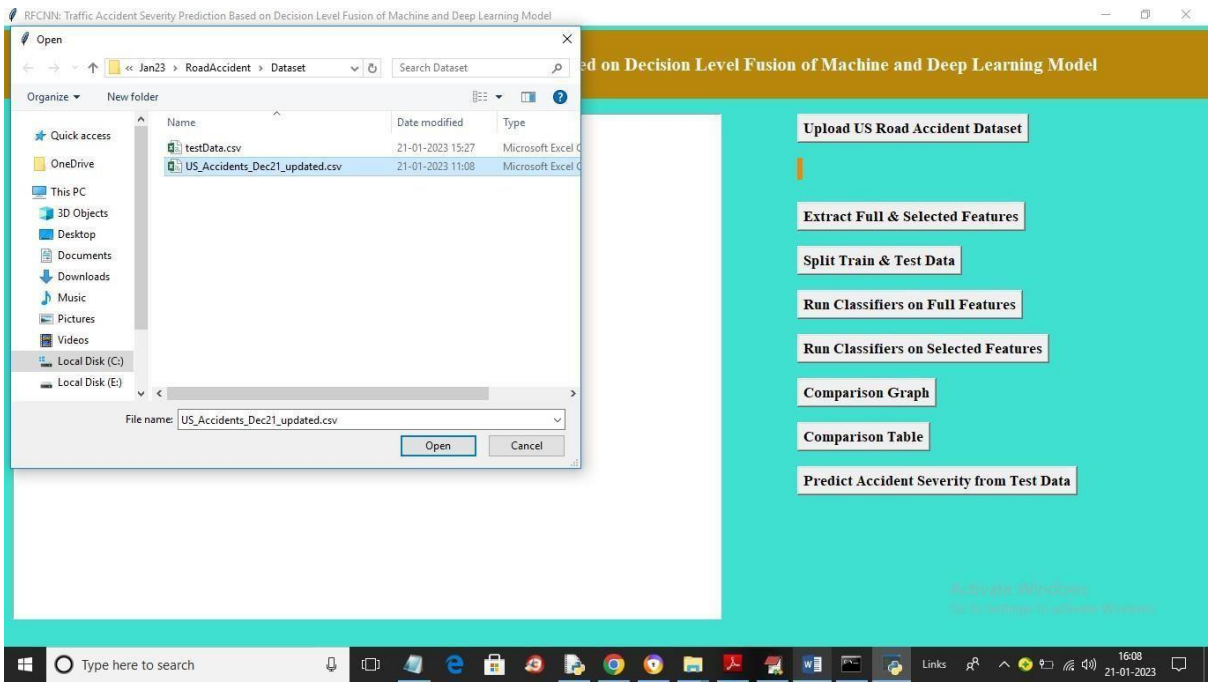


Fig 9.3 Selecting US Road Accident dataset

In above screen selecting and uploading dataset file and then click on 'Open' button to load dataset and get below output

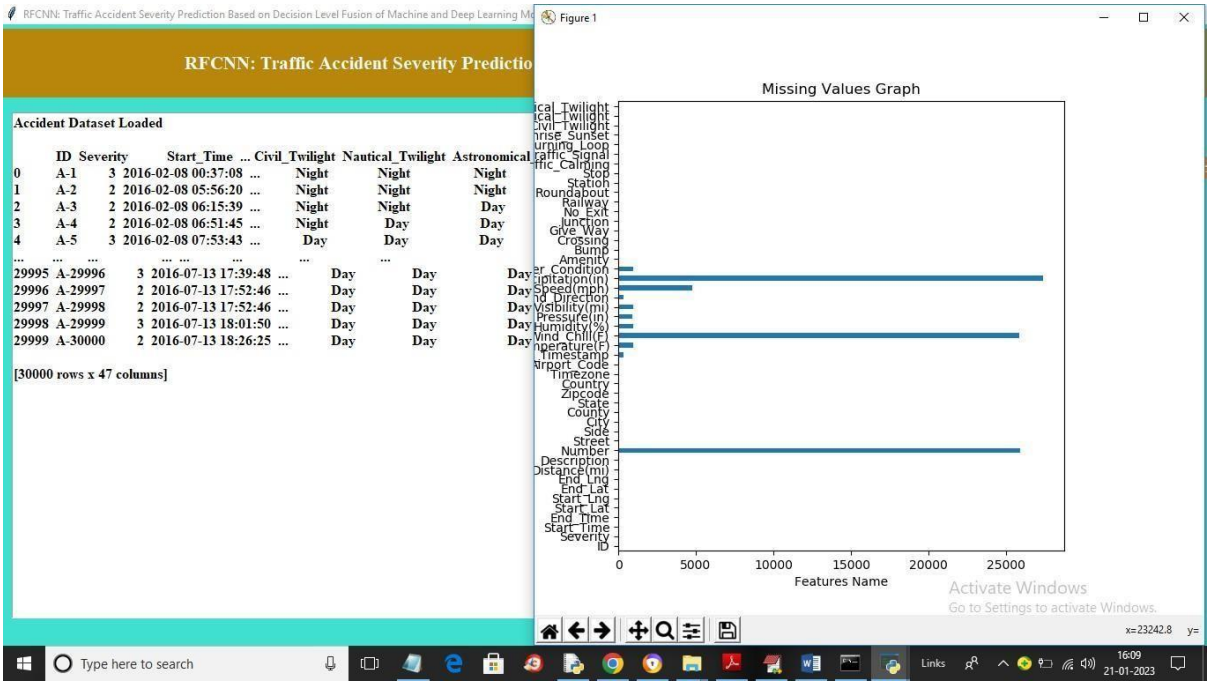


Fig 9.4 Traffic Accident Dataset Overview with Missing Values Analysis

In above screen dataset loaded and in graph x-axis represents count of missing values and y-axis represents name of features. In above graph we can see dataset contains so many missing features and now click on 'Extract Full & Selected Features' button to replace missing values and then separate full .

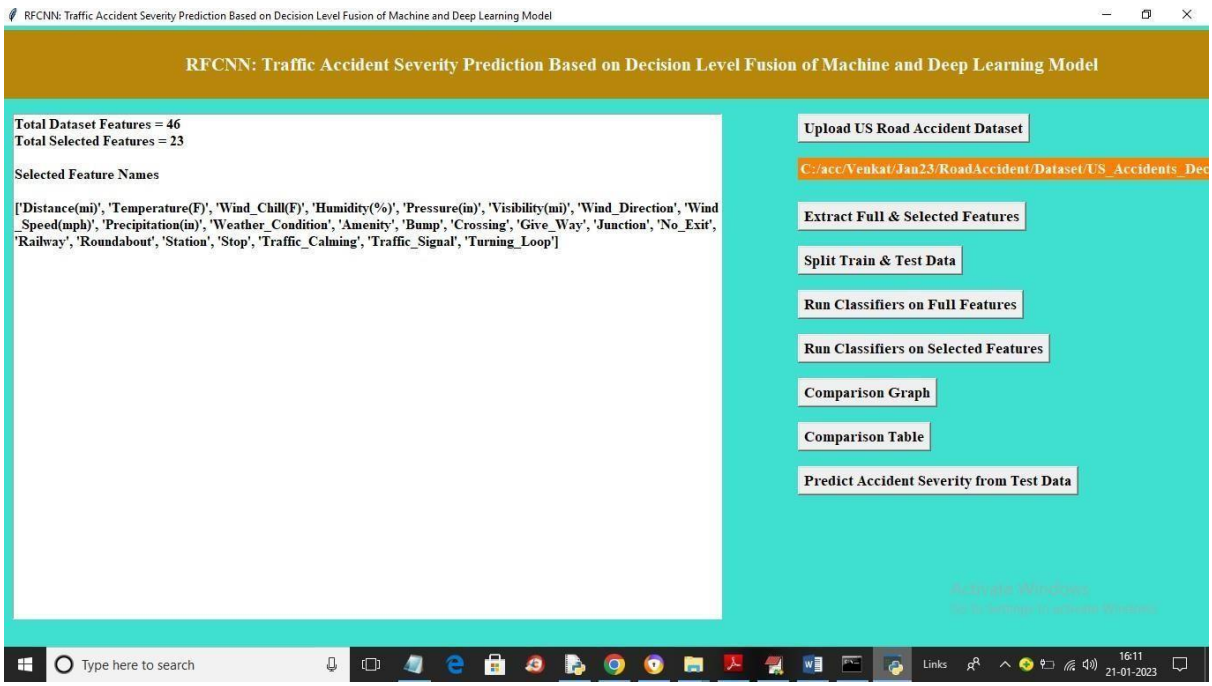
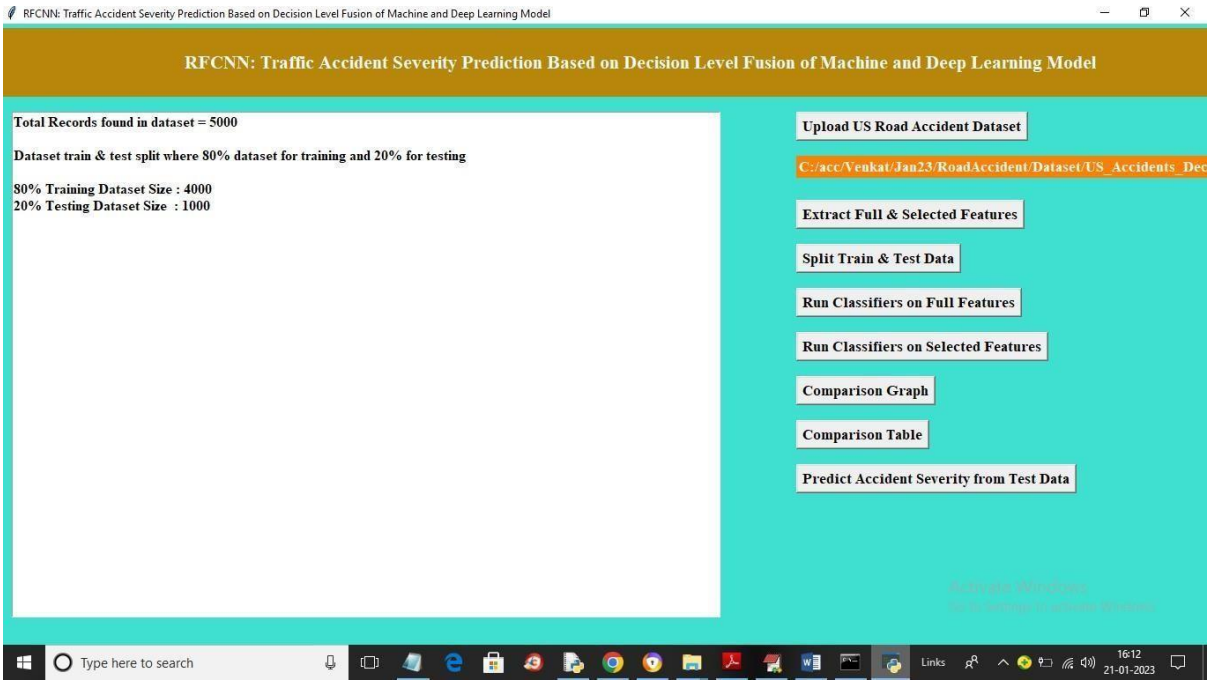


Fig 9.5 Feature Selection for Traffic Accident Severity Prediction

In above screen we can see dataset contains full column or features as 46 and then we selected 23 as selected features and the in next lines we can see names of selected features and now click on 'Split Train & Test Data' button to split dataset



into train and test and get below output.

Fig 9.6 Dataset Split for Training and Testing in Accident Severity Prediction

In above screen we are using 5000 records from dataset and application using 4000 records for training and 1000 for testing and now click on 'Run Classifiers on Full Dataset' button to train all machine learning classifiers on training data and test on

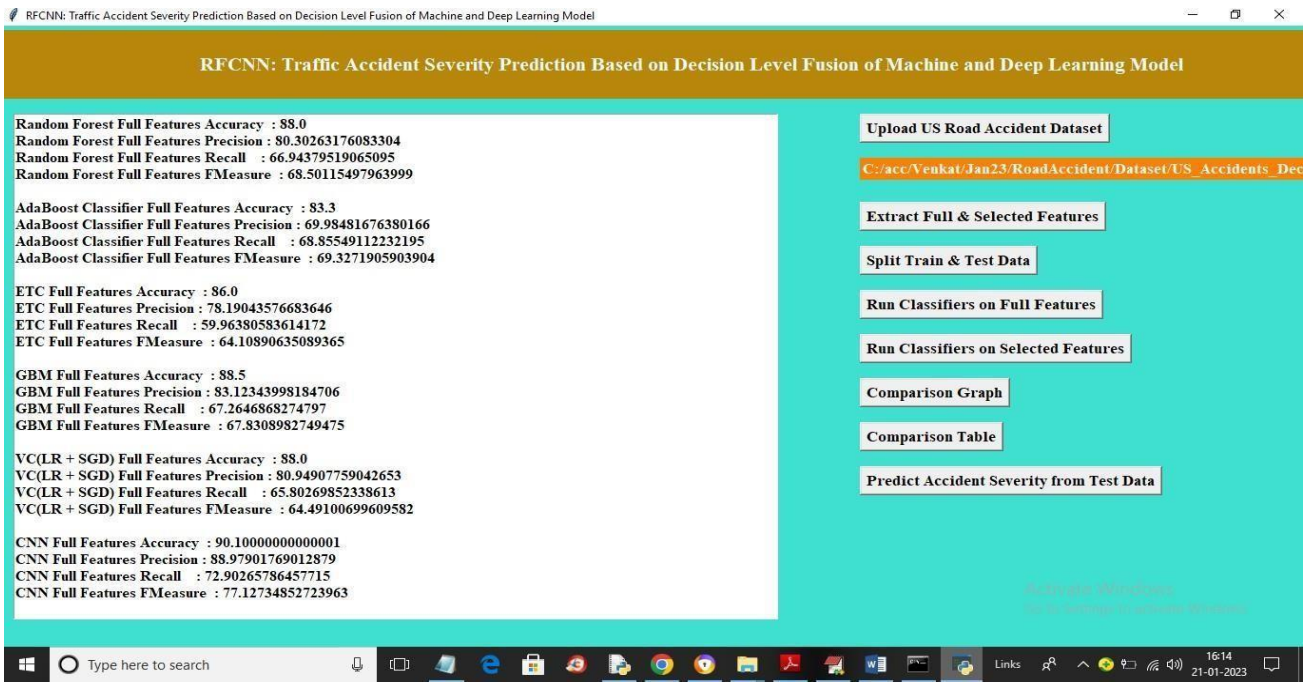


Fig 9.7 Model Performance Metrics for Traffic Accident Severity Prediction

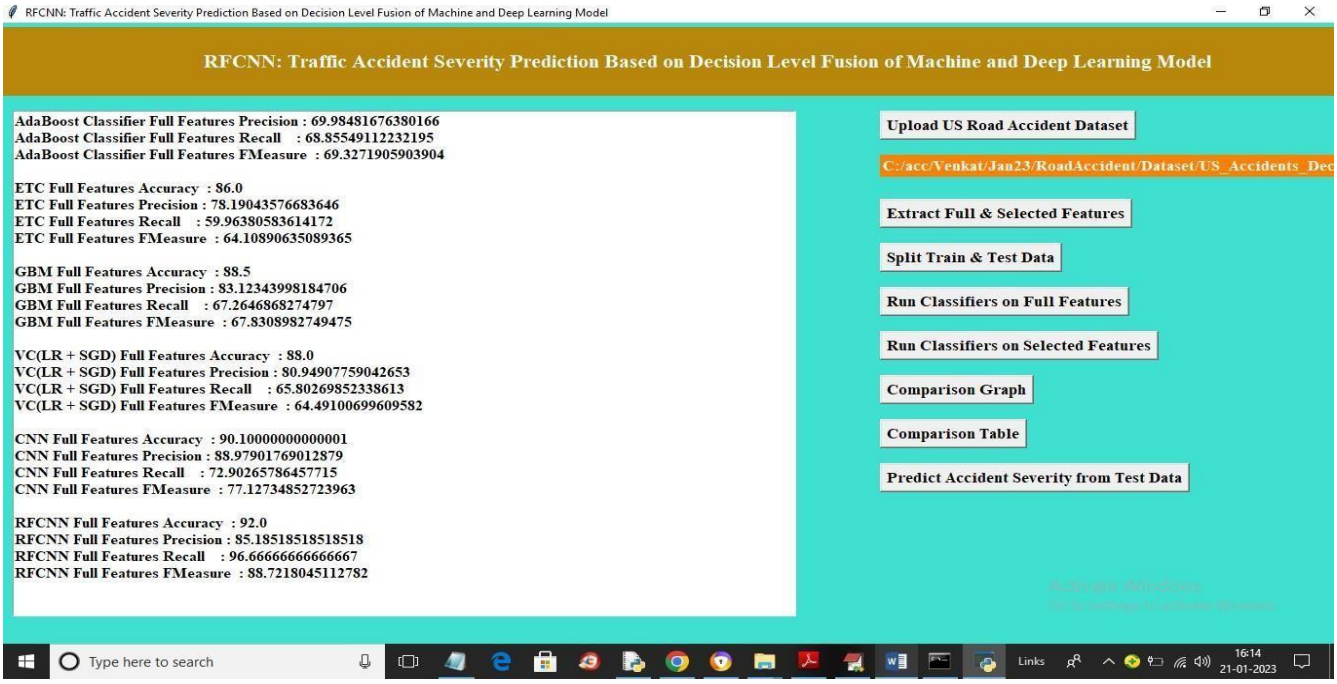


Fig 9.8 Results of Base Models

In above 2 screens we can see accuracy and other metrics on all features for all algorithms and we can see propose RFCNN got high accuracy as 92% and now click on 'Run Classifiers in Selected Features' button to train all algorithm on selected features and get below output

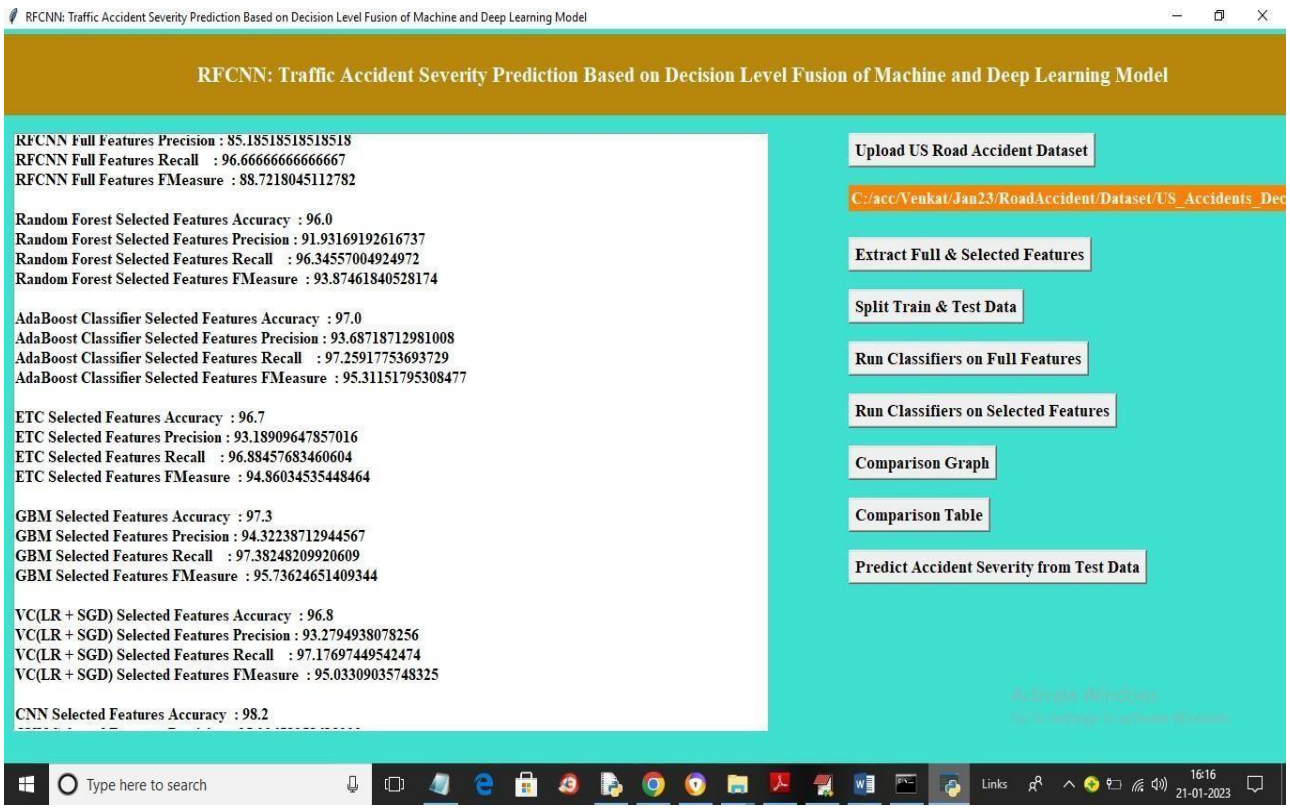


Fig 9.9 Final Model Evaluation Metrics for Traffic Accident Severity Prediction

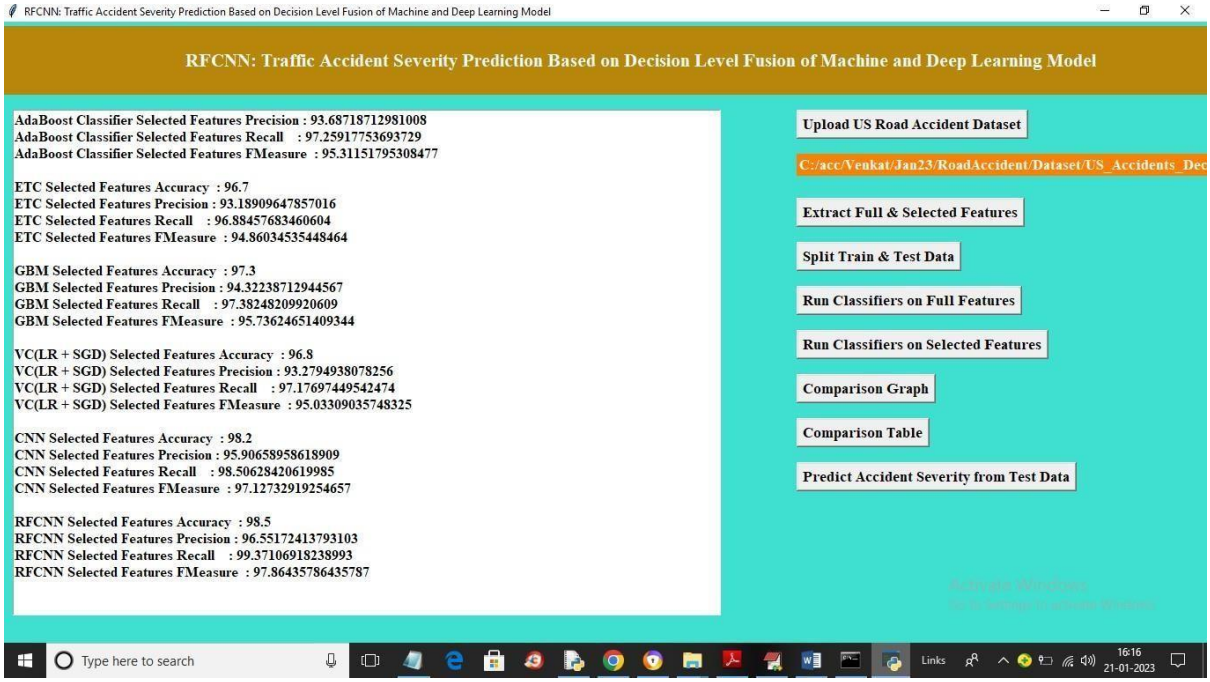


Fig 9.10 Performance Metrics of Selected Features for Accident Severity Prediction Models

In above screen all algorithm got more than 97% accuracy on selected features and propose RFCNN got high accuracy as 98.50% and now click on 'Comparison Graph' button to get below graph

Fig 9.11 Performance Comparison of Machine Learning Models for Accident Severity Prediction

In above graph x-axis represents algorithm names on full and selected features and y-axis represents accuracy and other metrics in different colour bars and in all we can see selected features algorithm got high performance and now click on 'Comparison Table' button to get below output

Algorithm Name	Accuracy	Precision	Recall	F1 SCORE
RF Full Features	87.8	77.21647141469865	65.93574635241302	67.79520951831607
AC Full Features	84.8	68.24090533353817	67.85353535353535	67.58529239534772
ETC Full Features	86.5	76.14489214489214	62.5794986906098	66.12354641392754
GBM Full Features	88.6	78.35386368942171	68.36980920314252	68.5556571632843
VC(LR+SGD) Full Features	88.0	71.0356133652581	66.02787130564907	63.69036699060169
CNN Full Features	90.4	88.42895099454763	72.58417508417509	76.35831527918134
RFCNN Full Features	92.0	82.54901960784315	89.4194756554307	85.13409961685824

Table 2 Results of Base models on all Features

This is the table showing accuracy and other metric values on full features and second table showing accuracy on selected features and in both tables we can see Propose RFCCN got high

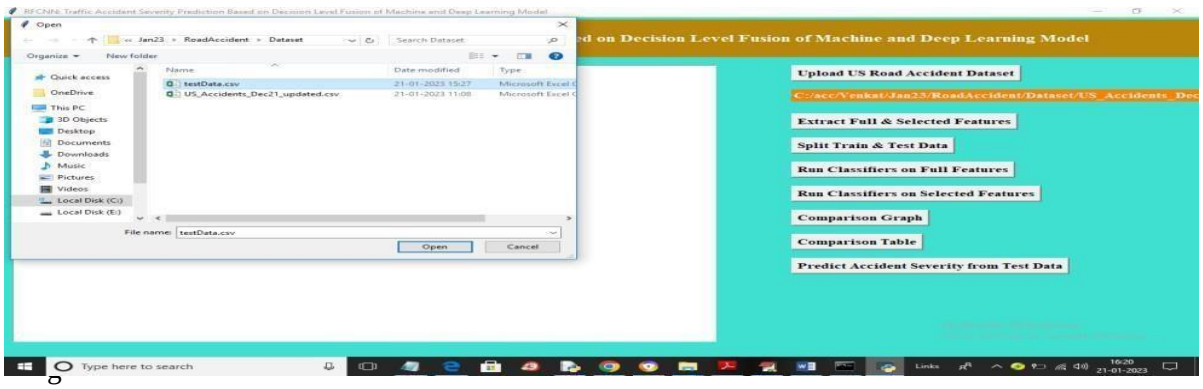


Fig 9.12 Uploading test data

In above screen selecting and uploading 'TestData.csv' file and then click on 'Open' button to load test data and get below output

Algorithm Name	Accuracy	Precision	Recall	FSCORE
RF Full Features	96.0	92.5202197424512	97.02224044676157	94.48329448329447
RF Full Features	97.0	94.08777407953882	97.87488853741563	95.79000182448458
RF Full Features	96.7	93.5495583740057	97.74894395303275	95.40929941288047
RF Full Features	97.3	94.64777680906712	98.0008331217985	96.1772887603454
RF Full Features	96.8	93.72662196191608	97.79092548116039	95.53548215462565
RF Full Features	98.2	96.18492618492618	98.81149968432496	97.41292323791016
RF Full Features	98.5	97.22222222222221	99.34210526315789	98.21849872405991

Table 3 Results of the Selected features

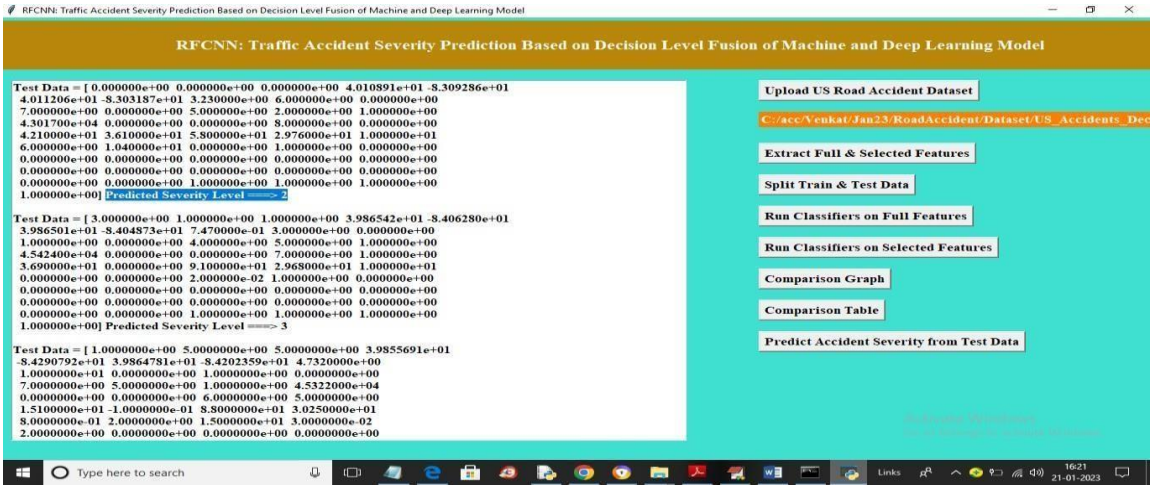


Fig 9.13 Traffic Accident Severity Prediction Using Machine and Deep Learning Fusion

In above screen in square bracket we can see test data and after =➔ arrow symbol we can see accident severity level as 2 or 3.

10. CONCLUSION AND FUTURE SCOPE

10.1 CONCLUSION :

The Ensemble Fusion Classifier (EFC) model significantly enhances accident severity prediction by integrating Machine Learning (ML) and Deep Learning (DL) techniques. By leveraging Random Forest (RF) for feature selection and Convolutional Neural Networks (CNN) for deep feature extraction, the model achieves higher accuracy, precision, and recall than traditional approaches. The ability to analyze real-time and historical traffic data allows for more proactive traffic management, improved emergency response, and better resource allocation. The EFC model overcomes the limitations of rule-based, statistical, and standalone ML models by incorporating a soft voting mechanism, ensuring robust and adaptable predictions. Its scalability and efficiency make it suitable for intelligent transportation systems (ITS), urban planning, and road safety regulations. With data-driven decision-making, traffic authorities can minimize accident risks and reduce fatalities. The model's capability to handle large datasets, spatial-temporal dependencies, and real-time updates further strengthens its reliability. By addressing challenges like data imbalance and computational inefficiencies, EFC proves to be a powerful tool in modern traffic accident analysis.

10.2 FUTURE SCOPE :

The EFC model holds immense potential for future advancements in smart transportation and road safety. One promising direction is real-time integration with IoT-enabled traffic systems, allowing for dynamic accident risk assessment and traffic flow optimization. Incorporating edge computing and federated learning can further enhance real-time accident prediction while maintaining data privacy. Future improvements may also include multi-modal data fusion, integrating sensor data, driver behavior analytics, and weather conditions for a more comprehensive accident prediction system. Expanding the model’s capabilities to predict accident hotspots over extended time frames can assist city planners in designing safer road infrastructures.

11. REFERENCES

[1] World Health Organization, Global Status Report on Road Safety 2015, World Health Org., Geneva, Switzerland, 2015.

[2] W. Gissane, ‘‘Accidents—A modern epidemic,’’ J. Inst. Health Educ., vol. 3, no. 1, pp. 16–18, 1965.

[3] H. Manner and L. Wünsch-Ziegler, ‘‘Analyzing the severity of accidents on the German autobahn,’’ Accident Anal. Prevention, vol. 57, pp. 40–48, Aug. 2013.

[4] J. C. Milton, V. N. Shankar, and F. L. Mannering, ‘‘Highway accident severities and the mixed logit model: An exploratory empirical analysis,’’ Accident Anal. Prevention, vol. 40, no. 1, pp. 260–266, 2008.

[5] N. V. Malyshkina and F. L. Mannering, ‘‘Markov switching multinomial logit model: An application to accident-injury severities,’’ Accident Anal. Prevention, vol. 41, no. 4, pp. 829–838, Jul. 2009.

[6] K. Haleem, P. Alluri, and A. Gan, ‘‘Analyzing pedestrian crash injury severity at signalized and non-signalized locations,’’ Accident Anal. Prevention, vol. 81, pp. 14–23, Aug. 2015.

[7] M. Bedard, G. H. Guyatt, M. J. Stones, and J. P. Hirdes, ‘‘The independent contribution of driver, crash, and vehicle characteristics to driver fatalities,’’ Accid Anal. Prevention, vol. 34, no. 6, pp. 717–727, 2002.

[8] S. S. Zajac and J. N. Ivan, ‘‘Factors influencing injury severity of motor vehicle– crossing pedestrian crashes in rural connecticut,’’ Accident Anal. Prevention, vol. 35, no. 3,
pp. 369–379, May 2003.

[9] W.-H. Chen and P. P. Jovanis, ‘‘Method for identifying factors contributing to driver- injury severity in traffic crashes,’’ Transp. Res. Rec., J. Transp. Res. Board, vol. 1717, no. 1,
pp. 1–9, Jan. 2000.

[10] S. Sarkar, S. Vinay, R. Raj, J. Maiti, and P. Mitra, ‘‘Application of optimized machine learning techniques for prediction of occupational accidents,’’ Comput. Oper. Res., vol. 106,
pp. 210–224, Jun. 2019.

[11] K. Rumar, ‘‘Transport safety visions, targets and strategies: Beyond 2000,’’ First Eur. Transp. Saf. Lecture, Brussels, Eur. Transp. Saf. Council, pp. 6–8, 1999.

[12] T. Hummel, ‘‘Land use planning in safer transportation network planning,’’ Leidschendam, Inst. Road Saf. Res., Citeseer, 2001

[13] van, J.N., Wang, C., & Bernardo, N.R. (2000). "Explaining two-lane highway crash rates using land use and hourly exposure." Accident Analysis & Prevention, 32(6), 787–795.

[14] Eluru, N., Bagheri, M., Miranda-Moreno, L.F., & Fu, L. (2012). "A latent class modeling approach for identifying vehicle driver injury severity factors at highway-railway crossings." Accident Analysis & Prevention, 47, 119–127.

[15] Savolainen, P.T., Mannering, F.L., Lord, D., & Quddus, M.A. (2011). "The statistical analysis of highway crash-injury severities: A review and assessment of methodological alternatives." Accident Analysis & Prevention, 43(5), 1666–1676.

[16] Dissanayake, S., & Lu, J.J. (2002). "Factors influential in making an injury severity difference to older drivers involved in fixed object–passenger car crashes." Accident Analysis & Prevention, 34(5), 609–618.

[17] Abdel-Aty, M. (2003). "Analysis of driver injury severity levels at multiple locations using ordered probit models." Journal of Safety Research, 34(5), 597–603.

[18] Kim, J.K., Ulfarsson, G.F., Shankar, V.N., & Mannering, F.L. (2013). "A note on modeling pedestrian-injury severity in motor-vehicle crashes with the mixed logit model." Accident Analysis & Prevention, 50, 1266–1272.

[19] Lee, C., & Abdel-Aty, M. (2005). "Comprehensive analysis of vehicle–pedestrian crashes at intersections in Florida." Accident Analysis & Prevention, 37(4), 775–786.

[20] Shankar, V.N., Mannering, F., & Barfield, W. (1996). "Statistical analysis of accident severity on rural freeways." Accident Analysis & Prevention, 28(3), 391–401.

[21] Zhu, X., & Srinivasan, S. (2011). "A comprehensive analysis of factors influencing the injury severity of large-truck crashes." Accident Analysis & Prevention, 43(1), 49–57.

[22] Chen, F., Chen, S., & Ma, X. (2016). "Crash frequency modeling using real-time environmental and traffic data and its application in site ranking." Accident Analysis & Prevention, 96, 254–264.

[23] Islam, S., & Mannering, F. (2006). "Driver aging and its effect on male and female single- vehicle accident injuries: Some additional evidence." Journal of Safety Research, 37(3), 267– 276.

[24] Yamamoto, T., & Shankar, V.N. (2004). "Bivariate ordered-response probit model of driver's and passenger's injury severities in collisions with fixed objects." Accident Analysis & Prevention, 36(5), 869–876.

[25] Zajac, S.S., & Ivan, J.N. (2003). "Factors influencing injury severity of motor vehicle– crossing pedestrian crashes in rural Connecticut." Accident Analysis & Prevention, 35(3), 369– 379.

[26] Quddus, M.A., Noland, R.B., & Chin, H.C. (2002). "An analysis of motorcycle injury and vehicle damage severity using ordered probit models." Journal of Safety Research, 33(4), 445– 462.

[27] Kockelman, K.M., & Kweon, Y.J. (2002). "Driver injury severity: An application of ordered probit models." Accident Analysis & Prevention, 34(3), 313–321.

[28] Duncan, C., Khattak, A.J., & Council, F.M. (1998). "Applying the ordered probit model to injury severity in truck-passenger car rear-end collisions." Transportation Research Record: Journal of the Transportation Research Board, 1635(1), 63–71.

[29] Zhou, M., & Sisiopiku, V.P. (1997). "Relationship between volume-to-capacity ratios and accident rates." Transportation Research Record: Journal of the Transportation Research Board, 1581(1), 47–52.

[30] Khattak, A.J., & Targa, F. (2004). "Injury severity and total harm in truck-involved work zone crashes." Transportation Research Record: Journal of the Transportation Research Board, 1877(1), 106–116.

[31] Chang, L.Y., & Wang, H.W. (2006). "Analysis of traffic injury severity: An application of non-parametric classification tree techniques." Accident Analysis & Prevention, 38(5), 1019– 1027.