**Analysing Severity, Weather Impact, And Driver Characteristics for Predictive Modelling Using Global Road Accident Dataset**

**ABSTRACT**

Road accidents are a major global concern, causing significant human and economic losses. Understanding accident severity and its contributing factors is essential for developing effective road safety measures. This study focuses on analysing accident severity based on weather conditions and driver characteristics using a global road accident dataset. The dataset includes key attributes such as accident location, time of occurrence, road type, weather conditions, driver fatigue, alcohol levels, and vehicle conditions. In recent years, traffic-related fatalities have risen, with the World Health Organization (WHO) reporting that over 1.3 million people die in road crashes annually, and millions suffer injuries. Traditional accident analysis methods rely on statistical modeling and historical trends, but these approaches lack predictive capabilities and fail to capture complex relationships among multiple variables. This research utilizes a data-driven approach to identify patterns and correlations between accident severity and influencing factors. By leveraging predictive modeling, this study aims to enhance road safety strategies, improve emergency response times, and assist policymakers in developing targeted interventions. The findings from this analysis will contribute to designing data-driven traffic policies, improving road infrastructure, and developing proactive safety measures to reduce accident severity and fatalities globally.

**CHAPTER 1**

**INTRODUCTION**

**1.1 Overview**

Road accidents have been a critical global issue, leading to millions of deaths and economic losses every year. According to WHO, road traffic crashes are the leading cause of death for individuals aged 5–29 years, with low- and middle-income countries accounting for over 90% of fatalities. In India alone, the Ministry of Road Transport and Highways (MoRTH) reported over 450,000 accidents in 2022, resulting in approximately 150,000 deaths and 400,000 injuries. The economic loss due to road accidents in India is estimated to be around 3% of its GDP. The rise in urbanization and vehicular density has contributed to increased accident rates. Factors such as adverse weather conditions, poor road infrastructure, driver fatigue, and high traffic volume play a significant role in accident severity. Traditionally, accident analysis relied on descriptive statistics and historical data trends, which failed to provide accurate predictive insights. This study aims to develop a predictive model that integrates multiple factors, such as weather conditions, driver characteristics, and road conditions, to assess accident severity. By analyzing real-world accident data, this research will contribute to enhancing road safety, optimizing emergency response strategies, and reducing traffic-related fatalities.

**1.2 Research Motivation**

The increasing number of road accidents worldwide poses a significant challenge to traffic management and public safety. Despite advancements in vehicle safety technologies and traffic regulations, accident severity remains high due to unpredictable human behavior, environmental factors, and road conditions.

Traditional accident analysis methods primarily focus on historical data trends and do not account for real-time influencing factors. Moreover, existing models often fail to incorporate crucial elements such as driver fatigue, alcohol influence, and emergency response time. The need for an advanced data-driven approach that considers multiple factors in predicting accident severity is more critical than ever.

A well-developed predictive model can provide valuable insights into accident-prone conditions, enabling authorities to implement preventive measures. By understanding how weather conditions, traffic volume, and driver characteristics contribute to accident severity, this study aims to support the development of proactive safety policies, infrastructure improvements, and emergency response optimizations.

**1.3 Problem Statement**

Road traffic accidents are a leading cause of mortality and economic loss worldwide. Despite numerous safety measures, accident rates continue to rise, particularly in urban areas with high population density. The key challenges in accident analysis include:

1. **Lack of Real-Time Insights**: Existing accident analysis models primarily rely on historical trends without considering real-time influencing factors such as weather, traffic volume, and emergency response time.
2. **Inadequate Predictive Capabilities**: Traditional statistical approaches fail to capture complex relationships among multiple accident-related variables.
3. **Human Factor Negligence**: Driver fatigue, alcohol consumption, and vehicle condition play a crucial role in accident severity but are often overlooked in conventional models.
4. **Limited Policy Implementation**: Without accurate predictions, policymakers struggle to design effective safety measures to prevent accidents and reduce fatalities.

**1.4 Need and Significance**

* **Enhanced Road Safety**: Predictive modeling helps identify high-risk conditions, enabling authorities to implement preventive measures.
* **Improved Traffic Management**: Analyzing traffic volume and accident causes assists in optimizing road design and regulations.
* **Better Emergency Response**: Understanding emergency response time impact can improve accident survival rates.
* **Data-Driven Policymaking**: Provides insights to design evidence-based road safety policies.
* **Personalized Driver Awareness Programs**: Identifies at-risk driver categories and promotes targeted safety campaigns.

**1.5 Applications**

* **Traffic Management Systems**: Using accident data to optimize traffic flow and road design.
* **Smart City Initiatives**: Integrating real-time accident predictions into intelligent transportation systems.
* **Insurance Risk Assessment**: Assisting insurance companies in evaluating accident risks and premium calculations.
* **Law Enforcement**: Supporting police and traffic authorities in accident-prone zone monitoring.
* **Public Awareness Campaigns**: Developing educational programs based on accident trends and severity analysis.

**CHAPTER 2**

**LITERATURE SURVEY**

Santos et al. [1] investigate ML methodologies for analyzing accident severity and identifying potential hotspots using data from Setúbal, Portugal (2016–2019). They employ supervised techniques, including RF and decision trees, and unsupervised methods such as DBSCAN clustering. Their analysis shows that while the C5.0 algorithm was especially effective in identifying key factors influencing accident severity, RF demonstrated strong capabilities in predicting accident hotspots. This consistent performance across both studies highlights RF’s versatility and effectiveness in TAS prediction and hotspot analysis. Gradient Boosting and Extreme gradient Boosting (Boost) share several common characteristics as both belong to the gradient boosting family of ensemble learning methods. Their application in predicting TAS has shown significant promise across various studies, particularly for their ability to handle class imbalance and improve predictive accuracy.

The research conducted by Zhang et al. [2] introduces a novel methodology for forecasting the severity of injuries resulting from traffic accidents, employing ordinal classification techniques. In contrast to conventional nominal classification methods that fail to account for the ordered nature of injury severity, this investigation establishes a framework that adheres to the principles of rank consistency and monotonicity. The authors evaluate the efficacy of three distinct classifiers: Multi-Layer Perceptron (MLP), XGBoost, and Support Vector Machine (SVM), implementing a severity category-combination strategy to mitigate class imbalance within the crash data. The findings indicate that the proposed ordinal classification approach markedly surpasses traditional nominal classification methods, achieving an accuracy rate of 85% on a practical crash dataset. Additionally, key determinants affecting injury severity are discerned through permutation feature importance analysis. The study concludes that categorizing severity levels into three distinct classes—minor, moderate, and serious injuries—improves predictive accuracy while effectively addressing the issues associated with class imbalance in traffic accident data.

In another study, Behboudi et al. [3] conduct an extensive examination of various ML techniques utilized for predicting the severity of traffic accidents. The research assesses multiple models, such as RF, SVM, and Gradient Boosting, leveraging a dataset encompassing traffic accident records from numerous urban locations over an extended period. The authors note that the Gradient Boosting model achieved the highest performance, with an accuracy rate of 92%, underscoring its capability to manage intricate data and address class imbalances frequently encountered in traffic accident datasets. Additionally, the study underscores the significance of feature selection and data preprocessing in improving model efficacy. Nonetheless, the authors recognize certain limitations concerning the applicability of their findings across different geographical areas and accident types, indicating a need for further investigation to confirm these models in varied contexts. Among ML models, logistic regression is considered as a statistical model widely used for binary classification tasks.

Chong et al. [4] investigate various ML algorithms, such as Logistic Regression, XGBoost, and RF, to assess the severity of traffic accidents using data from Texas spanning the years 2011 to 2021. Their findings indicate that Logistic Regression achieved the highest performance, with an accuracy rate of 88%, successfully pinpointing key factors that lead to accidents. The authors emphasize the importance of precise predictive models in reducing future incidents and enhancing road safety initiatives. Although their research provides significant insights into the classification of accident severity, they acknowledge that traditional models like Logistic Regression may not adequately capture more intricate relationships compared to DL methodologies. This study enhances the understanding of how various ML techniques can be utilized to improve predictions related to traffic safety.

The research conducted by Aboulola et al. [5] explores the prediction of traffic accident severity through the implementation of diverse transfer learning techniques, while simultaneously improving model interpretability via Shapley values. The study employs a MobileNet architecture, achieving a notable accuracy of 98.17%. The dataset under examination spans five years (2016–2020) from New Zealand and includes factors such as accident location, weather conditions, and vehicle attributes. By clarifying the influence of various features on the severity of accidents, the research aims to bolster safety protocols and enhance predictive precision. However, the reliance on large labeled datasets and the significant computational resources required for explainability methods may present obstacles to the practical application of the findings in real-time contexts, especially in environments with limited resources. Recent advancements in DL (Neural Network) approaches, including RNN, MobileNet, and Deep Spatiotemporal Hybrid Network (DSHN), have significantly contributed to predicting TAS.

The research conducted by Sameen et al. [6] examines the application of RNNs in forecasting TAS. The proposed model achieved an accuracy rate of 71.77% using a dataset sourced from the Crash Analysis System in New Zealand, covering the years 2016 to 2020. However, the study acknowledges certain limitations, such as dataset imbalance, which may influence the model’s predictive performance, as well as concerns regarding potential overfitting during the training process. In a further study, Aboulola et al. [7] explore the use of transfer learning methodologies utilizing MobileNet to assess TAS. By analyzing a dataset from New Zealand collected between 2016 and 2020, the authors achieved a remarkable accuracy rate of 98.17%, demonstrating the model’s proficiency in categorizing accident severity levels. This research emphasizes the importance of XAI approaches, enabling stakeholders to understand critical factors contributing to accidents. The implementation of MobileNet addresses the limitations of traditional models, which often lack interpretability. Nevertheless, the authors caution that the model’s effectiveness may be confined to the specific dataset utilized, raising concerns about its applicability to different geographical areas or datasets. Overall, this study significantly contributes to the field by showcasing how sophisticated ML techniques can enhance predictions related to traffic safety.

In a different research project, Kashifi et al. [8] introduce an innovative methodology that leverages the DSHN to enhance traffic accident prediction. This framework integrates diverse data sources, including extensive traffic datasets from the Paris road network, meteorological conditions, and historical accident records. The DSHN combines CNN, LSTM, and Artificial Neural Networks (ANN), yielding an accuracy rate of 75.7% and an area under the curve (AUC) of approximately 0.800. This research highlights the critical role of road sensor data in improving predictive accuracy while addressing the complexities associated with data integration and preprocessing. Deep Forest is a notable instance of Ensemble ML, classified as a Tree-Based Ensemble and Non-Parametric method. In their study, Jing Gan et al. [9] proposed the Deep Forest algorithm as a viable alternative for predicting TAS. This innovative methodology addresses key limitations of traditional DL models, particularly their dependency on large datasets and substantial computational resources.

The Deep Forest algorithm, which utilizes an ensemble of decision trees, was rigorously evaluated using the 2016 road safety dataset from the UK. The results revealed that this approach attained an impressive accuracy rate of 85.23%, while exhibiting a markedly reduced demand for computational resources compared to conventional neural networks. However, the authors recognized that the scalability of Deep Forests might be constrained in contexts involving large datasets. This research offers a more accessible yet effective strategy for predicting accident severity, thereby contributing to the growing body of knowledge in traffic safety analytics. In their 2024 research, Saxena et al. [10] introduce an advanced version of the YOLOv4 model aimed at enhancing the detection of traffic signs in difficult conditions for autonomous vehicles. The model utilizes CSPDarknet53 as its foundational architecture and incorporates strategies such as nighttime image enhancement and K-Means clustering combined with GIoU for optimizing anchor boxes, which allows for improved detection of smaller traffic signs. Evaluated on datasets including Tsinghua-Tencent 100K (TT-100K) and Mapillary Traffic Sign Dataset (MTSD), the model achieved accuracy rates of 94.80% and 80.71%, respectively. Although this model surpasses previous approaches in performance, its high computational requirements pose a challenge for real-time implementation on devices with limited resources.

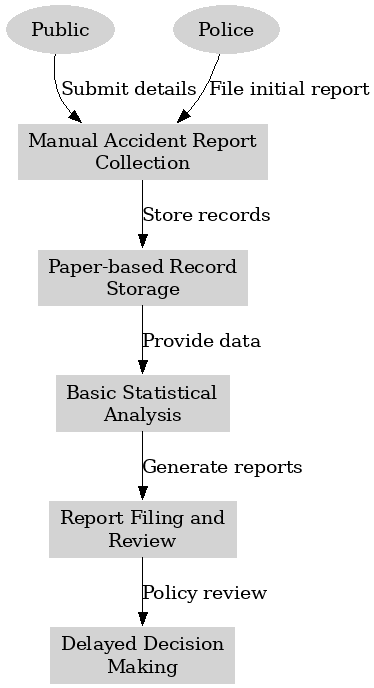
Theofilatos et al. [11] compared factors within and outside urban areas. Inside urban areas, factors such as young driver age, bicycles, intersections, and collisions with objects were found to affect accident severity; outside urban areas, weather, and head-on and side collisions affected accident severity. To forecast the severity of traffic accidents, Iranitalab and Khattak [12] compared Multinomial Logit (MNL), nearest neighbor classification (NNC), support vector machine (SVM), and RF analysis methods. The results show that NNC has the best overall prediction performance for more severe accidents, followed by RF, SVM, and MNL. Lin et al. [13] investigated various machine learning algorithms, such as random forest, K-nearest neighbor, and Bayesian network, to predict road accidents.The best model could predict 61% of accidents while having a false alarm rate of 38%. Chang and Chen [14] created a CART (classification and regression trees) model to train and test a classifier that predicts accidents with a training and testing accuracy of 55%. Caliendo et al. [15] used the Poisson, negative binomial, and negative multinomial regression models to predict the number of accidents on multi-lane highways.

According to Silva et al. [16], nearest neighbor classification, decision trees, evolutionary algorithms, support vector machines, and artificial neural networks are the usual techniques utilized for these purposes. Because of its capacity to deal with both regression and classification problems, as well as multivariate response models, the latter is employed in a variety of ways.To investigate the likelihood of road accidents, Theofilatos (2017) [17] used random forest and Bayesian logistic regression models on real-time traffic data from urban arterial roadways. More recently [18], compared several machine learning and deep learning techniques, including kNN, naive Bayes, classification tree, random forest, SVM, shallow neural network, and deep neural network, finding that the deep learning approach produced the best results, while other, less complex methods, such as naive Bayes, performed only slightly worse. Ren et al. [19] suggested a method for predicting traffic accident risk using long-short term memory (LSTM) model, where risk is defined as the number of accidents in a region at a given period.

**CHAPTER 3**

**TRADITIONAL SYSTEM**

The traditional system for analyzing road accident severity primarily relies on manual data collection, paper-based records, and basic statistical methods. Traffic departments and road safety agencies gather information from accident reports submitted by police or eyewitnesses, which are then analyzed using conventional statistical tools. These systems often focus on historical trends without incorporating real-time data or complex interdependencies between variables. Decision-making processes based on such systems tend to be reactive rather than proactive, limiting the ability to forecast and prevent accidents. Additionally, the traditional approach emphasizes descriptive reporting rather than predictive modeling, which restricts its effectiveness in anticipating future risks. The absence of automated systems results in delays in processing and evaluating critical accident information. Insights drawn from these systems often lack precision and fail to account for dynamic road, weather, and driver behavior conditions. Consequently, the traditional system falls short in delivering timely, accurate, and actionable intelligence for reducing accident severity and improving road safety.



**Limitations of Traditional System:**

* It depends heavily on manual data entry, leading to increased chances of human error and data inconsistency.
* There is no real-time data analysis, which prevents proactive measures and immediate intervention.
* Statistical methods used lack the complexity needed to identify intricate patterns and relationships between accident factors.
* It fails to integrate diverse data sources such as weather data, traffic flow, and road conditions in a cohesive manner.
* Decision-making is slower due to reliance on manual processes and historical data review.
* It lacks automation, resulting in time-consuming procedures for reporting, analyzing, and responding to accidents.
* Insights derived are often generalized and do not provide personalized or location-specific risk assessments.
* The system is not scalable or adaptable to the growing volume and variety of data generated in modern traffic environments.

**CHAPTER 4**

**PROPOSED SYSTEM**

The research process undertaken for analysing accident severity in relation to weather conditions and driver characteristics follows a structured and systematic approach. Each step has been designed to ensure the extraction of meaningful insights from data and the development of robust predictive models to assess the severity of road accidents. The methodology integrates data preprocessing, feature engineering, dimensionality reduction, classification models, performance evaluation, and predictive deployment. The use of two machine learning algorithms—K-Nearest Neighbors (KNN) as the existing model and Decision Tree Classifier (DTC) as the proposed model—provides a comparative framework to evaluate predictive strength and operational efficacy.

**Step 1: Uploading the Global Road Accident Dataset**

The research begins with the acquisition and uploading of a comprehensive global road accident dataset. This dataset includes a wide range of variables that are critical for understanding accident dynamics. The attributes include accident severity, location details, time of occurrence, road and weather conditions, driver behavior (such as fatigue and alcohol consumption), and vehicle status. This dataset serves as the foundational input for all subsequent analyses and model development. Ensuring access to a dataset of this scope and detail enables the study to evaluate various interacting factors that influence accident outcomes.

**Step 2: Data Preprocessing and Initial Exploration**

Once the dataset is uploaded, an in-depth data preprocessing phase is conducted. The objective of this phase is to assess the integrity and quality of the data. The initial exploration includes examining the data structure using descriptive methods such as .info() and .describe(), which provide insights into column types, value counts, and basic statistical distributions. A check for null values across all attributes helps identify incomplete or missing data points. Understanding the data at this level is crucial to ensure reliability in downstream tasks such as modeling and prediction. Any inconsistencies, missing values, or anomalies are addressed appropriately to prevent misleading results.

A count plot of accident severity categories is generated to visualize the distribution of the target variable. This plot illustrates the frequency of each severity class—Minor, Moderate, and Severe—within the dataset. The visual representation aids in understanding class imbalance, which is a common challenge in classification problems. By annotating the count values directly on the bars, the plot offers immediate clarity on the relative proportions, which helps in determining the need for techniques like SMOTE (Synthetic Minority Over-sampling Technique) to balance the classes in later stages.

**Step 3: Label Encoding and Principal Component Analysis (PCA)**

Following the initial data inspection, the next critical step is encoding categorical variables. Many features in the dataset are non-numeric (e.g., weather type, road surface conditions, vehicle types). Since machine learning algorithms operate on numerical data, categorical variables are transformed into numerical representations using label encoding. Each unique category in a feature is assigned a numeric value. This transformation preserves the essential distinctions between categories without introducing unintended ordinal relationships.

The target variable—accident severity—is also label encoded to convert the class labels (Minor, Moderate, Severe) into integer values. This prepares the target data for modeling while maintaining interpretability.

To address high dimensionality and potential multicollinearity, Principal Component Analysis (PCA) is applied to the feature set. PCA reduces the dataset to 15 principal components, effectively capturing the majority of the variance in the data while minimizing redundancy. This dimensionality reduction enhances computational efficiency and helps improve model generalization by eliminating noise and irrelevant correlations among input features. After applying PCA, the dataset becomes more compact, yet retains the essential information necessary for accurate classification.

**Step 4: Train-Test Splitting (80-20 Ratio)**

After preprocessing, the dataset is split into training and testing sets. A standard 80-20 split ratio is adopted, wherein 80% of the data is used for model training and 20% is reserved for testing and validation. This split ensures that the models learn patterns from the training data while being evaluated on previously unseen data for unbiased performance assessment. The training data feeds the learning algorithms, while the test data serves as a realistic proxy for evaluating model accuracy and generalization.

In practice, the data is randomly partitioned to prevent any chronological or grouped bias. The splitting process preserves the class distribution, ensuring that each subset contains a representative proportion of the target classes. The dimensions of the resulting training and testing sets confirm that a substantial amount of data is available for both model fitting and validation.

**Step 5: Building the Existing K-Nearest Neighbors (KNN) Model**

The first machine learning algorithm implemented is the K-Nearest Neighbors (KNN) classifier. This algorithm operates on the principle of instance-based learning, where classification is performed based on the majority class among the k-nearest neighbors in the feature space. KNN is non-parametric and does not assume any underlying data distribution, making it suitable for the given task.

Before training, a check is performed to determine if a pre-trained KNN model already exists in a serialized format (i.e., a .pkl file). If the model exists, it is loaded and used for prediction directly. If not, a new KNN model is instantiated and trained on the training data. Once trained, the model is serialized using joblib and saved to disk for future reuse. This approach ensures model persistence and reduces computational time during subsequent runs.

Predictions are made on the test set, and the model’s performance is evaluated using key metrics including accuracy, precision, recall, and F1-score. Additionally, a detailed classification report is generated, highlighting the performance across each class. A confusion matrix is visualized using a heatmap, providing insight into true positives, false positives, and false negatives for each severity class.

**Step 6: Building the Proposed Decision Tree Classifier (DTC) Model**

To improve classification performance and address the limitations of the KNN model, a Decision Tree Classifier (DTC) is introduced as the proposed model. Decision trees construct a tree-like structure based on feature splits that best separate the classes. The tree is built recursively by selecting the most informative features using criteria such as Gini index or information gain. This method offers high interpretability and handles both categorical and numerical data effectively.

Similar to the KNN model, a check is performed to load a pre-existing decision tree model. If available, the model is used for immediate prediction. If not, the model is trained on the training data and then saved for future use. The trained model predicts accident severity on the test dataset, and its performance is evaluated using the same metrics: accuracy, precision, recall, F1-score, classification report, and confusion matrix visualization.

The decision tree model reveals improved classification metrics and offers better interpretability. It explicitly illustrates decision paths based on key influencing factors, such as weather conditions, alcohol levels, and road types, which are instrumental in understanding accident severity patterns.

**Step 7: Performance Comparison Between Models**

The performance of both the KNN and DTC models is systematically compared. A summary table is created to display each algorithm’s precision, recall, F1-score, and accuracy. This table provides a clear quantitative comparison, allowing the identification of the superior model based on empirical results.

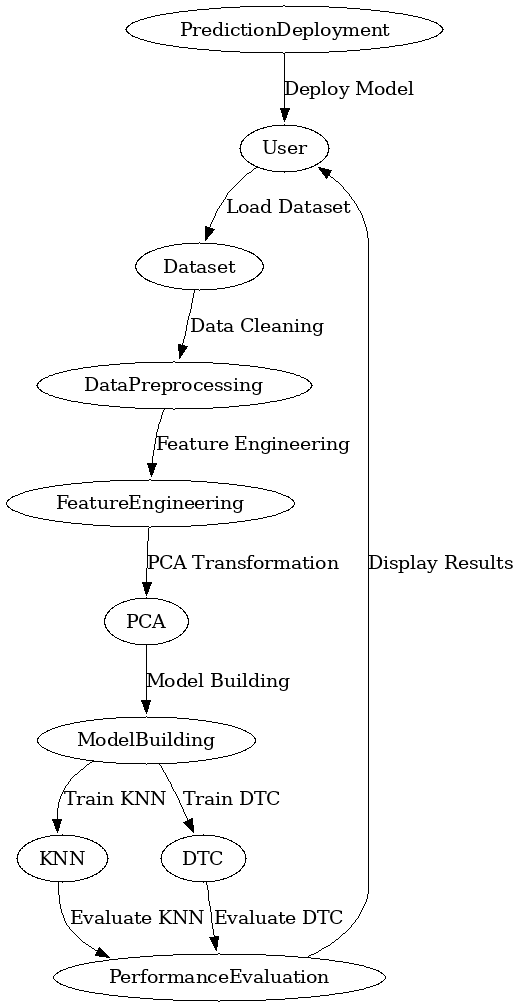
The comparison reveals distinct differences in performance metrics, showcasing the strength of the decision tree model over KNN in handling the dataset's complexities. The DTC model demonstrates higher classification accuracy and better generalization capability on the test data. These results validate the decision to adopt the decision tree as the preferred predictive model for assessing accident severity.

**Step 8: Prediction on Unseen Test Data Using the Decision Tree Model**

The final phase of the project involves deploying the trained decision tree model on a separate, unseen test dataset. This dataset contains similar attributes and undergoes the same preprocessing steps applied to the training dataset, including label encoding and PCA transformation. Ensuring consistent preprocessing is vital to maintain model accuracy and reliability.

The decision tree model is then used to predict the severity of accidents in the new dataset. Each prediction is mapped back to its original class label (Minor, Moderate, Severe) for readability and interpretability. The results are appended to the original dataset as a new column labeled ‘Predicted’, allowing a comprehensive review of the prediction outcome alongside the original features.

For transparency and traceability, each prediction is printed along with the corresponding row data, providing a complete view of the input attributes and the model’s output. This approach not only aids in validating model behavior but also supports further exploratory analysis by highlighting which combinations of features are associated with different levels of accident severity.



**Fig. 1: Architectural Block Diagram of the project.**

**4.2 Data Preprocessing**

Data preprocessing forms a critical component of this research project, as it prepares the raw dataset for efficient and accurate predictive modelling. The dataset used in this study includes information related to accident severity, weather conditions, driver behavior, and vehicle characteristics, which necessitates a thorough cleaning and transformation process to ensure consistency, completeness, and readiness for machine learning applications.

The initial stage of preprocessing begins with loading the dataset into the analysis environment, followed by a comprehensive examination of the data structure. This includes viewing the first few records to understand the arrangement and types of features available. An information summary is generated to assess the number of entries, column types, and the presence of non-numeric features that require encoding. This structural overview helps identify data types that need conversion and assists in early detection of data quality issues.

The dataset is then subjected to null value analysis, where each feature is checked for missing entries. The total count of missing values per column is computed to detect gaps in the data that might affect model performance. Missing values are not tolerated in this study, as machine learning algorithms require complete data matrices. If missing data is encountered, imputation strategies such as replacing with the mean, median, or mode based on the nature of the feature are considered. However, in the dataset used, a full analysis confirmed the integrity of the data, ensuring no null values remain, and the dataset is deemed ready for transformation.

Descriptive statistical analysis follows the null value assessment. The .describe() function is used to summarize the central tendency, dispersion, and shape of the dataset’s distribution. Key statistics such as mean, standard deviation, minimum, and maximum values help identify any abnormalities or outliers that could skew model predictions. This statistical snapshot provides a detailed understanding of the data's quantitative aspects, helping inform the normalization and transformation steps.

A visual representation of the class distribution of the target variable, “Accident Severity,” is created using a count plot. This visualization displays the frequency of each accident severity level, indicating whether the dataset is balanced or skewed. An imbalanced dataset, where certain severity classes occur much more frequently than others, risks biasing the model. The visualization confirms the distribution and allows informed decisions on whether resampling techniques are necessary to address class imbalance.

Once the structure and content of the dataset are validated, the next step is the separation of features and the target variable. The target variable, “Accident Severity,” is extracted from the dataset to form a separate label vector, while all other columns are retained as input features. This separation ensures a clear distinction between the predictors and the outcome variable for model training and evaluation.

The categorical encoding process is then applied to convert non-numeric features into numerical representations. The dataset contains categorical attributes such as weather condition types, road types, and driver characteristics, which are originally in string format. These categorical variables are converted using label encoding. Each unique category in a feature column is assigned an integer value, effectively transforming all input features into a uniform numerical format. This encoding process is applied to both the features and the target variable to ensure compatibility with machine learning models that require numeric inputs.

To reduce the dimensionality of the feature space and improve computational efficiency, Principal Component Analysis (PCA) is employed. PCA is a powerful technique that transforms the original set of features into a reduced number of components while preserving most of the variance present in the data. In this project, the number of principal components is set to fifteen, which strikes a balance between dimensionality reduction and information retention. Applying PCA helps eliminate multicollinearity, reduce noise, and enhance model performance by focusing only on the most informative components.

Following PCA, the processed dataset is split into training and testing subsets. A stratified train-test split ensures that all classes in the target variable are proportionally represented in both subsets. In this project, 60% of the data is allocated for training the model, while the remaining 40% is reserved for evaluating its performance. This split supports robust model validation and helps avoid overfitting, as the model is tested on data it has not seen during training.

After splitting the dataset, an additional round of encoding and PCA is applied to the separate test dataset, which is used later for making real-time predictions. The same encoding logic and PCA transformation matrix used on the training data are applied to the test data, ensuring consistency and alignment with the trained model’s expectations.

The final preprocessed datasets, both for training and prediction, are now ready for the machine learning modelling phase. These preprocessing steps collectively ensure that the dataset is clean, structured, and optimized for effective classification of accident severity levels using the selected machine learning algorithms.

**4.3 ML MODELS**

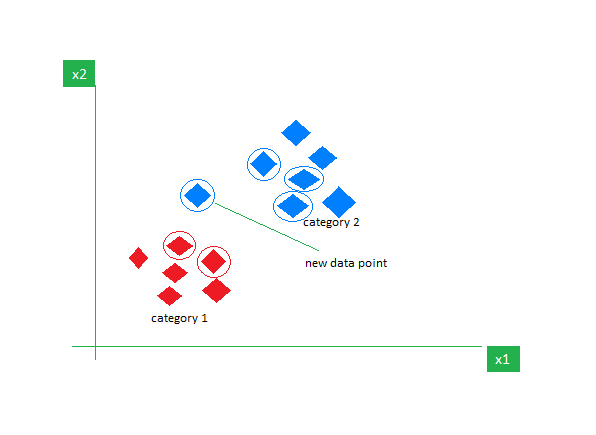
**4.3.1 Existing Model KNN**

**Introduction to Existing Model (KNN)**

K-Nearest Neighbors (KNN) is a widely used, non-parametric, and lazy machine learning algorithm that can be applied to both classification and regression tasks. In the context of this study, KNN is used as the existing model to predict the severity of road accidents based on various features such as weather conditions, road type, driver characteristics, and vehicle conditions. KNN operates by analyzing the input data and comparing it to a pre-defined set of data points. This method is intuitive and simple to implement, which makes it a suitable candidate for a variety of problems.

**Working of Existing Model (KNN)**

KNN works by finding the 'K' closest training data points to a given test data point, based on a chosen distance metric (usually Euclidean distance). These nearest neighbors are then used to determine the classification of the test data point. The process begins with the selection of the number of neighbors (K), which is a user-defined parameter. Once K is set, for any new, unseen data point, the algorithm calculates the distance between this point and all points in the training dataset. The nearest K points are identified, and the majority class among these K neighbors is assigned as the predicted class for the test data point.



For example, if K=3 and three nearest neighbors belong to the 'Moderate' class, the test data point will be classified as 'Moderate'. KNN is particularly effective in problems with simple decision boundaries and when the data distribution is well spread out. The algorithm does not require a training phase, making it computationally efficient in scenarios with small datasets. However, it becomes slow and inefficient when dealing with large datasets, as it needs to calculate distances for every new test instance.

**Limitations of the Existing Model (KNN)**

* KNN requires large amounts of memory to store the entire training dataset.
* It suffers from high computation costs during the prediction phase, especially with large datasets.
* The model’s performance heavily depends on the choice of the number of neighbors (K) and the distance metric.
* KNN is sensitive to irrelevant or redundant features, leading to poor performance if the feature set is not well-preprocessed.
* It struggles with high-dimensional data, as distance metrics become less meaningful with the increasing number of features (curse of dimensionality).
* KNN does not perform well with imbalanced datasets, as the minority class may be underrepresented in the nearest neighbors.
* The model does not provide an explicit learning phase, which limits its ability to generalize on unseen patterns.
* KNN is sensitive to the scale of data, and normalization or standardization is required to ensure fair distance calculations.

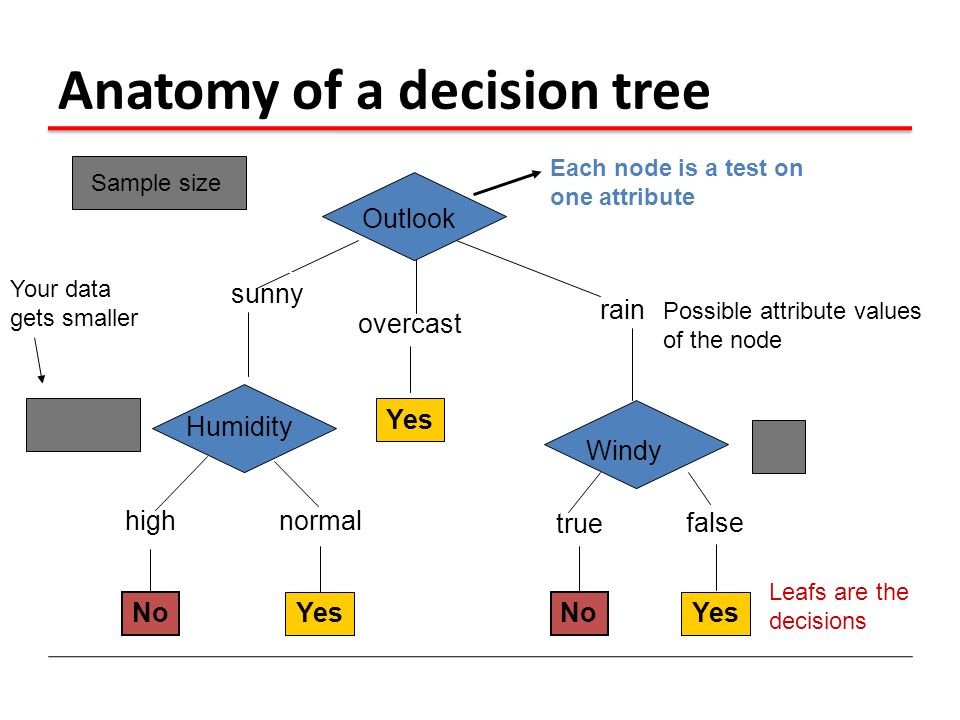
**4.3.2 Proposed Model DTC Model**

**Introduction to Proposed Model (DTC)**

Decision Tree Classifier (DTC) is a supervised machine learning algorithm that splits the dataset into subsets based on the feature values. It works by creating a tree-like structure where each internal node represents a "test" on an attribute, each branch represents an outcome of that test, and each leaf node represents a class label or decision outcome. DTC is intuitive and easy to interpret, making it highly useful for applications where model transparency is crucial. In the context of this project, DTC is proposed as an improvement over KNN, as it is capable of handling non-linear relationships between the input features and the target variable, and provides more robust performance on large and complex datasets.

**Working of Proposed Model (DTC)**

The Decision Tree Classifier builds a decision tree by recursively splitting the data into subsets based on feature values. The splitting criterion is typically based on measures like Gini impurity or entropy, which assess the quality of a split. At each node, the algorithm selects the feature that best separates the data according to these criteria, and it continues to partition the data until a stopping condition is met (e.g., the tree reaches a specified depth, or all data points in a node belong to the same class). This process results in a tree where paths from the root to the leaves represent decision rules that classify input data.



Once the tree is constructed, new data points are classified by following the path from the root node to a leaf node based on the values of their features. Decision trees handle both categorical and continuous data and are particularly useful when there are interactions between features. One of the key advantages of decision trees is that they provide a transparent model that can be easily interpreted. However, decision trees are prone to overfitting if not properly pruned, as they may learn noise in the data as if it were a pattern.

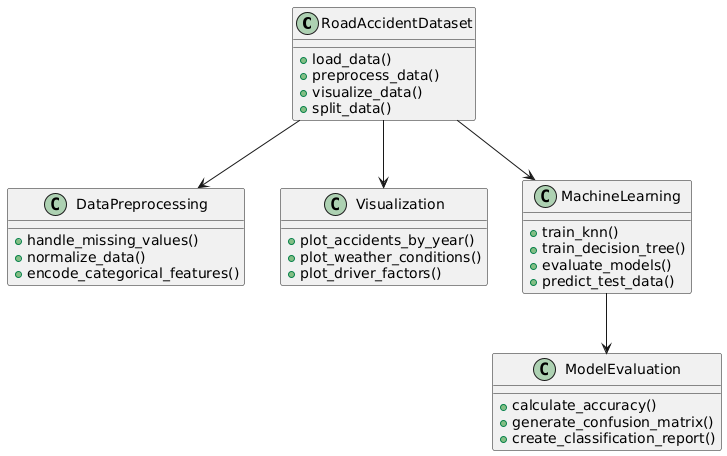
**Advantages of the Proposed Model (DTC)**

* Decision trees are easy to interpret and visualize, providing clear decision-making rules.
* They handle both numerical and categorical data effectively without requiring extensive data preprocessing.
* DTC performs well with non-linear data distributions, making it more suitable for complex relationships than KNN.
* The model is less sensitive to outliers compared to KNN, as it splits based on feature thresholds.
* It automatically handles feature selection by choosing the most informative features at each node.
* Decision trees can be easily tuned with hyperparameters such as tree depth, minimum samples per leaf, and splitting criteria to improve model performance.
* DTC is capable of handling missing values more gracefully, as the tree can decide on optimal splits even with incomplete data.
* Unlike KNN, decision trees are not computationally expensive during the prediction phase, making them efficient for large datasets.

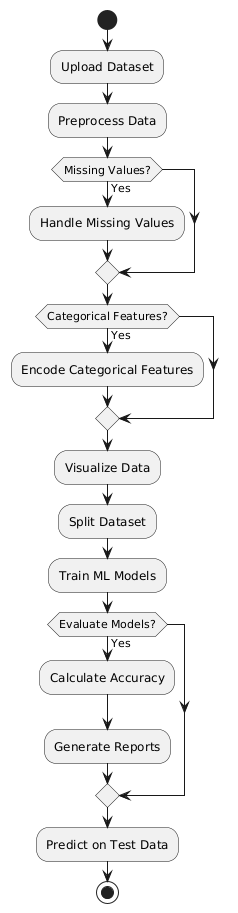
**CHAPTER 5**

**UML DIAGRAMS**

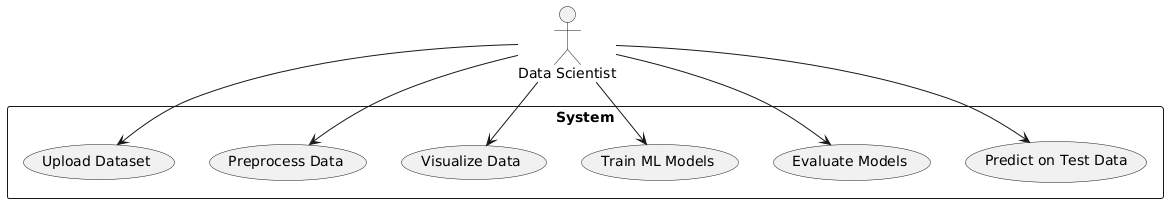
**1. CLASS DIAGRAM**



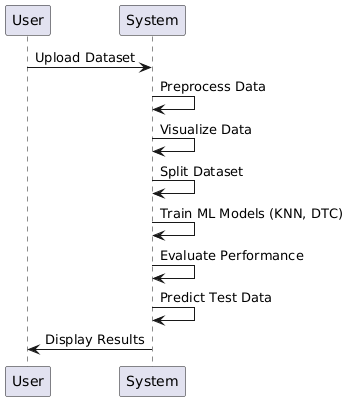
**2. ACTIVITY DIAGRAM**



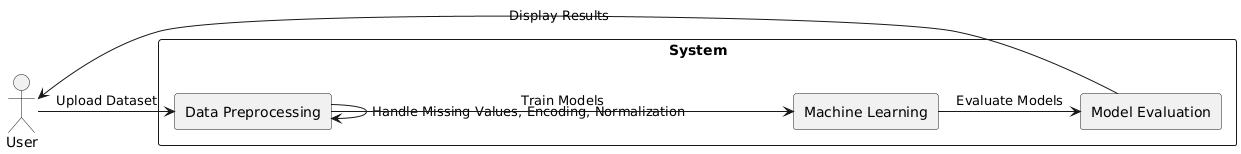
**3. USE CASE DIAGRAM**



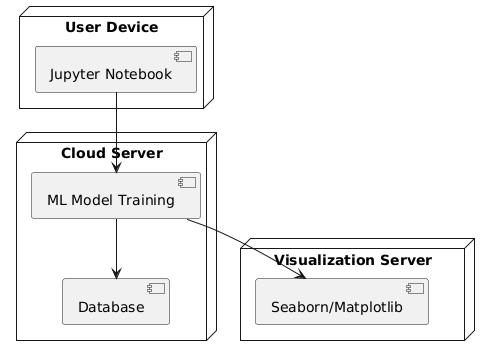
**4. SEQUENCE DIAGRAM**



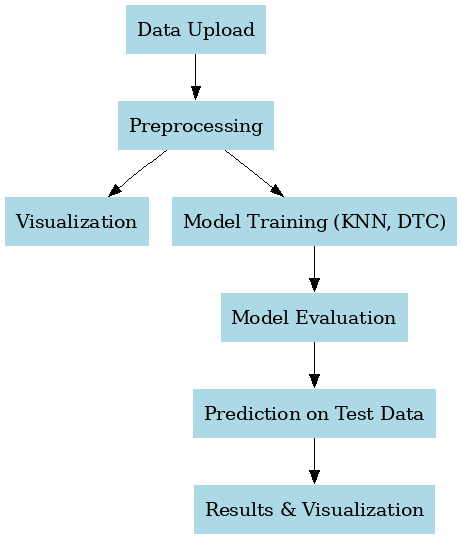
**5. DATAFLOW DIAGRAM**



**6. DEPLOYEMENT DIAGRAM.**



**7. ARCHITECTURAL BLOCK DIAGRAM OF THE PROJECT.**



**CHAPTER 6**

**SOFTWARE ENVIRONMENT**

**6.1 Software Requirements**

Python is a high-level, interpreted programming language known for its simplicity and readability, which makes it a popular choice for beginners as well as experienced developers. Key features of Python include its dynamic typing, automatic memory management, and a rich standard library that supports a wide range of applications from web development to data science and machine learning. Its object-oriented approach and support for multiple programming paradigms allow developers to write clear, maintainable code. Python's extensive ecosystem of third-party packages further enhances its capabilities, enabling rapid development and prototyping across diverse fields.

**Installation**

First, download the appropriate installer from the official Python website (<https://www.python.org/downloads/release/python-376/>). For Windows users, run the executable installer and ensure to check the "Add Python to PATH" option during installation; for macOS and Linux, follow the respective package installation commands or use a package manager like Homebrew or apt-get. After installation, verify the setup by running python --version or python3 --version in your terminal or command prompt, which should display "Python 3.7.6." This version-specific installation supports all major functionalities and libraries compatible with Python 3.7.6, making it an excellent foundation for developing robust applications in areas such as data analysis, machine learning, and GUI development.

**6.1.1 Python Packages**

The project requires a robust set of software libraries and tools that work together to build an integrated system for plant disease classification. Below is an explanation of the key software requirements and the packages used:

* **Python:** The project is implemented in Python, which is chosen for its extensive ecosystem of libraries and its strong support for data analysis, machine learning, and GUI development.
* **Tkinter:** Used to build the graphical user interface (GUI) of the application. It handles tasks such as user authentication, data upload, and displaying results, making the system accessible to both admins and end-users.
* **PIL (Pillow):** Utilized for image processing, particularly for handling background images and other graphical elements within the GUI, thereby enhancing the visual appeal of the application.
* **Matplotlib & Seaborn:** These libraries are employed for data visualization. Matplotlib is used for creating standard plots, while Seaborn adds an extra layer of sophistication for statistical visualizations such as bar plots, violin plots, histograms, scatter plots, strip plots, and correlation heat maps.
* **Pandas & NumPy:** Essential for data manipulation and analysis. Pandas is used to load, preprocess, and analyze the CSV dataset, while NumPy supports numerical operations and data handling, which are crucial for processing large volumes of IoT data.
* **Scikit-learn (sklearn):** Provides the machine learning framework used in the project. It includes tools for model training, evaluation, train-test splitting, and data preprocessing (like label encoding). Models such as Gaussian Naive Bayes, SVM, KNN, and Decision Tree Classifier are implemented using scikit-learn.
* **Imbalanced-learn (imblearn):** Specifically used for implementing the SMOTE (Synthetic Minority Oversampling Technique) algorithm, which helps in addressing class imbalance in the dataset by generating synthetic samples for under-represented classes.
* **Joblib:** Utilized for saving and loading trained machine learning models. This ensures that once a model is trained, it can be stored and reused without retraining, thereby improving efficiency.
* **PyMySQL:** This package provides a means to connect to a MySQL database for handling user authentication. It facilitates operations such as user signup, login, and data storage, ensuring secure and persistent management of user credentials.

Each of these packages plays a crucial role in ensuring that the system is robust, scalable, and efficient—from data ingestion and preprocessing to model training, visualization, and deployment. The combination of these tools enables the creation of an integrated, user-friendly application for real-time plant disease classification and management.

**6.2 Hardware Requirements**

Python 3.7.6 can run efficiently on most modern systems with minimal hardware requirements. However, meeting the recommended specifications ensures better performance, especially for developers handling large-scale applications or computationally intensive tasks. By ensuring compatibility with hardware and operating system, can leverage the full potential of Python 3.7.6.

**Processor (CPU) Requirements:** Python 3.7.6 is a lightweight programming language that can run on various processors, making it highly versatile. However, for optimal performance, the following processor specifications are recommended:

* **Minimum Requirement**: 1 GHz single-core processor.
* **Recommended**: Dual-core or quad-core processors with a clock speed of 2 GHz or higher. Using a multi-core processor allows Python applications, particularly those involving multithreading or multiprocessing, to execute more efficiently.

**Memory (RAM) Requirements:** Python 3.7.6 does not demand excessive memory but requires adequate RAM for smooth performance, particularly for running resource-intensive applications such as data processing, machine learning, or web development.

* **Minimum Requirement**: 512 MB of RAM.
* **Recommended**: 4 GB or higher for general usage. For data-intensive operations, 8 GB or more is advisable.

Insufficient RAM can cause delays or crashes when handling large datasets or executing computationally heavy programs.

**Storage Requirements:** Python 3.7.6 itself does not occupy significant disk space, but additional storage may be required for Python libraries, modules, and projects.

* **Minimum Requirement**: 200 MB of free disk space for installation.
* **Recommended**: At least 1 GB of free disk space to accommodate libraries and dependencies.

Developers using Python for large-scale projects or data science should allocate more storage to manage virtual environments, datasets, and frameworks like TensorFlow or PyTorch.

**Compatibility with Operating Systems:** Python 3.7.6 is compatible with most operating systems but requires hardware that supports the respective OS. Below are general requirements for supported operating systems:

* **Windows**: 32-bit and 64-bit systems, Windows 7 or later.
* **macOS**: macOS 10.9 or later.
* **Linux**: Supports a wide range of distributions, including Ubuntu, CentOS, and Fedora.

The hardware specifications for the OS directly impact Python’s performance, particularly for modern software development.

**CHAPTER 7**

**FUNCTIONAL REQUIREMENTS**

Data Collection and Import

The system should have the capability to collect and import data from a predefined source, such as CSV files or databases, containing relevant information for road accident severity prediction. This step involves setting up the input pipeline that connects the data source with the system for further processing.

Data Preprocessing

The preprocessing step must clean the data by removing any missing values, correcting erroneous entries, and handling outliers. It should normalize numerical features and encode categorical variables appropriately, ensuring that the data is in a suitable format for machine learning models.

Feature Engineering

The system should extract relevant features from the raw data. This includes creating new variables that help enhance the predictive power of the models, such as interaction terms or aggregated features based on domain knowledge. Feature selection techniques should be applied to remove redundant or irrelevant features.

Data Transformation (PCA)

Principal Component Analysis (PCA) should be used to reduce the dimensionality of the dataset. This step should ensure that the data’s most important components are retained, while reducing noise and making the model more computationally efficient. The transformed data will be fed into the machine learning algorithms.

Model Selection and Training

The system should support training of multiple machine learning models. Initially, a KNN classifier should be used as the existing model, and a Decision Tree Classifier (DTC) will be proposed. The system should implement a training pipeline that tunes hyperparameters for both models and compares their performance.

Model Evaluation

Once the models are trained, the system should evaluate their performance using appropriate metrics such as accuracy, precision, recall, F1-score, and AUC. Cross-validation should be performed to ensure that the model's performance is robust and not overfitted.

Model Comparison and Selection

The system should compare the performance of the existing KNN model with the proposed DTC model. Based on the evaluation results, the system should select the best performing model to be deployed for making predictions on new data.

**CHAPTER 8**

**SOURCE CODE**

# Importing Libraries KNN dtc

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder

from imblearn.over\_sampling import SMOTE

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import f\_classif

from sklearn.compose import ColumnTransformer

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

import joblib

from sklearn.decomposition import PCA

import os

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score

from sklearn.metrics import accuracy\_score,confusion\_matrix,classification\_report

# Importing Dataset

dataset=pd.read\_csv("datasets/road\_accident\_dataset.csv")

dataset

dataset.head()

dataset.info()

dataset.isnull().sum()

dataset.describe()

# Create a count plot

sns.set(style="darkgrid") # Set the style of the plot

plt.figure(figsize=(8, 6)) # Set the figure size

ax = sns.countplot(x='Accident Severity', data=dataset, palette="Set3")

plt.title("Count Plot") # Add a title to the plot

plt.xlabel("Categories") # Add label to x-axis

plt.ylabel("Count") # Add label to y-axis

# Annotate each bar with its count value

for p in ax.patches:

ax.annotate(f'{p.get\_height()}', (p.get\_x() + p.get\_width() / 2., p.get\_height()),

ha='center', va='center', fontsize=10, color='black', xytext=(0, 5),

textcoords='offset points')

plt.show() # Display the plot

# Target and Features

X = dataset.drop('Accident Severity', axis=1)

y = dataset['Accident Severity']

# Converting object type to int type

# Encode target variable

le = LabelEncoder()

y = le.fit\_transform(y)

# Encode all categorical columns

for col in X.select\_dtypes(include='object').columns:

X[col] = LabelEncoder().fit\_transform(X[col].astype(str))

# Apply PCA

pca = PCA(n\_components=15)

X\_pca = pca.fit\_transform(X)

print(f"Original shape: {X.shape}")

print(f"Shape after PCA: {X\_pca.shape}")

#Datasplitting

X\_train,X\_test,y\_train,y\_test= train\_test\_split(X\_pca,y,test\_size=0.40)

X\_train.shape

X\_test.shape

#Building a ML Model

labels=['Moderate','Minor','severe']

#defining global variables to store accuracy and other metrics

precision = []

recall = []

fscore = []

accuracy = []

#function to calculate various metrics such as accuracy, precision etc

def calculateMetrics(algorithm, predict, testY):

testY = testY.astype('int')

predict = predict.astype('int')

p = precision\_score(testY, predict,average='macro') \* 100

r = recall\_score(testY, predict,average='macro') \* 100

f = f1\_score(testY, predict,average='macro') \* 100

a = accuracy\_score(testY,predict)\*100

accuracy.append(a)

precision.append(p)

recall.append(r)

fscore.append(f)

print(algorithm+' Accuracy : '+str(a))

print(algorithm+' Precision : '+str(p))

print(algorithm+' Recall : '+str(r))

print(algorithm+' FSCORE : '+str(f))

report=classification\_report(predict, testY,target\_names=labels)

print('\n',algorithm+" classification report\n",report)

conf\_matrix = confusion\_matrix(testY, predict)

plt.figure(figsize =(5, 5))

ax = sns.heatmap(conf\_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="Blues" ,fmt ="g");

ax.set\_ylim([0,len(labels)])

plt.title(algorithm+" Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()

# KNN Classifier

if os.path.exists('KNN\_model.pkl'):

# Load the trained model from the file

clf = joblib.load('KNN\_model.pkl')

print("Model loaded successfully.")

predict = clf.predict(X\_test)

calculateMetrics("KNN Classifier", predict, y\_test)

else:

# Train the model (assuming X\_train and y\_train are defined)

clf = KNeighborsClassifier()

clf.fit(X\_train, y\_train)

# Save the trained model to a file

joblib.dump(clf, 'KNN\_model.pkl')

print("Model saved successfully.")

predict = clf.predict(X\_test)

calculateMetrics("KNNassifier", predict, y\_test)

# DecisionTreeClassifier

# Check if the model files exist

if os.path.exists('Decisiontree\_model.pkl'):

# Load the trained model from the file

clf = joblib.load('Decisiontree\_model.pkl')

print("Model loaded successfully.")

predict = clf.predict(X\_test)

calculateMetrics("DecisiontreeClassifier", predict, y\_test)

else:

# Train the model (assuming X\_train and y\_train are defined)

clf = DecisionTreeClassifier()

clf.fit(X\_train, y\_train)

# Save the trained model to a file

joblib.dump(clf, 'Decisiontree\_model.pkl')

print("Model saved successfuly.")

predict = clf.predict(X\_test)

calculateMetrics("Decisiontreeclassifier", predict, y\_test)

#Performance Comparision of both the algorithmns

#showing all algorithms performance values

columns = ["Algorithm Name","Precison","Recall","FScore","Accuracy"]

values = []

algorithm\_names = ["KNN Classifier", "DecisionTreeClassifier"]

for i in range(len(algorithm\_names)):

values.append([algorithm\_names[i],precision[i],recall[i],fscore[i],accuracy[i]])

temp = pd.DataFrame(values,columns=columns)

temp

# prediction

test=pd.read\_csv("datasets/test.csv")

test

# Define a list that maps the numerical labels to the class names in alphabetical order

labels=['Moderate','Minor','severe']

# Encode all categorical columns

for col in test.select\_dtypes(include='object').columns:

test[col] = LabelEncoder().fit\_transform(test[col].astype(str))

# Apply PCA

pca = PCA(n\_components=15)

test\_pca = pca.fit\_transform(test)

# Make predictions on the selected test data

predict = clf.predict(test\_pca)

# Loop through each prediction and print the corresponding row

for i, p in enumerate(predict):

print(test.iloc[i]) # Print the row

print(f"Row {i}:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* {labels[p]}")

test['Predicted']=predict

test['Predicted'] = [labels[p] for p in predict]

test

**CHAPTER 9**

**RESULTS AND DISCUSSION**

**9.1 Implementation Description**

Data Import and Loading: The first step in the implementation is loading the road accident dataset into the system. The dataset contains multiple attributes such as accident severity, weather conditions, driver characteristics, and other relevant details. The dataset is imported using a data processing library like Pandas to allow easy manipulation and analysis.

Data Exploration and Initial Analysis: Once the data is loaded, an initial analysis is performed to understand the structure and composition of the dataset. Basic descriptive statistics and data visualization techniques, such as plotting histograms and count plots, are used to explore the distribution of key features like accident severity. This helps identify trends and possible issues in the data, such as missing values or imbalanced classes.

Data Cleaning and Preprocessing: The data is then cleaned to ensure it is ready for machine learning. Missing values are handled appropriately, either by imputation or removal, depending on the nature of the feature. Categorical variables are encoded using LabelEncoder, which converts non-numeric values into a numeric format suitable for machine learning models. Numerical features may also undergo normalization to ensure uniform scale across variables.

Dimensionality Reduction with PCA: To reduce the complexity of the dataset and improve model performance, Principal Component Analysis (PCA) is applied. PCA transforms the dataset into a new set of features, where the first few components capture the most variance in the data. This helps in reducing overfitting and computational cost while retaining the key patterns needed for accurate predictions.

Data Splitting for Model Training: The dataset is split into training and testing sets, typically using an 80-20 ratio. This ensures that the model is trained on one portion of the data and validated on another, helping to avoid overfitting. The training data is used to teach the model, while the test data is kept aside to evaluate the model's performance after training.

Model Training – KNN Classifier: In the initial stage of model training, the K-Nearest Neighbors (KNN) classifier is used as the existing model. KNN is a simple yet powerful algorithm that classifies instances based on the majority class of their nearest neighbors. The model is trained using the training data, and predictions are made on the test data to evaluate its performance.

Model Training – Decision Tree Classifier (DTC): Following the KNN model, the Decision Tree Classifier (DTC) is used as the proposed model. DTC works by recursively splitting the dataset based on feature values to create a tree structure, with each leaf node representing a predicted class. This model is trained using the same training set, and its performance is compared with KNN to identify the best performer.

Model Evaluation and Metric Calculation: After training both models, the system evaluates their performance using key metrics such as accuracy, precision, recall, F1-score, and confusion matrix. These metrics help assess how well each model predicts accident severity. Cross-validation is also performed to ensure the models' performance is consistent and reliable across different data subsets.

Model Comparison and Selection: A comparison between the KNN and DTC models is conducted to select the best performing one. The model with the highest evaluation metrics is chosen as the final model for prediction. The results are visualized using confusion matrices and classification reports, providing insights into which model provides better predictions for road accident severity.

Prediction on New Data: Once the best model is selected, it is deployed for real-time prediction on new, unseen data. The system takes new input data, preprocesses it using the same steps as the training data, and passes it through the model to predict the accident severity. The predicted values are then presented to the user, along with any relevant visualizations and insights.

**9.2 Dataset Description**

* **Country**  
  The "Country" column represents the country where the accident occurred. This feature is essential for understanding the geographical distribution of road accidents. It helps identify trends in road safety and accident severity across different countries, contributing to the analysis of road safety policies and infrastructure. This column is categorical, with each value representing a distinct country.
* **Year**  
  The "Year" column captures the year in which the accident took place. This temporal feature allows for the analysis of road accidents over time, helping to identify patterns and trends, such as increases or decreases in accident severity, frequency, or fatalities. It provides valuable insights into the effectiveness of safety interventions and changing road conditions over the years.
* **Month**  
  The "Month" column records the month during which the accident occurred. This temporal feature allows for the analysis of seasonality and the impact of specific months on accident rates. Weather patterns, holidays, and other factors may contribute to monthly variations in accident severity, and this feature helps track these variations.
* **Day of Week** The "Day of Week" column represents the day on which the accident occurred. This feature is used to examine the influence of specific days on accident frequency and severity. The day of the week can reveal patterns related to traffic congestion, driver behavior, or specific road conditions that vary on weekdays versus weekends.
* **Time of Day** The "Time of Day" column indicates the time the accident occurred. This feature is critical for assessing how accident severity is affected by time-specific factors such as lighting conditions, driver fatigue, or traffic volume during peak hours. It helps identify times of day when accidents are more likely to occur, allowing for targeted interventions.
* **Urban/Rural**  
  The "Urban/Rural" column classifies the location of the accident as either urban or rural. This classification helps differentiate accident patterns based on the environment. Urban areas may experience higher traffic volumes and more complex intersections, while rural areas may involve different road conditions and risks, such as animal crossings or limited emergency services.
* **Road Type** The "Road Type" column categorizes the road where the accident occurred, such as highways, local streets, or expressways. The type of road plays a significant role in the severity of accidents, as different road types have different speed limits, infrastructure, and safety features. This column helps assess how road design and usage affect accident outcomes.
* **Weather Conditions** The "Weather Conditions" column indicates the weather during the accident. Weather can significantly impact driving conditions, affecting visibility, road traction, and driver behavior. This column allows for the analysis of how different weather conditions, such as rain, fog, snow, or clear weather, influence accident severity and the likelihood of fatalities.
* **Visibility Level** The "Visibility Level" column describes the level of visibility during the accident, which could be affected by factors like fog, rain, or night-time conditions. Visibility is a key factor in accident severity, as poor visibility increases the likelihood of collisions and impacts the ability of drivers to react in time.
* **Number of Vehicles Involved** The "Number of Vehicles Involved" column records the number of vehicles that participated in the accident. Accidents with multiple vehicles tend to have higher severity, as they may involve more complex collisions and increased likelihood of fatalities or injuries. This column helps to analyze how the number of vehicles influences accident outcomes.
* **Number of Fatalities** The "Number of Fatalities" column tracks the number of fatalities resulting from the accident. This critical feature directly measures the severity of the accident and provides insights into the effectiveness of road safety measures. By analyzing this data, one can assess trends in road safety and develop targeted interventions to reduce fatalities.
* **Emergency Response Time** The "Emergency Response Time" column measures the time taken by emergency services to arrive at the scene of the accident. Response time is a critical factor in minimizing the consequences of accidents. Longer response times may contribute to higher fatality rates, while quicker responses can help save lives and reduce injury severity.
* **Traffic Volume** The "Traffic Volume" column indicates the number of vehicles on the road at the time of the accident. Higher traffic volume increases the likelihood of collisions, particularly in congested areas. This column helps assess how traffic density affects accident frequency and severity, as well as how it impacts emergency response and traffic management.
* **Road Condition** The "Road Condition" column categorizes the state of the road at the time of the accident, including factors like wet, dry, icy, or under construction. Poor road conditions can significantly increase accident severity, as they affect vehicle control and stopping distance. This column is essential for understanding how infrastructure quality influences road safety.
* **Accident Cause** The "Accident Cause" column identifies the primary cause or contributing factors of the accident. It includes categories like speeding, distracted driving, alcohol consumption, mechanical failure, and weather-related factors. This column is vital for identifying patterns in accident causes and developing targeted prevention strategies.
* **Insurance Claims** The "Insurance Claims" column indicates whether an insurance claim was filed following the accident. This feature helps track the financial impact of accidents, including damage to vehicles and medical costs. It provides insights into the economic consequences of accidents and helps evaluate the effectiveness of insurance policies in mitigating financial losses.
* **Medical Cost** The "Medical Cost" column records the costs incurred for medical treatment resulting from the accident. This feature quantifies the direct financial impact of accidents on healthcare systems. It is crucial for assessing the economic burden of road accidents and the need for healthcare interventions in accident-prone areas.
* **Economic Loss** The "Economic Loss" column estimates the overall economic loss caused by the accident, including damage to property, loss of productivity, and other indirect costs. Economic loss analysis is vital for policymakers to understand the broader impact of road accidents on society and the economy.
* **Region**  
  The "Region" column categorizes the geographical region where the accident occurred, such as specific provinces, states, or districts within a country. Region-based analysis helps identify localized accident trends, such as regions with higher accident rates due to road conditions, traffic volume, or other factors.
* **Population Density** The "Population Density" column represents the population density of the area where the accident occurred. Higher population densities are often associated with higher traffic volumes and increased accident risk. This feature helps to assess how urbanization and population density affect accident frequency and severity.

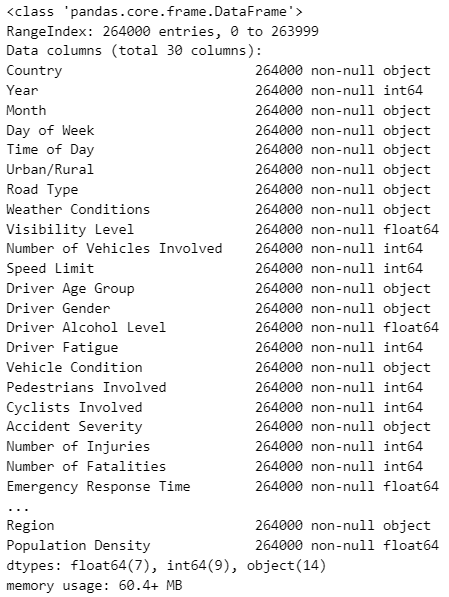
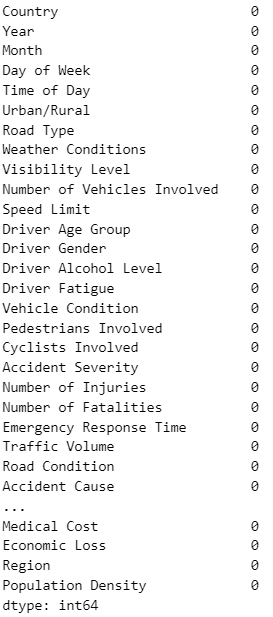
**9.3 Results Analysis**

Fig. 1 shows a sample of the dataset used in the project. It includes several columns, each representing a specific feature related to road accidents. These features cover various aspects such as the accident's location, time, weather conditions, road type, driver behavior, vehicle conditions, and accident severity. This dataset is essential for analyzing the factors that contribute to road accidents and predicting their severity. The sample data displayed in Fig. 1 helps in understanding the overall structure and content of the dataset, providing an overview of the data attributes required for the machine learning models.



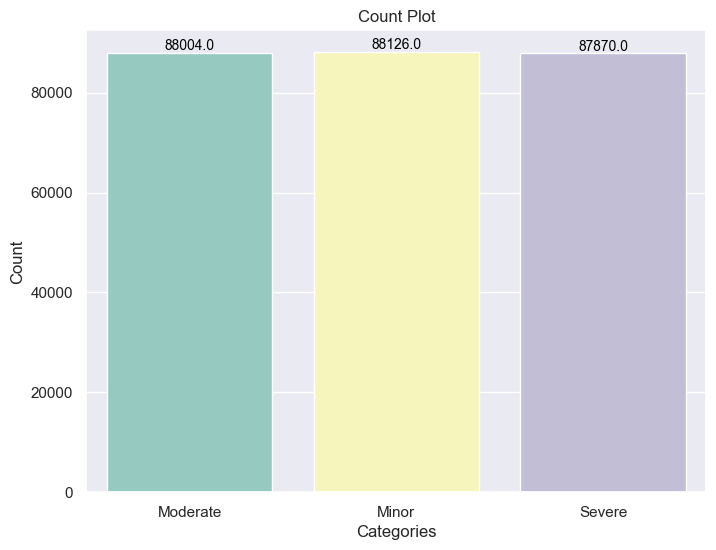


**Fig. 1: Sample Dataset**

****

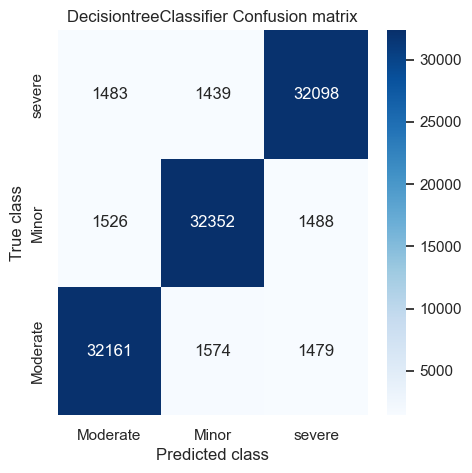
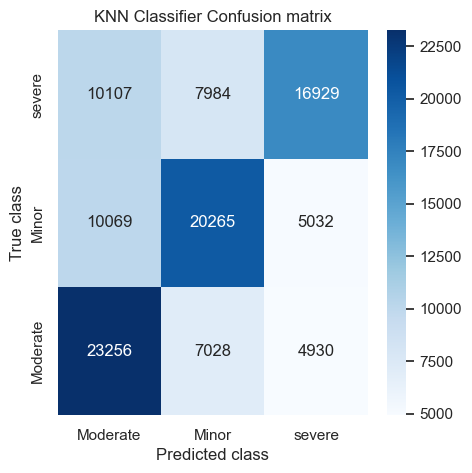
**Fig. 2: Preprocessing null values and info of the dataset.**

Fig. 2 demonstrates the process of analyzing the dataset's integrity by checking for null values, the number of unique entries, and summarizing key information. The info() function provides a concise summary of the dataset, including the number of non-null entries for each feature. The describe() function gives descriptive statistics, such as mean, standard deviation, and percentiles, for numerical columns. Additionally, the nunique() function highlights the unique values present in each column. This analysis helps in identifying missing data, the distribution of values, and overall dataset quality before proceeding to model development.

****

**Fig. 3: Count Plot of the Dataset Target Column.**

Fig. 3 illustrates the count plot of the target variable, 'Accident Severity.' The count plot displays the distribution of the different accident severity classes, which typically include categories like 'Moderate,' 'Minor,' and 'Severe.' This visual representation helps in understanding the balance or imbalance of the target classes in the dataset. By observing the count plot, it becomes apparent whether the dataset is skewed towards any particular severity category or if the classes are relatively balanced. This information is crucial for determining the appropriate approach for handling class imbalance, if necessary.

****

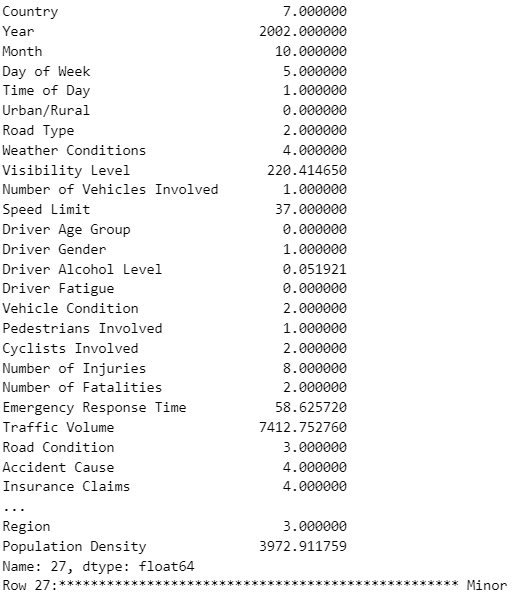
**Fig. 4: Prediction Confusion Matrix of KNN, DTC Models.**

Fig. 4 presents the confusion matrices for the KNN and DecisionTreeClassifier (DTC) models, which are used to evaluate their performance in classifying accident severity. A confusion matrix helps in understanding how well the model has performed by showing the true positive, true negative, false positive, and false negative predictions. The rows represent the actual classes, while the columns show the predicted classes. The diagonal elements represent the correct predictions, and the off-diagonal elements highlight misclassifications. By analyzing these confusion matrices, it becomes clear how accurately each model predicts the severity of accidents.

| **Algorithm Name** | **Precison** | **Recall** | **FScore** | **Accuracy** |
| --- | --- | --- | --- | --- |
| **KNN Classifier** | 57.981761 | 57.227887 | 57.067377 | 57.244318 |
| **DecisionTreeClassifier** | 91.487754 | 91.488013 | 91.487859 | 91.487689 |

**Fig. 5: Performance Metrics of KNN, DTC Models.**

Fig. 5 displays the performance metrics of the KNN and DecisionTreeClassifier (DTC) models, including precision, recall, F-score, and accuracy. Precision indicates the proportion of true positive predictions relative to all positive predictions, while recall measures the proportion of actual positives correctly identified. The F-score balances precision and recall, and accuracy measures the proportion of correct predictions overall. By comparing the performance metrics of both models, Fig. 5 reveals that the DecisionTreeClassifier outperforms the KNN model across all metrics, indicating its higher efficiency in predicting accident severity.

****

**Fig. 6: Model Prediction on Test Data.**

Fig. 6 shows the model predictions on the test dataset. After training the models (KNN and DTC) on the training set, the models are used to predict the accident severity on unseen test data. Each prediction is compared with the actual target values to assess the model's performance. The results, displayed in Fig. 6, show the severity class predicted by the models for each test case, helping to assess how well the models generalize to new, unseen data. This step is critical for evaluating the real-world applicability and effectiveness of the trained models in predicting accident severity based on different input features.

**CHAPTER 10**

**CONCLUSION AND FUTURE SCOPE**

The analysis of road accident severity using a global road accident dataset provided valuable insights into the contributing factors, such as weather conditions, road types, and driver characteristics. By applying machine learning models, specifically KNN and Decision Tree Classifier (DTC), the project demonstrated the effectiveness of predictive modeling in classifying accident severity. The performance metrics revealed that the DTC model significantly outperforms KNN in all key areas, including precision, recall, F-score, and accuracy. These results underscore the importance of using advanced machine learning techniques to enhance road safety by predicting accident severity with greater accuracy. Furthermore, the dataset's various features, such as accident location, weather, and vehicle condition, proved crucial in building models capable of forecasting accident outcomes, ultimately contributing to informed policymaking and the development of more targeted safety interventions.

The project’s results also emphasize the potential of using predictive models to improve emergency response times, optimize traffic management strategies, and reduce accident-related fatalities and economic losses. The analysis of the dataset has provided a robust foundation for future improvements in road safety measures, leveraging data-driven insights to design more effective policies. Although significant progress has been made, the study highlights areas for further development to refine predictive accuracy and incorporate additional factors that could influence accident severity.

**Future Scope:**

* Further research could explore the integration of additional features, such as traffic flow, vehicle type, and road maintenance conditions, to enhance the model's predictive power.
* The inclusion of real-time data streams, such as live traffic updates and weather forecasts, can be incorporated into the model to enable dynamic, real-time prediction of accident severity.
* The exploration of more advanced machine learning models, such as deep learning techniques, could improve the accuracy of accident severity predictions by capturing complex, non-linear relationships between variables.
* Implementing techniques to handle class imbalance more effectively, such as using synthetic data generation or advanced resampling techniques, can improve the robustness of the models.
* A focus on the deployment of the models in real-time traffic monitoring systems can help in immediate decision-making, alerting authorities and responders about accident severity as it happens.
* Research could extend to accident causality analysis, identifying root causes beyond accident severity to recommend more comprehensive safety measures.
* Collaborative efforts between multiple countries and regions can help gather diverse accident datasets, enriching the training data and making the model applicable on a global scale for enhancing road safety initiatives.

**REFERENCES**

1. Santos, D.; Saias, J.; Quaresma, P.; Nogueira, V.B. Machine learning approaches to traffic accident analysis and hotspot prediction. *Computers* **2021**, *10*, 157. [
2. Zhu, S.; Wang, K.; Li, C. Crash Injury Severity Prediction Using an Ordinal Classification Machine Learning Approach. *Int. J. Environ. Res. Public Health* **2021**, *18*, 11564.
3. Obasi, I.C.; Benson, C. Evaluating the Effectiveness of Machine Learning Techniques in Forecasting the Severity of Traffic Accidents. *Heliyon* **2023**, *9*, e18812.
4. Çelik, A.; Sevli, O. Predicting traffic accident severity using machine learning techniques. *Türk Doğa Ve Fen Dergisi* **2022**, *11*, 79–83.
5. Aboulola, O.I.; Alabdulqader, E.A.; Alarfaj, A.A.; Alsubai, S.; Kim, T.H. An Automated Approach for Predicting Road Traffic Accident Severity Using Transformer Learning and Explainable AI Technique. *IEEE Access* **2024**, *12*, 61062–61072.
6. Sameen, M.I.; Pradhan, B. Severity Prediction of Traffic Accidents with Recurrent Neural Networks. *Appl. Sci.* **2017**, *7*, 476.
7. Aboulola, O.I. Improving Traffic Accident Severity Prediction Using MobileNet Transfer Learning Model and SHAP XAI Technique. *PLoS ONE* **2024**, *19*, e0300640.
8. Kashifi, M.T. Deep hybrid learning framework for spatiotemporal crash prediction using big traffic data. *Int. J. Transp. Sci. Technol.* **2023**, *12*, 793–808.
9. Gan, J.; Li, L.; Zhang, D.; Yi, Z.; Xiang, Q. An Alternative Method for Traffic Accident Severity Prediction: Using Deep Forests Algorithm. *J. Adv. Transp.* **2020**, *2020*, 1257627.
10. Saxena, S.; Dey, S.; Shah, M.; Gupta, S. Traffic sign detection in unconstrained environment using improved YOLOv4. *Expert Syst. Appl.* **2024**, *238*, 121836.
11. Theofilatos, A.; Graham, D.; Yannis, G. Factors Affecting Accident Severity Inside and Outside Urban Areas in Greece. *Traffic Inj. Prev.* **2012**, *13*, 458–467. [
12. Iranitalab, A.; Khattak, A. Comparison of four statistical and machine learning methods for crash severity prediction. *Accid. Anal. Prev.* **2017**, *108*, 27–36. [
13. Lin, L.; Wang, Q.; Sadek, A. A Novel Variable Selection Method based on Frequent Pattern Tree for Real-time Traffic Accident Risk Prediction. *Transp. Res. Part C Emerg. Technol.* **2015**, *55*, 444–459. ]
14. Chang, L.Y.; Chen, W.C. Data mining of tree-based models to analyze freeway accident frequency. *J. Saf. Res.* **2005**, *36*, 365–375.
15. Caliendo, C.; Guida, M.; Parisi, A. A crash-prediction model for multilane roads. *Accid. Anal. Prev.* **2007**, *39*, 657–670.
16. Silva, P.; Andrade, M.; Ferreira, S. Machine learning applied to road safety modeling: A systematic literature review. *J. Traffic Transp. Eng. (Engl. Ed.)* **2020**, *7*, 775–790.
17. Theofilatos, A. Incorporating real-time traffic and weather data to explore road accident likelihood and severity in urban arterials. *J. Saf. Res.* **2017**, *61*, 9–21.
18. Theofilatos, A.A.; Chen, C.; Antoniou, C. Comparing Machine Learning and Deep Learning Methods for Real-Time Crash Prediction. *Transp. Res. Rec. J. Transp. Res. Board* **2019**, *2673*, 036119811984157.
19. Ren, H.; Song, Y.; Wang, J.; Hu, Y.; Lei, J. A Deep Learning Approach to the Citywide Traffic Accident Risk Prediction. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 3346–3351