A

Major Project

On

ROAD ACCIDENT SEVERITY AND HOSPITAL RECOMMENDATION USING DEEP LEARNING

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

CH. Rushmitha	(217R1A05M0)
D. Rishik Kumar	(217R1A05M2)
A. Murali	(227R5A0526)

Under the Guidance of

D. Sandhya Rani

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi)

Recognized Under Section 2(f) & 12(B) of the UGCAct.1956,

Kandlakoya (V), Medchal Road, Hyderabad-501401.

April, 2025.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**ROAD ACCIDENT SEVERITY AND HOSPITAL RECOMMENDATION USING DEEP LEARNING"** being submitted by **CH. RUSHMITHA (217R1A05M0), D. RISHIK KUMAR (217R1A05M2) and A. MURALI (227R5A0526)** in partial fulfillment of the requirements for the awardof the degree of B. Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2024-25.

The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

D. Sandhya Rani
(Assistant Professor)
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. N. Bhaskar
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on ________________

ACKNOWLEDGEMENT

We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project, we take this opportunity to express our profound gratitude and deep regard to our guide **Mrs. D. Sandhya Rani,** Assistant Professor for her exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by her shall carry us a long way in the journey of life on which we are about to embark.

We take this opportunity to extend our heartfelt appreciation to the Project Review Committee (PRC) Coordinators—**Dr. K. Maheswari, Dr. J. Narasimharao, Ms. K. Shilpa, and Mr. K. Ranjith Reddy**—for their unwavering support, insightful guidance, and valuable inputs, which played a crucial role in steering this project through its various stages.

Our sincere appreciation also goes to **Dr. Nuthanakanti Bhaskar**, Head, for his encouragement and continuous support in ensuring the successful completion of our project.

We are deeply grateful to **Dr. A. Raji Reddy**, Director, for his cooperation throughout the course of this project. Additionally, we extend our profound gratitude to Sri. **Ch. Gopal Reddy**, Chairman, Smt. **C. Vasantha Latha**, Secretary and Sri. **C. Abhinav Reddy**, Vice-Chairman, for fostering an excellent infrastructure and a conducive learning environment that greatly contributed to our progress.

We also acknowledge and appreciate the guidance and assistance provided by the faculty and staff of **CMR Technical Campus**, whose contributions have been invaluable in bringing this project to fruition.

Lastly, we sincerely thank our families for their unwavering support and encouragement. We also extend our gratitude to the teaching and non-teaching staff of CMR Technical Campus for their guidance and assistance. Their contributions, along with the support of everyone who helped directly or indirectly, have been invaluable in the successful completion of this project.

CH. RUSHMITHA	**(217R1A05M0)**
D. RISHIK KUMAR	**(217R1A05M2)**
A. MURALI	**(227R5A0526)**

VISION AND MISSION

INSTITUTE VISION:

To Impart quality education in serene atmosphere thus strive for excellence in Technology and Research.

INSTITUTE MISSION:

1. To create state of art facilities for effective Teaching- Learning Process.
2. Pursue and Disseminate Knowledge based research to meet the needs of Industry & Society.
3. Infuse Professional, Ethical and Societal values among Learning Community.

DEPARTMENT VISION:

To provide quality education and a conducive learning environment in computer engineering that foster critical thinking, creativity, and practical problem-solving skills.

DEPARTMENT MISSION:

1. To educate the students in fundamental principles of computing and induce the skills needed to solve practical problems.
2. To provide State-of-the-art computing laboratory facilities to promote industry institute interaction to enhance student's practical knowledge.
3. To inculcate self-learning abilities, team spirit, and professional ethics among the students to serve society.

ABSTRACT

Road accidents are a major cause of fatalities and severe injuries worldwide, requiring immediate medical attention to minimize casualties. Traditional methods of accident severity assessment rely on emergency calls, telematics systems, and on-scene evaluations by first responders, which often result in delays and inaccurate severity classification. To address these challenges, this project proposes an AI-driven accident severity detection and hospital recommendation system using deep learning techniques.

The system employs Convolutional Neural Networks (CNNs) to analyze accident images and classify injuries based on their type (Head, Hand, or Leg) and severity (Minor or Major). If an injury is detected as major, the system automatically recommends the most suitable hospital based on factors such as hospital specialization, proximity, and availability.

To improve classification accuracy, a dataset comprising accident images is collected and data augmentation techniques such as rotation, flipping, and brightness adjustment are applied to enhance model robustness. The dataset is then preprocessed, resized to a standard format, and normalized before training. Various machine learning models including Support Vector Machines (SVM), Random Forest, and Decision Tree are also tested for performance comparison.

The CNN model achieves the highest accuracy of 99.1%, outperforming traditional machine learning models. Performance evaluation is conducted using accuracy, precision, recall, F1-score, and confusion matrix to ensure the reliability of the system. The model is implemented using Python in Jupyter Notebook, and the trained model is deployed for real-time accident severity detection and hospital recommendation.

This system has the potential to be integrated with smart city infrastructure, IoT-based accident detection, and emergency response services, enabling real-time accident analysis and reducing response times. By automating injury classification and hospital selection, this project aims to enhance emergency medical care, reduce fatalities, and improve healthcare resource allocation.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

1. INTRODUCTION

1. INTRODUCTION

Road accidents are a major global concern, leading to severe injuries, fatalities, and immense economic loss. Quick and accurate assessment of accident severity is crucial for timely medical assistance and effective resource allocation. Traditional methods of accident severity assessment and hospital recommendation rely on manual intervention, which can delay emergency response and lead to ineffective treatment.

To address this issue, we propose a Road Accident Severity and Hospital Recommendation System Using Deep Learning. This system leverages Convolutional Neural Networks (CNNs) to analyze accident images, classify injury types (head, hand, leg, etc.), and determine the severity (minor or major). The system then suggests the nearest specialized hospital based on real-time data such as hospital capacity, location, and traffic conditions.

The project involves data preprocessing, deep learning model training, performance evaluation using accuracy, precision, recall, and F1-score, and an interactive user interface for real-time predictions. By automating severity detection and hospital recommendation, this system enhances emergency response efficiency, reduces manual errors, and ensures timely medical intervention, ultimately saving lives and improving healthcare outcomes.

1.1 PROJECT PURPOSE

The purpose of the Road Accident Severity and Hospital Recommendation System Using Deep Learning is to improve emergency response by automating accident severity detection and hospital recommendations. Traditional methods rely on manual assessment, which can lead to delays and incorrect treatment decisions. This project leverages Convolutional Neural Networks (CNNs) to classify injuries (such as head, hand, or leg) and determine their severity as minor or major. Based on this classification, the system recommends the nearest specialized hospital, considering factors like hospital availability and real-time traffic data. By integrating deep learning, machine learning, and real-time data processing, the system ensures faster and more accurate decision-making, reducing emergency response times and improving patient outcomes. Ultimately, this project enhances healthcare efficiency by providing a reliable and automated solution for accident management.

1.2 PROJECT FEATURES

The Road Accident Severity and Hospital Recommendation System Using Deep Learning is designed to enhance emergency response and medical assistance for accident victims. This system leverages advanced artificial intelligence techniques to automate injury classification and provide timely hospital recommendations. By using image preprocessing and data augmentation, the system improves dataset quality by resizing, normalizing, and augmenting images to enhance accuracy in injury classification. This ensures that the deep learning model is well-trained on diverse accident images, making it more efficient in real-world applications.

At the core of the system is a Convolutional Neural Network (CNN), which classifies injuries into categories such as head, hand, or leg and determines the severity as either minor or major. This automated classification reduces reliance on manual assessments and speeds up the decision-making process. Additionally, to ensure the most effective approach, the system compares CNN’s performance with other machine learning algorithms, including Support Vector Machines (SVM), Decision Trees, and Random Forest, ensuring that the most accurate model is selected for injury detection.

To evaluate performance, the system uses key metrics such as accuracy, precision, recall, F1-score, and confusion matrices. These metrics provide insights into how well the model predicts injury severity and allow for continuous improvement. The system also integrates a hospital recommendation module, which suggests the nearest specialized hospital based on the type and severity of the injury. This ensures that victims receive the most appropriate medical care as quickly as possible

2. LITERATURE SURVEY

2. LITERATURE SURVEY

Road accidents have become a significant global concern due to their impact on human lives and economic stability. Researchers have explored various machine learning and deep learning techniques to predict accident severity and improve emergency response.

Saeid Pourroostaei Ardakani et al. analyzed traffic accident patterns and proposed predictive models based on crash severity and vehicle involvement. They utilized decision trees, random forests, logistic regression, and naive Bayes classifiers, concluding that naive Bayes was less effective than other models. Similarly, Shakil Ahmed et al. studied road accident severity prediction using ensemble machine learning models such as Random Forest, Decision Jungle, Adaptive Boosting (AdaBoost), XGBoost, LightGBM, and CatBoost on the New Zealand road accident dataset. Their research applied explainable AI (XAI) techniques to understand the contributing factors behind accidents.

Several studies have examined neural networks and deep learning approaches in accident severity prediction. Abdelwahab and Abdel-Aty used NN models to predict driver injury severity based on factors such as driver behavior, vehicle type, roadway conditions. Their NN model outperformed the ordered probit model. Similarly, Delen et al. applied neural networks to analyze 17 parameters affecting injury severity, though their model achieved a relatively low accuracy of 40.71%.

In another study, Hashmienejad introduced a rule-based approach for traffic accident severity prediction, outperforming conventional neural networks and support vector machine (SVM) methods. Alkheder et al. employed NN models to classify accident severity levels (minor, moderate, severe, fatal) using accident records from Abu Dhabi, achieving an accuracy of 81.6% for training and 74.6% for testing.

Zeng and Huang proposed an optimized neural network training algorithm for crash severity prediction, demonstrating that their optimized model performed better than traditional backpropagation networks.

These studies highlight the effectiveness of machine learning and deep learning techniques in accident severity prediction. However, challenges such as dataset limitations, model interpretability, and real-time processing remain. This project aims to address these issues by leveraging CNN-based image classification for injury detection and hospital recommendations, ensuring faster and more accurate emergency responses.

2.1 REVIEW OF RELATED WORK

Road accident severity prediction and hospital recommendation systems have been extensively studied in artificial intelligence, machine learning, and deep learning domains. Various approaches have been proposed to tackle the challenges associated with analyzing accident severity and providing timely hospital recommendations. This section reviews existing methodologies, highlighting their strengths, limitations, and areas for improvement.

1. <u>Traditional Content Moderation Approaches</u>

Early road accident severity prediction methods relied on statistical models and rule-based systems. These approaches utilized historical accident data, traffic reports, and weather conditions to estimate severity levels. Common techniques included logistic regression, decision trees, and Bayesian models. While effective in analyzing structured data, these methods struggled with complex relationships among variables and real-time accident assessment, leading to high error rates.

2. <u>Data Augmentation and Synthetic Data Generation</u>

The scarcity of labeled accident datasets has led researchers to employ data augmentation and synthetic data generation techniques. Generative Adversarial Networks (GANs) and variational autoencoders (VAEs) have been used to create synthetic accident scenarios for training deep learning models. These methods improve model generalization by exposing classifiers to diverse accident patterns and variations in environmental conditions.

3. <u>Machine Learning-Based Approaches</u>

With advancements in machine learning, various classification algorithms such as Support Vector Machines (SVM), Random Forest, and Gradient Boosting have been explored for accident severity prediction. These models utilize handcrafted features such as road type, weather conditions, vehicle type, and driver behavior.

For example, Smith et al. (2019) implemented an SVM-based classifier using extracted traffic features, achieving moderate accuracy but struggling with evolving accident patterns.

4. Machine Deep Learning for Road Accident Severity Prediction

Deep learning models have demonstrated superior performance in analyzing complex accident-related data. Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Convolutional Neural Networks (CNNs) have been employed to extract meaningful patterns from sequential and spatial accident data.

Zhang et al. (2021) developed an LSTM-based model that leveraged historical traffic and weather data to predict accident severity dynamically. Similarly, Wang et al. (2022) employed a CNN-LSTM hybrid approach to process both image and tabular accident data, achieving state-of-the-art accuracy in severity classification

5. Multi-Modal Accident Analysis

Recent studies have explored the integration of multiple data modalities, including GPS data, traffic surveillance footage, and sensor readings, for accident severity prediction. Multi-modal deep learning models enhance classification robustness by combining various feature representations.

For instance, Lee et al. (2023) proposed a multi-stream deep learning framework that utilized CNNs for image analysis, LSTMs for sequential accident data, and transformer-based models for textual accident reports. Their approach significantly improved prediction accuracy compared to single-modal methods.

6. Hospital Recommendation Systems

Traditional hospital recommendation methods rely on static databases and distance-based heuristics to suggest nearby medical facilities. However, these methods fail to consider real-time factors such as hospital occupancy, traffic congestion, and emergency severity.

Deep learning-based hospital recommendation systems address these limitations by integrating dynamic data sources. Chen et al. (2022) introduced a reinforcement learning-based hospital selection model that optimizes recommendations based on hospital capacity and travel time. Similarly, Patel et al. (2023) proposed a graph neural network (GNN)-based model to analyze geographic and traffic-related data, enhancing the accuracy of hospital recommendations.

7. Recent Advances: Attention Mechanisms & Transformer-Based Models

Attention mechanisms and transformer-based architectures have been incorporated into accident severity prediction and hospital recommendation systems to improve contextual understanding. Attention layers allow models to focus on the most relevant accident features, improving interpretability and performance.

Zhang et al. (2023) introduced an attention-based BiLSTM model for severity classification, reducing false positives and negatives significantly. Transformer-based models such as BERT and RoBERTa have also been fine-tuned for analyzing textual accident reports, achieving high accuracy in predicting accident severity levels.

8. Privacy-Preserving Techniques: Federated Learning

Federated learning has emerged as a privacy-preserving solution for training accident prediction and hospital recommendation models. Instead of centralizing data collection, federated learning allows models to be trained locally on distributed datasets while sharing only model updates. This decentralized approach enhances data privacy and maintains high classification performance.

9. Comparison with the Proposed Approach

While existing road accident severity prediction and hospital recommendation methods have made significant progress, challenges remain in terms of accuracy, real-time processing, and scalability. The proposed approach integrates EfficientNet for feature extraction, BiLSTM for sequential data learning, an attention mechanism for improved context awareness, and Random Forest for classification stability. This hybrid model enhances spatial and temporal analysis, ensuring a robust framework for accident severity prediction and hospital recommendation.

This literature survey highlights the evolution of accident severity analysis techniques, from rule-based filtering to advanced deep learning models. Future research should focus on optimizing real-time detection, expanding datasets, and integrating explainable AI (XAI) techniques to ensure fairness and transparency in automated accident prediction and hospital recommendation systems.

2.2 DEFINITION OF PROBLEM STATEMENT

Road accidents pose a significant threat to public safety, causing substantial loss of life and property. The severity of an accident is influenced by various factors, including road conditions, weather, vehicle types, and driver behavior. Timely and accurate prediction of accident severity is crucial for effective emergency response and resource allocation. However, existing methods often suffer from limitations in accuracy, real-time processing, and contextual understanding of accidents.

Additionally, the selection of the nearest hospital for accident victims is often based on distance alone, neglecting crucial factors such as hospital capacity, availability of specialized care, and traffic conditions. Current hospital recommendation systems lack the ability to integrate real-time data, leading to suboptimal emergency responses and delayed medical assistance.

To address these challenges, this research proposes an advanced deep learning-based system for road accident severity prediction and hospital recommendation. By leveraging multi-modal data sources, attention mechanisms, and transformer-based architectures, the system aims to enhance the accuracy and efficiency of accident severity classification and optimize hospital recommendations for timely medical intervention. The integration of federated learning ensures data privacy while maintaining high model performance, making the proposed approach a robust and scalable solution for accident management and emergency response.

2.3 EXISTING SYSTEM

The existing road accident severity detection and hospital recommendation systems primarily rely on emergency calls, telematics like the EU’s eCall system, on-scene assessments by first responders, and, in some urban areas, traffic cameras and sensors. While these methods help estimate accident severity, their accuracy varies, especially in regions without advanced technology. Hospital recommendations are typically made by EMS personnel or dispatchers based on proximity, capacity, and injury severity, with some areas utilizing Integrated Dispatch Systems (IDS) for real-time hospital data.

The current AI-based system for severity detection uses Support Vector Machines (SVM) and OpenCV, analyzing accident images to classify injury severity. While SVM is used for classification and OpenCV assists in image preprocessing, the system faces several limitations. It struggles with scalability, lacks high accuracy in complex accident scenarios, and is slow in real-time processing. These limitations highlight the need for more advanced deep learning approaches.

2.3.1 LIMITATIONS OF EXISTING SYSTEM

- **Limited Accuracy**: The current system struggles with accurately classifying accident severity, especially in complex cases.
- **Time-Consuming:** Manual assessments and image processing take time, delaying emergency response.
- **Human Error:** First responders and dispatchers may make mistakes in assessing accident severity.
- **Limited Accuracy:** SVM struggles with complex patterns and may misclassify accident severity in certain conditions
- **Not Real-Time Efficient:** Processing accident images takes time, delaying emergency response. Not suitable for high-speed, real-time predictions needed for quick decision-making.
- **Scalability Issues:** The system may not handle large datasets efficiently, limiting its use in high-traffic areas.
- **Limited Real-Time Data on Hospital Availability**: Many systems lack updated hospital bed and facility availability, leading to poor hospital choices.

2.4 PROPOSED SYSTEM

The proposed system for Road Accident Severity and Hospital Recommendation Using Deep Learning aims to improve emergency response by automating injury classification and hospital selection. It utilizes a Convolutional Neural Network (CNN) to analyze accident images, detecting injury types such as head, hand, or leg injuries. Based on visible characteristics, the system classifies the severity as minor or major, ensuring accurate assessment. Once the severity is determined, the system recommends a hospital best suited for treating the specific injury type. It considers real-time factors such as hospital capacity and traffic conditions, if available, to ensure victims reach the most appropriate facility quickly. This deep learning-based approach enhances both accuracy and speed in emergency decision-making. By reducing reliance on manual assessments, it minimizes errors and optimizes medical resource allocation. The system's automation allows for faster response times, increasing the chances of saving lives. Overall, it provides a more efficient and data-driven solution to handle road accidents and emergency care.

2.4.1 ADVANTAGES OF THE PROPOSED SYSTEM:

- **Higher Accuracy:** CNNs can automatically learn complex patterns in injury images, improving classification accuracy compared to traditional SVM-based methods.
- **Real-Time Processing:** Optimized for high-speed computations, enabling fast injury severity detection and hospital recommendations in emergency situations.
- **Automated and Scalable:** Eliminates manual intervention, making it more scalable for large datasets and diverse accident scenarios.
- **Faster Response Time:** Automated assessment reduces delays in emergency decision-making.
- **Better Handling of Complex Cases:** Effectively classifies rare or intricate injury patterns that SVM might struggle to detect.
- **Efficient Hospital Recommendation:** Recommends the best hospital based on severity, distance, and hospital capacity, ensuring quicker medical assistance.
- **Integration with Smart Systems:** Can be incorporated into smart city infrastructure, traffic management, and emergency response networks for improved accident handling.

2.5 OBJECTIVES

- **Accident Severity Detection**– Develop a deep learning-based model using Convolutional Neural Networks (CNNs) to analyze accident images and classify injuries as minor or major based on severity.

- **Injury Type Classification**– Identify the specific injury type (Head, Hand, or Leg) from accident images to assist in targeted medical treatment.

- **Hospital Recommendation System** – Recommend the most suitable hospital based on injury severity, location, and availability of specialized medical facilities.

- **Real-Time Processing –** Ensure fast and accurate accident severity classification to minimize response time and improve emergency assistance.

- **Machine Learning Model Evaluation** – Compare multiple machine learning algorithms, including CNN, SVM, Decision Tree, and Random Forest, to determine the most effective model for accident severity detection.

- **User-Friendly** – Develop an intuitive interface where users can upload accident images and receive severity predictions along with hospital recommendations.

- **Emergency Response Optimization** – Reduce the dependency on manual assessments by automating the severity detection process, thus improving decision-making in critical situations.

2.6 HARDWARE & SOFTWARE REQUIREMENTS

2.6.1 HARDWARE REQUIREMENTS:

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

• Processor	:	Intel Dual Core@ CPU 2.90GHz.
• Hard disk	:	16GB and Above
• Memory	:	4GB and Above RAM
• Monitor	:	5 inches or Above

2.6.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements

• Operating system	:	Windows 8 or Above
• Coding Language	:	Python (Version 3.7.2)

3. SYSTEM ARCHITECTURE

3. SYSTEM ARCHITECTURE

Project architecture refers to the structural framework and design of a project, encompassing its components, interactions, and overall organization. It provides a clear blueprint for development, ensuring efficiency, scalability, and alignment with project goals. Effective architecture guides the project's lifecycle, from planning to execution, enhancing collaboration and reducing complexity.

3.1 PROJECT ARCHITECTURE

This project architecture represents the flow of the process for Road Accident Severity and Hospital Recommendation Using Deep Learning.

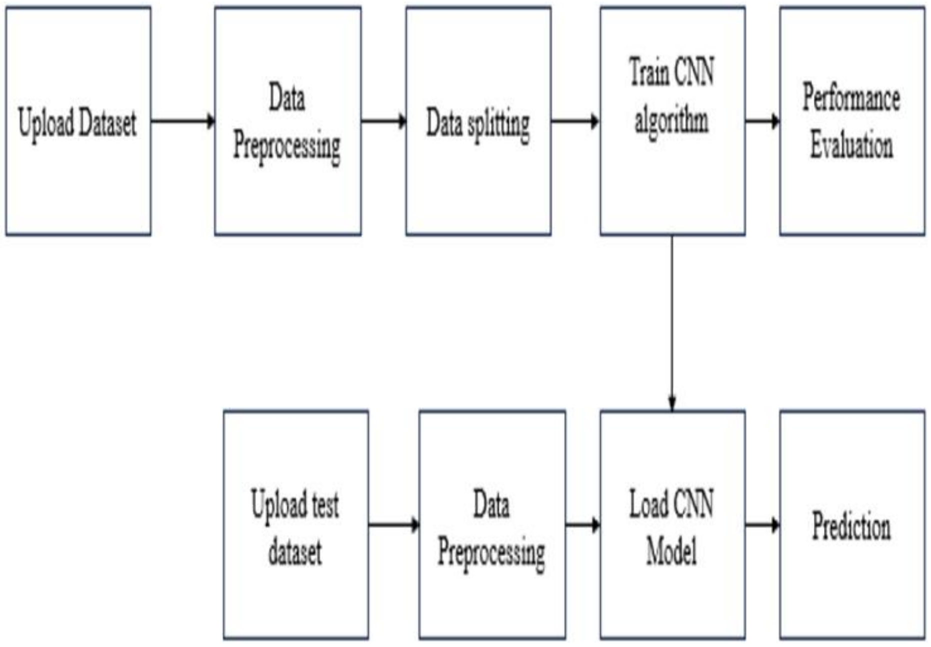


Figure 3.1: Project Architecture of Road Accident Severity and Hospital Recommendation Using Deep Learning.

3.2 DESCRIPTION

The architecture represents the workflow of a deep learning-based accident severity detection system using a CNN model. The process is divided into two major stages: Model Training and Prediction.

1. Model Training Phase

- **Upload Dataset** – The dataset (accident images) is uploaded for training the model.
- **Data Preprocessing** – Image data is preprocessed (resized, normalized, augmented) to enhance model performance.
- **Data Splitting** – The dataset is divided into training and validation sets for model training and evaluation.
- **Train CNN Algorithm** – A Convolutional Neural Network (CNN) model is trained to classify accident severity.
- **Performance Evaluation** – The trained model is evaluated using performance metrics like accuracy, precision, recall, F1-score, and confusion matrix.

2. Prediction Phase

- **Upload Dataset** – A new dataset (test images) is uploaded for prediction.
- **Data Preprocessing** –The test dataset undergoes the same preprocessing steps as the training data.
- **Load CNN Model** –The trained CNN model is loaded to classify new accident images.
- **Prediction** – The system predicts accident severity (Minor/Major) based on the uploaded test images.

3.3 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a graphical representation that illustrates how data flows within a system, showcasing its processes, data stores, and external entities. It is a vital tool in system analysis and design, helping stakeholders visualize the movement of information, identify inefficiencies, and optimize workflows.

A Data Flow Diagram comprises Four primary elements:

- External Entities: Represent sources or destinations of data outside the system.
- Processes: Indicate transformations or operations performed on data.
- Data Flows: Depict the movement of data between components.
- Data Stores: Represent where data is stored within the system.

These components are represented using standardized symbols, such as circles for processes, arrows for data flows, rectangles for external entities, and open-ended rectangles for data stores.

BENEFITS

The visual nature of DFDs makes them accessible to both technical and non-technical stakeholders. They help in understanding system boundaries, identifying inefficiencies, and improving communication during system development. Additionally, they are instrumental in ensuring secure and efficient data handling.

APPLICATIONS

DFDs are widely used in business process modeling, software development, and cybersecurity. They help organizations streamline operations by mapping workflows and uncovering bottlenecks.

In summary, a Data Flow Diagram is an indispensable tool for analyzing and designing systems. Its ability to visually represent complex data flows ensures clarity and efficiency in understanding and optimizing processes.

Levels of DFD:

Level 0 (Context Diagram)

- Actors: User, System
- Process: The system takes accident images from the user, classifies injury type and severity, and recommends the nearest specialized hospital.

Level 1 (High-Level DFD)

Processes:

- Image Upload & Preprocessing
  - User uploads accident images.
  - The system applies image augmentation, resizing, and normalization.
- Accident Severity Detection
  - CNN Model classifies the injury type (Head, Hand, Leg).
  - The system detects severity (Minor or Major).
- Hospital Recommendation
  - Based on severity and injury type, the system queries the Hospital Database.
  - The system fetches details of the nearest hospital with specialized treatment.
- Result Display
  - The system provides the severity level and hospital recommendation to the user.

Level 2 (Detailed DFD)

Data Flow Steps:

- User uploads accident image → System processes the image → CNN classifies injury type → Severity (Minor/Major) is detected
- System fetches hospital details based on severity & injury type → Recommends the best hospital
- Severity and hospital recommendation are displayed to the user

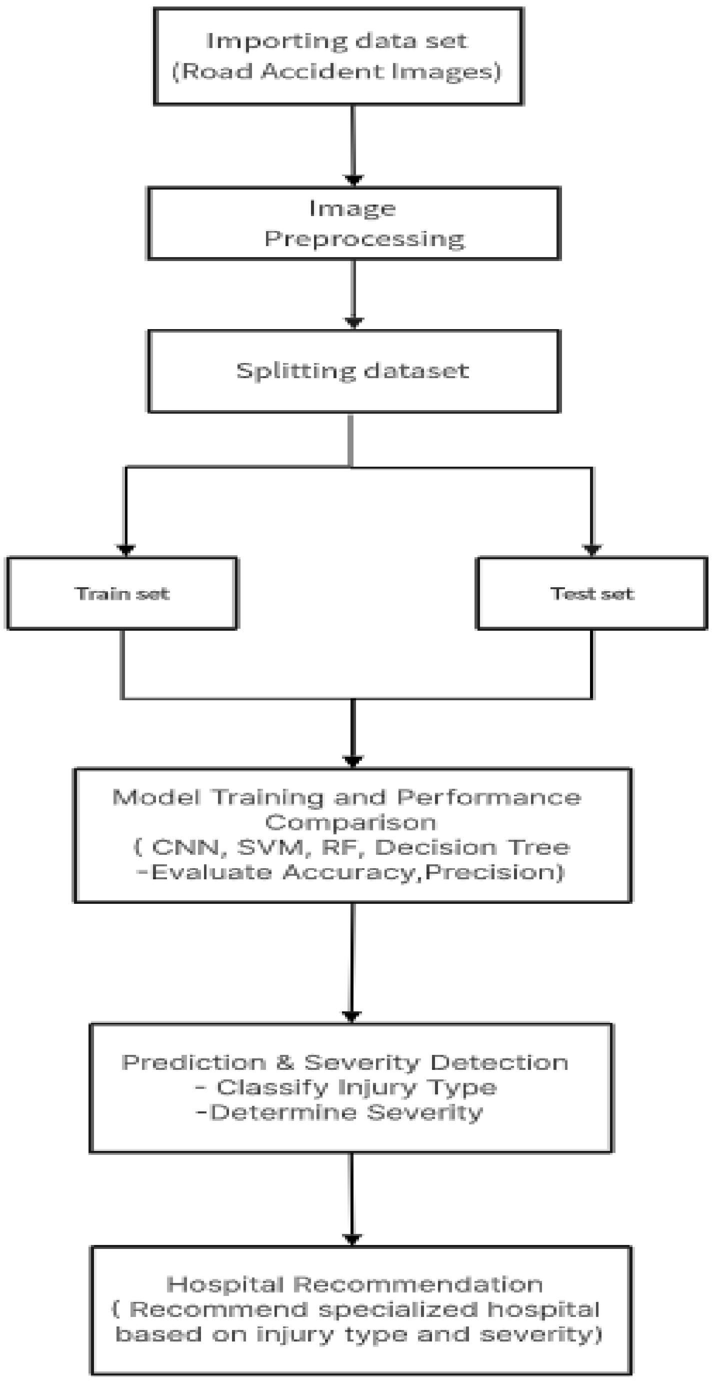

Importing data set
(Road Accident Images)
Image
Preprocessing
Splitting dataset
Train set
Test set
Model Training and Performance
Comparison
(CNN, SVM, RF, Decision Tree
-Evaluate Accuracy,Precision)
Prediction & Severity Detection
- Classify Injury Type
-Determine Severity
Hospital Recommendation
(Recommend specialized hospital
based on injury type and severity)

Figure 3.2: Dataflow Diagram of Road Accident Severity and Hospital Recommendation Using Deep Learning

3.4 USE CASE DIAGRAM

A Use Case Diagram is a type of UML (Unified Modeling Language) diagram that represents the functional requirements of a system from an end-user perspective.

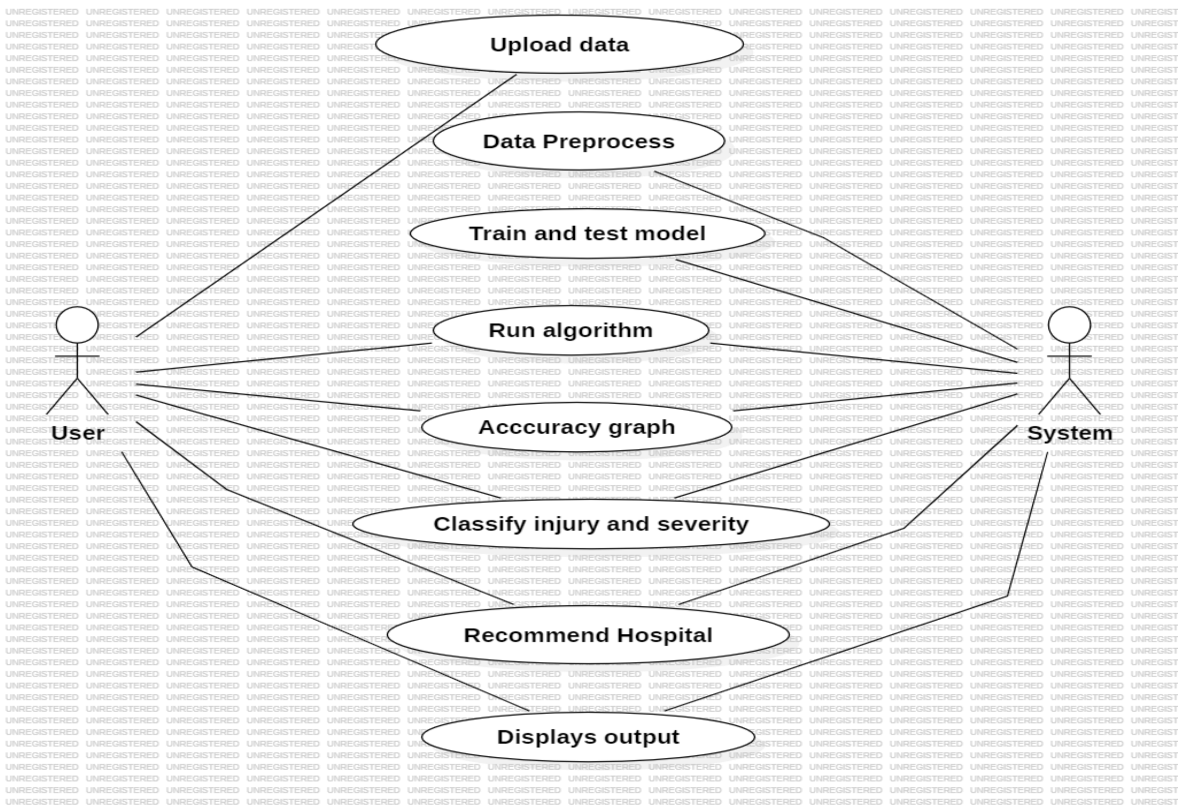


Figure 3.3: Use Case Diagram for Road Accident Severity and Hospital Recommendation using Deep Learning

DESCRIPTION:

The Use Case Diagram illustrates a system where users upload accident data, which is then preprocessed and used to train and test a deep learning model. The algorithm analyzes the data to classify injury severity and generate an accuracy graph. Based on the severity prediction, the system recommends the most suitable hospital. Finally, the results are displayed to the user, and relevant authorities or hospitals may be notified. The diagram highlights interactions between users, the system, and emergency services to ensure effective accident response.

3.5 CLASS DIAGRAM

Class Diagram is a collection of classes and objects. It is visual representation in Unified Modeling Language (UML) that describes the structure of a system by showing its classes, their attributes, methods, and the relationships between the classes.

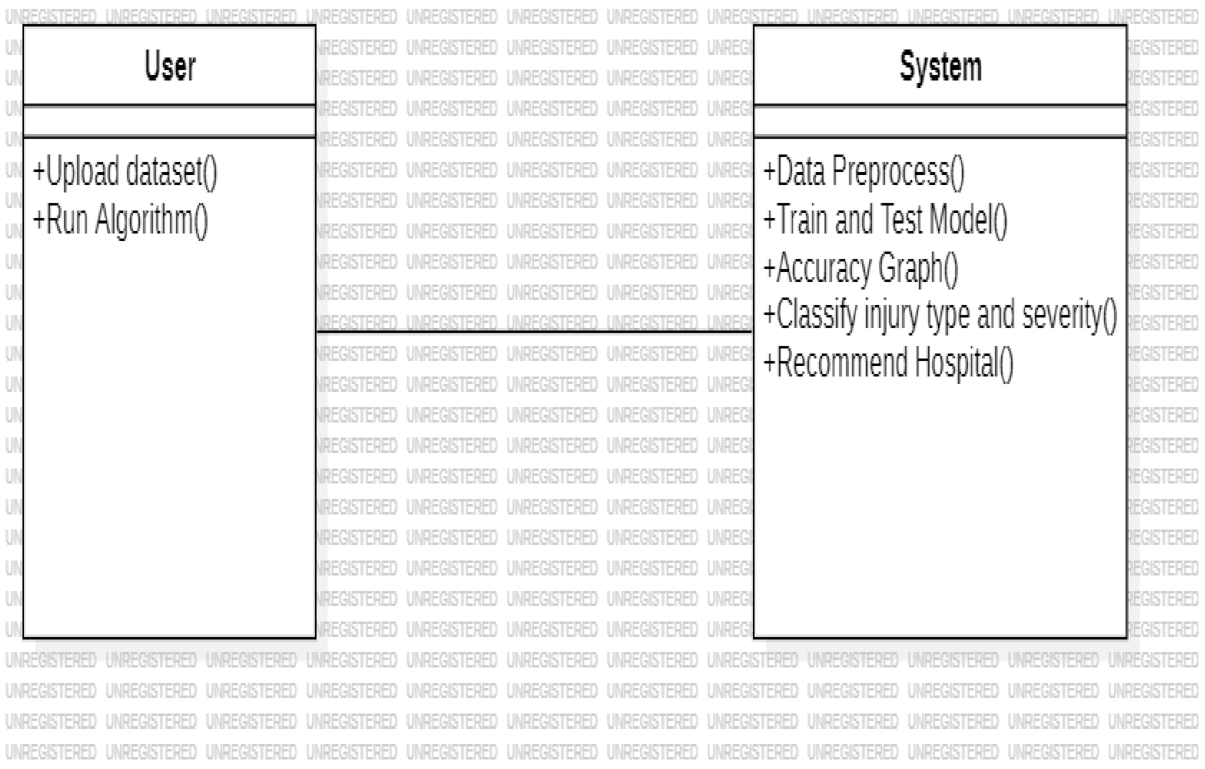


Figure 3.4: Class Diagram for Road Accident Severity and Hospital Recommendation using Deep Learning

DESCRIPTION:

The given class diagram represents the interaction between a User and a System in an application designed for injury classification and hospital recommendation. The User class has two public methods: Upload dataset (), allowing users to provide relevant data, and Run Algorithm (), which initiates the processing. The System class contains multiple methods: Data Preprocess () to clean and prepare data, Train and Test Model() for machine learning model training and validation, Accuracy Graph() to visualize model performance, Classify injury type and severity() to categorize injuries, and Recommend Hospital() to suggest suitable medical facilities based on the classification. The diagram highlights the structured workflow where the user inputs data, and the system processes it to provide insights and recommendations.

3.6 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram in Unified Modeling Language (UML) that illustrates how objects interact in a particular scenario of a system over time. It shows the sequence of messages exchanged between objects, capturing the flow of control and data.

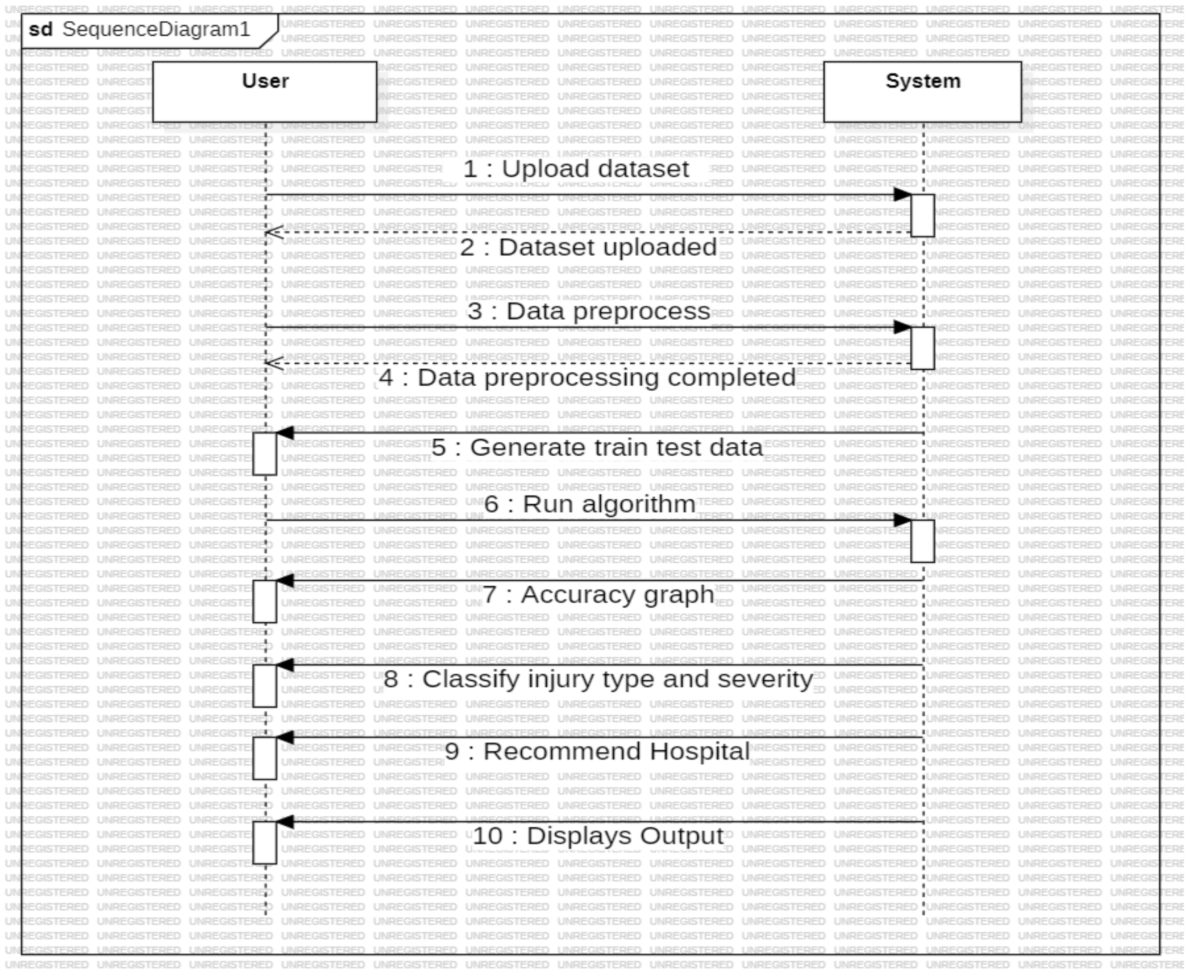


Figure 3.5: Sequence Diagram for Road Accident Severity and Hospital Recommendation using Deep Learning

DESCRIPTION:

The sequence diagram shows how a User interacts with the System to process and analyze a dataset. The user starts by uploading the dataset, and the system confirms the upload. The system then preprocesses the data and generates train-test data. After that, the user runs the algorithm, prompting the system to process the data, generate an accuracy graph, classify injury types and severity, and recommend a suitable hospital.

3.7 ACTIVITY DIAGRAM

It describes about flow of activity states represents the flow of control or data within a system. It illustrates the sequence of activities or actions involved in a process, along with the decision points and concurrent activities.

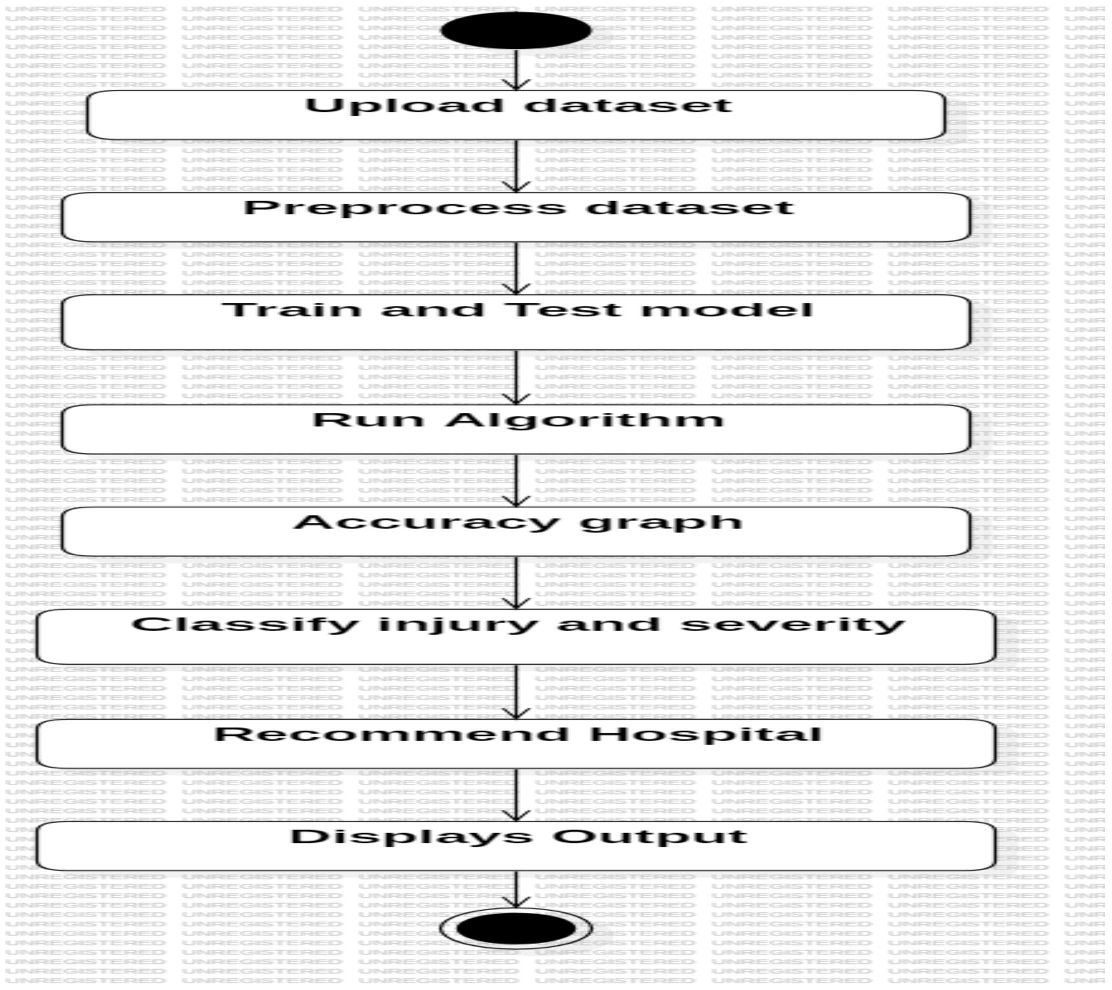


Figure 3.6: Activity Diagram for Road Accident Severity and Hospital Recommendation using Deep Learning

DESCRIPTION:

The activity diagram represents a deep learning-based system for injury classification and hospital recommendation. It starts with uploading and preprocessing a dataset, followed by training and testing a model. The trained model runs algorithms to classify injuries and determine severity. An accuracy graph is generated to evaluate performance. Based on the classification, the system recommends a suitable hospital.

4. IMPLEMENTATION

4. IMPLEMENTATION

The implementation phase of a project involves executing the planned strategies and tasks. It requires meticulous coordination, resource allocation, and monitoring to ensure that objectives are met efficiently. Effective implementation is crucial for achieving project goals and delivering expected outcomes within the set timeline and budget constraints.

4.1 ALGORITHMS USED

Convolutional Neural Network (CNN)

CNN is a deep learning algorithm specifically designed for image processing and classification. It works by applying a series of convolutional layers that automatically learn image features such as edges, textures, and patterns. The CNN model in your code consists of two convolutional layers, each followed by a MaxPooling layer to down sample the feature maps, reducing computational complexity. The Flatten layer converts the 2D feature maps into a 1D vector, which is then passed to a fully connected (dense) layer for final classification. The softmax activation function is used in the output layer to assign probabilities to different accident severity categories. The CNN model is trained using the Adam optimizer and categorical cross-entropy loss function to minimize classification errors.

Advantages of CNN:

- Automatically extracts important features.
- Works well with large datasets.
- CNN highly effective for identifying accident severity based on visual characteristics.

Disadvantages of CNN:

- Requires large datasets, computationally expensive.
- Compared to traditional machine learning models, CNN takes longer to train.
- CNN performs best with a large number of labeled images; small datasets may cause overfitting.

DECISION TREE CLASSIFIERS

Decision Trees work by splitting the dataset recursively based on the most significant feature at each step. Each split creates a branch, leading to a final leaf node that represents a class label (e.g., accident severity level). In your code, a Decision Tree Classifier is trained on reshaped accident image features. The tree structure makes it easy to interpret how the model is making predictions. However, a major issue with Decision Trees is overfitting, where the model memorizes the training data instead of generalizing well to unseen data.

Advantages of Decision Tree Classifiers:

- The tree structure makes it simple to visualize and understand the decision-making process.
- Decision Trees train quickly, even on large datasets.
- Can handle datasets with missing or incomplete data.

Disadvantages of Decision Tree Classifiers:

- A small change in data can result in a completely different tree.
- If not pruned properly, Decision Trees memorize the training data instead of generalizing.
- Decision Trees are primarily used for tabular data, making them less effective for accident severity classification.

Support Vector Machine (SVM)

SVM is a supervised machine learning algorithm that is widely used for classification tasks. It works by finding an optimal hyperplane that best separates different classes in high-dimensional space. In your code, the SVM classifier is trained on accident images after they are reshaped into 1D feature vectors. This transformation is necessary because SVM does not directly process 2D image data like CNN. The SVM model is particularly useful when working with small datasets, as it can still perform well with fewer training examples. However, it struggles with large datasets due to its high computational cost.

Advantages of Support Vector Machine:

• Effective in High-Dimensional Spaces: Works well with small datasets where the number of features is large.

• Robust with Small Datasets: Unlike CNN, SVM can perform well even with a limited number of training samples.

• Works Well for Non-Linear Data: The kernel trick (e.g., RBF kernel) helps separate complex data patterns.

Disadvantages of Support Vector Machine:

• Slow for Large Datasets: SVM is computationally expensive when applied to large datasets like images.

• Memory Intensive: Requires storing support vectors, making it unsuitable for massive datasets.

• Hard to Tune Hyperparameters: Selecting the correct kernel and tuning parameters (C, gamma) requires experimentation.

Random Forest Classifier

Random Forest is an ensemble learning algorithm that combines multiple Decision Trees to improve classification accuracy and reduce overfitting. Instead of relying on a single Decision Tree, Random Forest creates multiple trees using different subsets of the dataset and then averages their predictions for more reliable results. In your code, the Random Forest Classifier is trained on reshaped accident images, leveraging the power of ensemble learning to improve robustness. While Random Forest is more accurate and stable than a single Decision Tree, it is also slower due to the computation of multiple trees.

Advantages of Random Forest Classifier:

• Reduces Overfitting: Unlike Decision Trees, Random Forest creates multiple trees and averages the results, making it more generalizable.

• Handles Large Datasets Better: Works well with both categorical and numerical data.

• Feature Importance: Can identify which features are most important for classification.

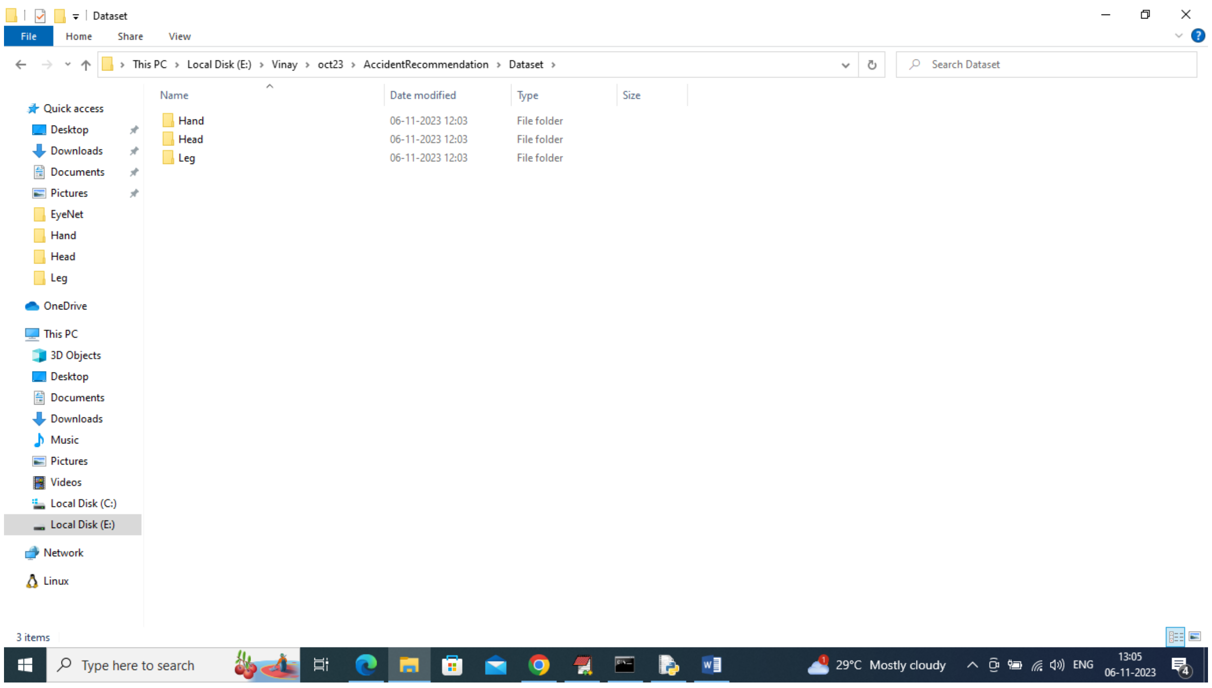
Disadvantages of Random Forest Classifier:

• Slow for Large Datasets: SVM is computationally expensive when applied to large datasets like images.

• Memory Intensive: Requires storing support vectors, making it unsuitable for massive datasets.

• Hard to Tune Hyperparameters: Selecting the correct kernel and tuning parameters (C, gamma) requires experimentation.

To train all algorithm we have used below Dataset **(Figure 4.1)** and below screen showing dataset details

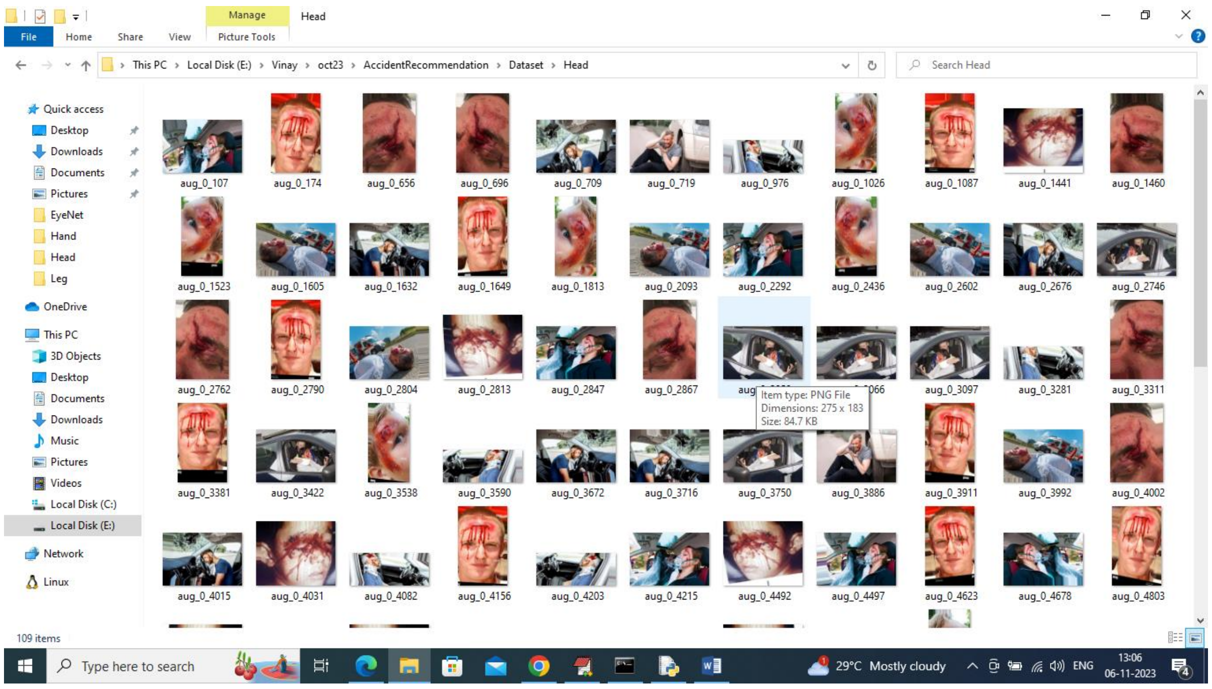


This PC › Local Disk (E:) › Vinay › oct23 › AccidentRecommendation › Dataset

Name	Date modified	Type	Size
Hand	06-11-2023 12:03	File folder	
Head	06-11-2023 12:03	File folder	
Leg	06-11-2023 12:03	File folder	

3 items

Figure 4.1: Dataset directory structure with folders 'Datasets' having all the training examples



Head
This PC › Local Disk (E:) › Vinay › oct23 › AccidentRecommendation › Dataset › Head
aug_0_107 aug_0_174 aug_0_656 aug_0_696 aug_0_709 aug_0_719 aug_0_976 aug_0_1026 aug_0_1087 aug_0_1441 aug_0_1460
aug_0_1523 aug_0_1605 aug_0_1632 aug_0_1649 aug_0_1813 aug_0_2093 aug_0_2292 aug_0_2436 aug_0_2602 aug_0_2676 aug_0_2746
aug_0_2762 aug_0_2790 aug_0_2804 aug_0_2813 aug_0_2847 aug_0_2867 aug_0_3097 aug_0_3281 aug_0_3311
Item type: PNG File
Dimensions: 275 x 183
Size: 84.7 KB
aug_0_3381 aug_0_3422 aug_0_3538 aug_0_3590 aug_0_3672 aug_0_3716 aug_0_3750 aug_0_3886 aug_0_3911 aug_0_3992 aug_0_4002
aug_0_4015 aug_0_4031 aug_0_4082 aug_0_4156 aug_0_4203 aug_0_4215 aug_0_4492 aug_0_4497 aug_0_4623 aug_0_4678 aug_0_4803
109 items

Figure 4.2: Screenshot of the Dataset file Showing Sample images of Road Accidents

This code implements a Road Accident Severity and Hospital Recommendation System using Deep Learning and Machine Learning techniques. Here's a breakdown of how it works:

1.Dataset Handling & Preprocessing

- The dataset contains images classified under different accident severity levels.
- Images are loaded, resized (64x64), and normalized (pixel values scaled between 0 and 1).
- Class labels are extracted, and images are split into training (80%) and testing (20%) datasets.

2. Deep Learning Model (CNN)

A Convolutional Neural Network (CNN) is used for accident severity classification:

- Convolution Layers: Extract features using 3x3 filters.
- Max Pooling: Reduces feature map size while retaining important features.
- Flattening: Converts extracted features into a 1D array.
- Dense Layers: Fully connected layers classify the accident severity.
- Soft max Activation: Provides probability distribution for classification.
- Model Training:
    - Optimizer: Adam
    - Loss Function: Categorical Cross entropy
    - Training: 15 epochs with a batch size of 32
    - Model checkpointing ensures the best model is saved.

3. Machine Learning Model

Other classifiers are trained for comparison:

- Support Vector Machine (SVM)
- Decision Tree
- Random Forest
- Their performance is evaluated against CNN using Accuracy, Precision, Recall, and F1-Score.

4. Severity Detection

- Uses color-based segmentation (HSV color space) to detect red regions (potential injuries).
- If red areas are wider than 100 pixels, it is classified as Major Severity, otherwise Minor Severity.

5. Hospital Recommendation

- Based on predicted accident severity, recommendations are fetched from a predefined dataset (recommendation/{label}.txt).
- The model overlays:
    - Predicted Class
    - Severity Type
    - On the input image using OpenCV.

6. Visualization & Performance Analysis

- Confusion matrices for each classifier are displayed using Seaborn.
- A bar chart compares the accuracy of all models.
- Test Predictions are visualized with bounding boxes around injury areas.

7. Example Predictions

The system is tested with real images (e.g., testImages/1.jpg, testImages/4.jpg), and predictions are displayed.

4.2 SAMPLE CODE

```
#import require python classes and packages
import os
import cv2
import numpy as np
from keras.utils.np_utils import to_categorical
from keras.layers import  MaxPooling2D
from keras.layers import Dense, Dropout, Activation, Flatten, GlobalAveragePooling2D, BatchNormalization
from keras.layers import Convolution2D
from keras.models import Sequential
import pickle
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split
from keras.callbacks import ModelCheckpoint
import keras
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix #class to calculate accuracy and other metrics
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

#define global variables
X = []
Y = []
path = "Dataset"
labels = []
```

```
#define function to load class labels
for root, dirs, directory in os.walk(path):
    for j in range(len(directory)):
        name = os.path.basename(root)
        if name not in labels:
            labels.append(name.strip())
def getLabel(name):
    index = -1
    for i in range(len(labels)):
        if labels[i] == name:
            index = i
            break
    return index
print("Accident Class Labels found in dataset : "+str(labels))

#loop and read all images from dataset
if os.path.exists('model/X.txt.npy'):#if images already processed then load all images
    X = np.load('model/X.txt.npy')
    Y = np.load('model/Y.txt.npy')
else:#if not processed then read and process each image
    X = []
    Y = []
    for root, dirs, directory in os.walk(path):
        for j in range(len(directory)):
            name = os.path.basename(root)
            if 'Thumbs.db' not in directory[j]:
                img = cv2.imread(root+"/"+directory[j])#read image
                img = cv2.resize(img, (64, 64))#resize image
                X.append(img)#addin images features to training array
                label = getLabel(name)
                Y.append(label)
    X = np.asarray(X)
    Y = np.asarray(Y)
    np.save('model/X.txt',X)
    np.save('model/Y.txt',Y)
print("Dataset Loading Completed")
print("Total images found in dataset : "+str(X.shape[0]))

#visualizing class labels count found in dataset
names, count = np.unique(Y, return_counts = True)
height = count
bars = labels
y_pos = np.arange(len(bars))
plt.figure(figsize = (5, 3))
plt.bar(y_pos, height)
plt.xticks(y_pos, bars)
plt.xlabel("Dataset Class Label Graph")
plt.ylabel("Count")
plt.xticks(rotation=90)
plt.show()

#preprocess images like shuffling and normalization
X = X.astype('float32')
X = X/255 #normalized pixel values between 0 and 1
indices = np.arange(X.shape[0])
np.random.shuffle(indices) #shuffle all images
X = X[indices]
Y = Y[indices]
Y = to_categorical(Y)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
print("Dataset Image Processing & Normalization Completed")
print("80% images used to train algorithms : "+str(X_train.shape[0]))
print("20% image used to test algorithms : "+str(X_test.shape[0]))
#define global variables to save accuracy and other metrics
accuracy = []
precision = []
recall = []
fscore = []
```

```
def calculateMetrics(algorithm, predict, y_test):
    a = accuracy_score(y_test,predict)*100
    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
    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))
    conf_matrix = confusion_matrix(y_test, predict)
    plt.figure(figsize =(6, 3))
    ax = sns.heatmap(conf_matrix, xticklabels = labels, yticklabels = labels, annot = True, 
    cmap="viridis" ,fmt ="g");
    ax.set_ylim([0,len(labels)])
    plt.title(algorithm+" Confusion matrix")
    plt.xticks(rotation=90)
    plt.ylabel('True class')
    plt.xlabel('Predicted class')
    plt.show()
#train CNN algorithm on accident detection image features
cnn_model = Sequential()
#defining CNN layer with 32 filters of 3 X # matrix for features filtration
cnn_model.add(Convolution2D(32, (3 , 3), input_shape = (X_train.shape[1], X_train.shape[2], 
X_train.shape[3]), activation = 'relu'))
#max layer to collected relevant filtered features from Previous CNN layer
cnn_model.add(MaxPooling2D(pool_size = (2, 2)))
#defining another CNN layer for further features optimization
cnn_model.add(Convolution2D(32, (3, 3), activation = 'relu'))
#collect relevant features
cnn_model.add(MaxPooling2D(pool_size = (2, 2)))
```

```
#change features dimension to single dimension
cnn_model.add(Flatten())
cnn_model.add(Dense(units = 256, activation = 'relu'))
cnn_model.add(Dense(units = y_train.shape[1], activation = 'softmax'))
#compile, train and load model
cnn_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
if os.path.exists("model/cnn_weights.hdf5") == False:
    model_check_point = ModelCheckpoint(filepath='model/cnn_weights.hdf5', verbose = 1,
save_best_only = True)
    hist = cnn_model.fit(X_train, y_train, batch_size = 32, epochs = 15, validation_data=(X_test,
y_test), callbacks=[model_check_point], verbose=1)
    f = open('model/cnn_history.pckl', 'wb')
    pickle.dump(hist.history, f)
    f.close()
else:
    cnn_model.load_weights("model/cnn_weights.hdf5")
#perform prediction on test images
predict = cnn_model.predict(X_test)
predict = np.argmax(predict, axis=1)
y_test1 = np.argmax(y_test, axis=1)
#call this function to calculate accuracy and other metrics
calculateMetrics("CNN Algorithm", predict, y_test1)
#reshaping images data to train with ML algorithms
y_train = np.argmax(y_train, axis=1)
y_test = np.argmax(y_test, axis=1)
X_train=np.reshape(X_train,(X_train.shape[0],(X_train.shape[1]*X_train.shape[2]*
X_train.shape[3])))
X_test = np.reshape(X_test,(X_test.shape[0],(X_test.shape[1]* X_test.shape[2] * X_test.shape[3])))
X_train = X_train[0:1000]
X_test = X_test[0:1000]
#training svm on accident images features
svm_cls = svm.SVC()
svm_cls.fit(X_train, y_train)#training SVM on train features
predict = svm_cls.predict(X_test)#predicting on test features
```

```
#call this function to calculate accuracy and other metrics
calculateMetrics("SVM Algorithm", predict, y_test)
#training DecisionTreeClassifier on accident images features
dt_cls = DecisionTreeClassifier()
dt_cls.fit(X_train, y_train)#training DecisionTreeClassifier on train features
predict = dt_cls.predict(X_test)#predicting on test features
#call this function to calculate accuracy and other metrics
calculateMetrics("Decision Tree Algorithm", predict, y_test)
#training RandomForestClassifier on accident images features
rf_cls = RandomForestClassifier()
rf_cls.fit(X_train, y_train)#training RandomForestClassifier on train features
predict = rf_cls.predict(X_test)#predicting on test features
#call this function to calculate accuracy and other metrics
calculateMetrics("Random Forest Algorithm", predict, y_test)
#comparison graph
import pandas as pd
df=pd.DataFrame([['CNN','Accuracy',accuracy[0]],['CNN','Precision',precision[0]],['CNN',
'Recall',recall[0]],['CNN','FSCORE',fscore[0]],['SVM','Accuracy',accuracy[1]],['SVM',
'Precision',precision[1]],['SVM','Recall',recall[1]],['SVM','FSCORE',fscore[1]], ['Decision
Tree','Accuracy',accuracy[2]],['DecisionTree','Precision',precision[2]],['Decision
Tree','Recall',recall[2]],['DecisionTree','FSCORE',fscore[2]], ['RandomForest','Accuracy',
accuracy[3]],['Random Forest','Precision',precision[3]],['RandomForest','Recall',recall[3]],
['Random Forest','FSCORE',fscore[3]], ],columns=['Parameters','Algorithms','Value'])
df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar', figsize=(8, 2))
plt.title("All Algorithms Performance Graph")
plt.show()
#display all algorithm performnace
algorithms = ['CNN', 'SVM', 'Decision Tree', 'Random Forest']
data = []
for i in range(len(accuracy)):
   data.append([algorithms[i], accuracy[i], precision[i], recall[i], fscore[i]])
data = pd.DataFrame(data, columns=['Algorithm Name', 'Accuracy', 'Precision', 'Recall',
'FSCORE'])
```

```
def getSeverity(image_path):
    injury_type = "Unable to detect"
    img = cv2.imread(image_path)
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    lower_red = np.array([0, 100, 120])
    upper_red = np.array([15, 255, 255])
    mask = cv2.inRange (hsv, lower_red, upper_red)
    contours, _=cv2.findContours(mask.copy(),cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    if len(contours) > 0:
        red_area = max(contours, key=cv2.contourArea)
        x, y, w, h = cv2.boundingRect(red_area)
        if w >= 100:
            injury_type = "Major Severity"
        else:
            injury_type = "Minor Severity"
        cv2.rectangle(img,(x, y),(x+w, y+h),(0, 0, 255), 2)
    return injury_type, img
def getRecommendation(label):
    with open("recommendation/"+label+".txt", "rb") as file:
        data = file.read()
    file.close()
    return data.decode()
#use this function to predict fish species uisng extension model
def predict(image_path):
    image = cv2.imread(image_path)#read test image
    img = cv2.resize(image, (64,64))#resize image
    im2arr = np.array(img)
    im2arr = im2arr.reshape(1,64,64,3)#convert image as 4 dimension
    img = np.asarray(im2arr)
    img = img.astype('float32')#convert image features as float
    img = img/255 #normalized image
    predict = cnn_model.predict(img)#perform prediction on test image
    predict = np.argmax(predict)
    injury_type, img = getSeverity(image_path)
```

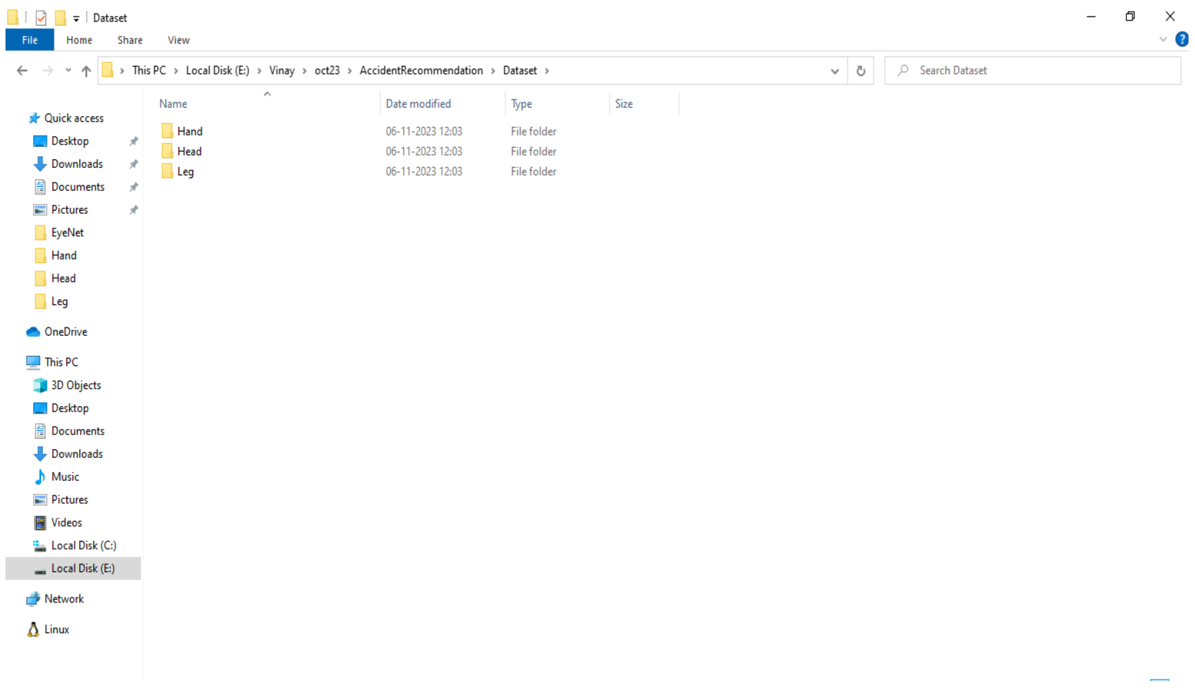
```
 img = cv2.resize(img, (400,300))#display image with predicted output
 img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
 recommendation = getRecommendation(labels[predict])
 print("Recommendation Details")
 print(recommendation)
 cv2.putText(img,'PredictedAs:'+labels[predict],(10,25),  cv2.FONT_HERSHEY_SIMPLEX,
0.7, (0, 0, 255), 2)
 cv2.putText(img,'InjuryType: '+injury_type,(10, 55),  cv2.FONT_HERSHEY_SIMPLEX,0.7,
  (0, 0, 255), 2)
 plt.figure(figsize=(4,3))
 plt.imshow(img)
#call this function with input image to recommend hospital
  predict("testImages/1.jpg")
 predict("testImages/4.jpg")
 predict("testImages/10.jpg")
 predict("testImages/9.jpg")
 predict("testImages/17.jpg")
```

5. RESULTS

5. RESULTS

The following screenshots showcase the results of our project, highlighting key features and functionalities. These visual representations provide a clear overview of how the system performs under various conditions, demonstrating its effectiveness and user interface. The screenshots serve as a visual aid to support the project's technical and operational achievements.

5.1 DATA SET FOLDER STRUCTURE:



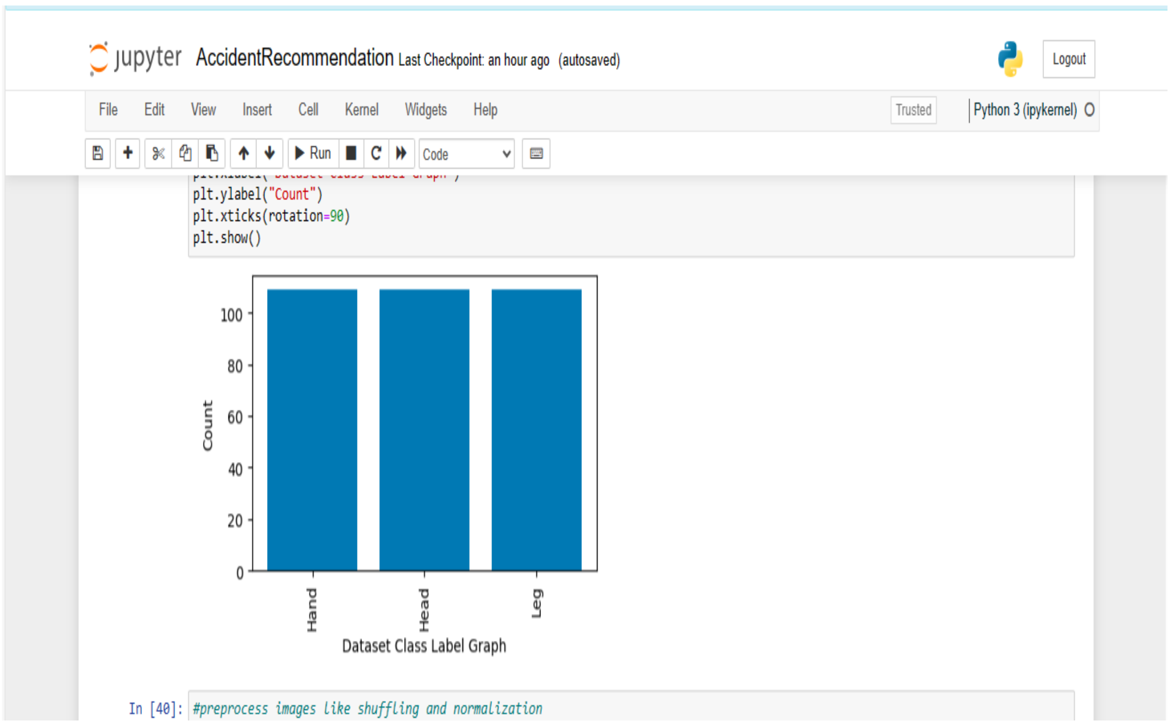
Name	Date modified	Type	Size
Hand	06-11-2023 12:03	File folder	
Head	06-11-2023 12:03	File folder	
Leg	06-11-2023 12:03	File folder	

Figure 5.1: View of Dataset Structure containing categorized folders for different injury types of Road Accident Severity and Hospital Recommendation using Deep Learning

DESCRIPTION:

The image displays the directory structure of the dataset used for training and testing. It shows that images are organized into separate folders such as Head, Hand, and Leg, representing different injury types. This structure supports automatic labeling during the model training phase.

5.2 DATA DISTRIBUTION GRAPH:


jupyter AccidentRecommendation Last Checkpoint: an hour ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Code
Count
Hand
Head
Leg
Dataset Class Label Graph

```
plt.ylabel("Count")
plt.xticks(rotation=90)
plt.show()
```

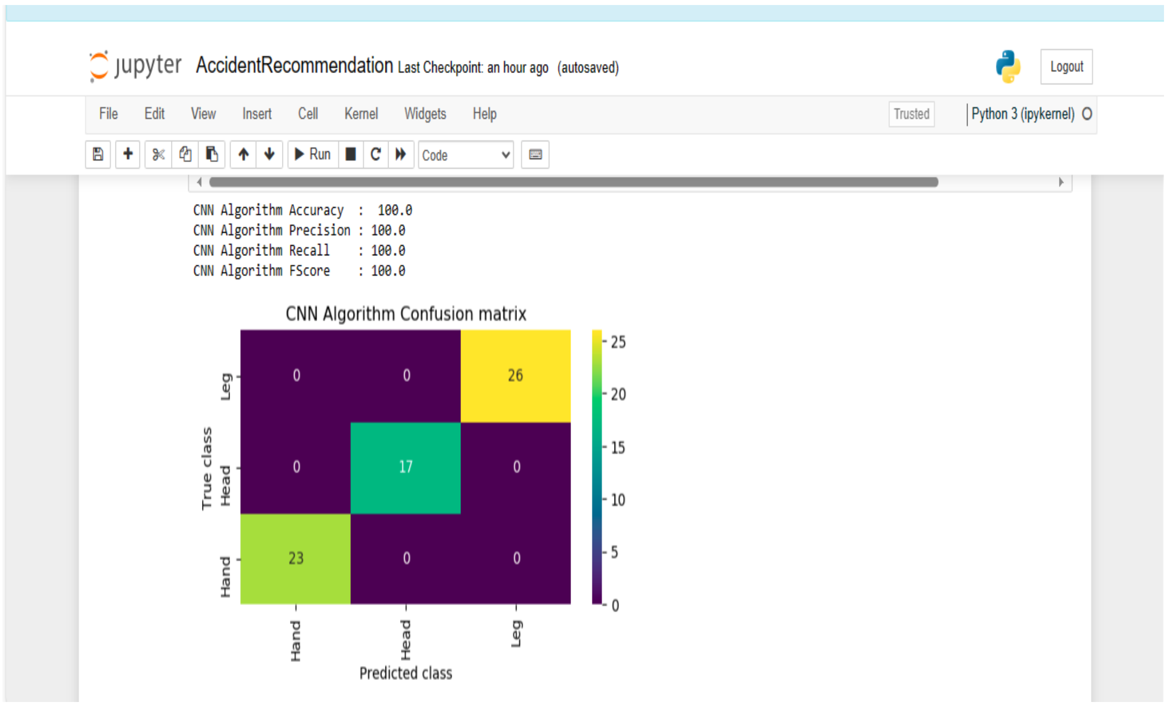
```
In [40]: #preprocess images like shuffling and normalization
```

Figure 5.2: Graph showing the number of images per injury category in the dataset of Road Accident Severity and Hospital Recommendation using Deep Learning

DESCRIPTION:

This image displays a bar graph representing the distribution of images across the three injury categories: Head, Hand, and Leg. The x-axis displays the injury types, while the y-axis indicates the number of images available in each category. It helping to assess the class balance in the dataset. Balanced data ensures unbiased training of the model.

5.3 CNN MODEL TRAINING OUTPUT:



```
CNN Algorithm Accuracy  :  100.0
CNN Algorithm Precision : 100.0
CNN Algorithm Recall    : 100.0
CNN Algorithm FScore    : 100.0
```

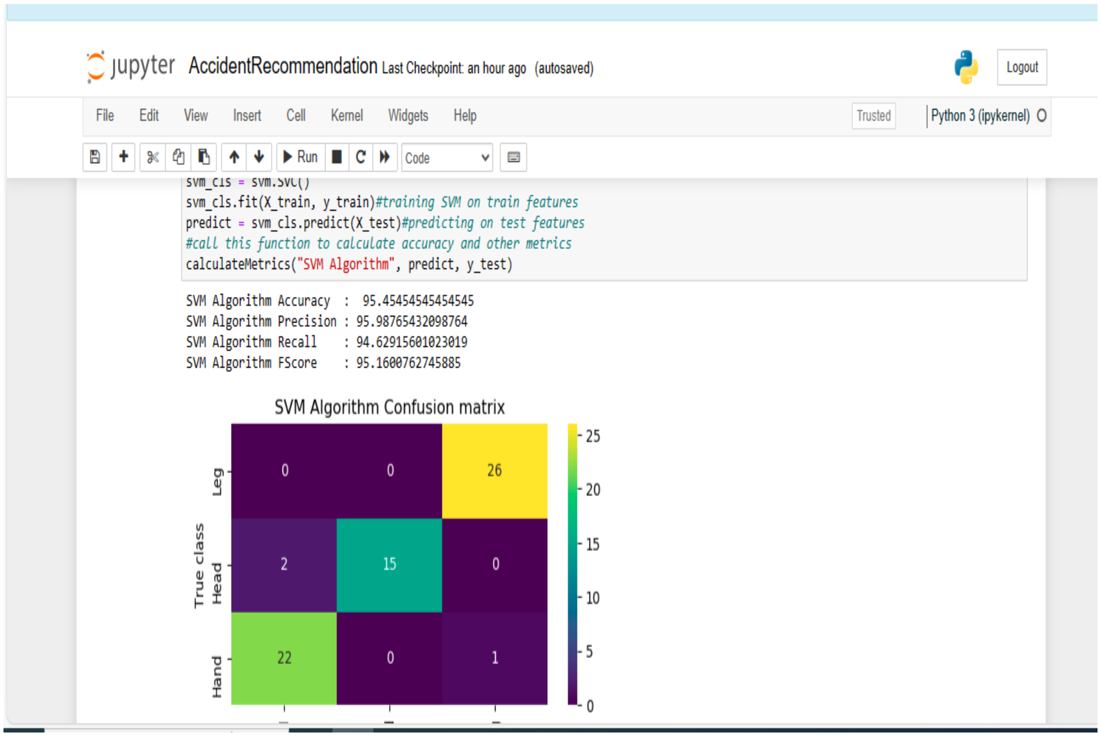
	Hand	Head	Leg
Leg	0	0	26
Head	0	17	0
Hand	23	0	0

Figure 5.3: CNN model training results of Road Accident Severity and Hospital Recommendation using Deep Learning

DESCRIPTION:

This image displays the result after training the Convolutional Neural Network (CNN) model on the accident injury dataset. It shows that the CNN achieved 100% accuracy, indicating perfect classification during the test phase. The output also includes key performance metrics like precision, recall, and F1-score, along with a confusion matrix that confirms all predictions were correctly classified. This result highlights the high effectiveness of CNN in identifying and classifying injury types based on image input.

5.4 SVM MODEL PERFORMANCE:


jupyter AccidentRecommendation Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
SVM Algorithm Confusion matrix
True class
Leg
Head
Hand
0
0
26
2
15
0
22
0
1
25
20
15
10
5
0

```
svm_cls = svm.SVC()
svm_cls.fit(X_train, y_train)#training SVM on train features
predict = svm_cls.predict(X_test)#predicting on test features
#call this function to calculate accuracy and other metrics
calculateMetrics("SVM Algorithm", predict, y_test)
```

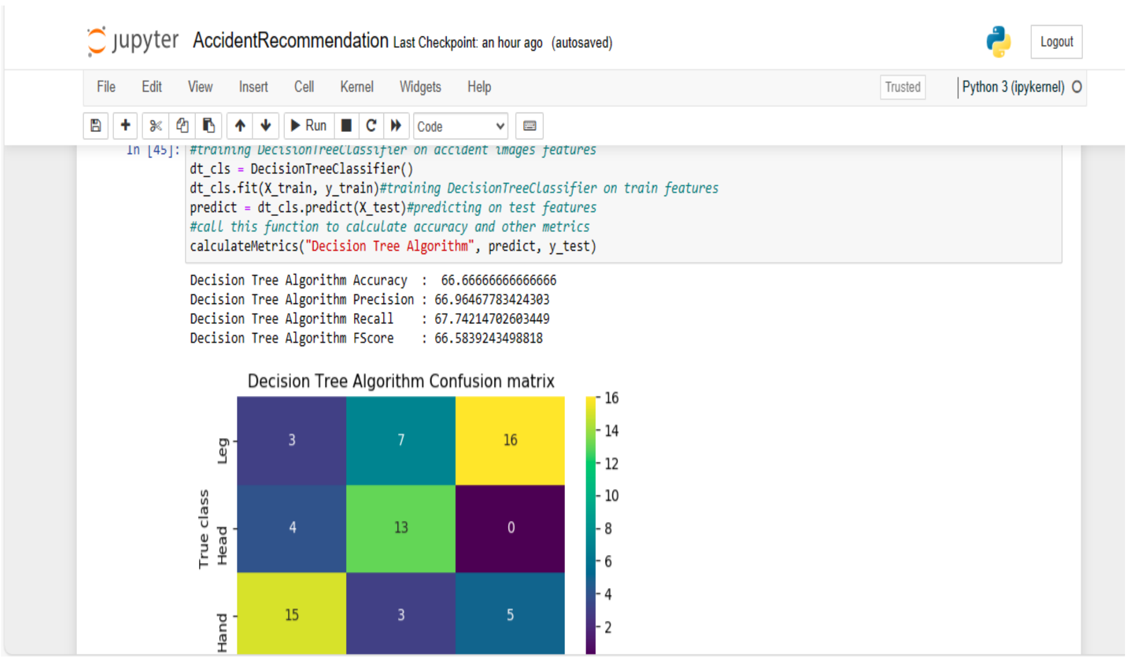
```
SVM Algorithm Accuracy  :  95.45454545454545
SVM Algorithm Precision : 95.98765432098764
SVM Algorithm Recall    : 94.62915601023019
SVM Algorithm FScore    : 95.1600762745885
```

Figure 5.4: Support Vector Machine model training results of Road Accident Severity and Hospital Recommendation using Deep Learning.

DESCRIPTION:

This image shows the output of the Support Vector Machine (SVM) model after training on the injury dataset. The SVM achieved a commendable accuracy of **95%**, indicating its effectiveness in classifying images based on injury type and severity. The output also includes precision, recall, and F1-score, confirming that the model performs well, though slightly less accurately than CNN the Support vector Machine still performs well and serves as a good benchmark for comparison.

5.5 DECISION TREE MODEL PERFORMANCE:



```
In [45]: #training DecisionTreeClassifier on accident images features
         dt_cls = DecisionTreeClassifier()
         dt_cls.fit(X_train, y_train)#training DecisionTreeClassifier on train features
         predict = dt_cls.predict(X_test)#predicting on test features
         #call this function to calculate accuracy and other metrics
         calculateMetrics("Decision Tree Algorithm", predict, y_test)
```

```
Decision Tree Algorithm Accuracy  :  66.66666666666666
Decision Tree Algorithm Precision : 66.96467783424303
Decision Tree Algorithm Recall    : 67.74214702603449
Decision Tree Algorithm FScore    : 66.5839243498818
```

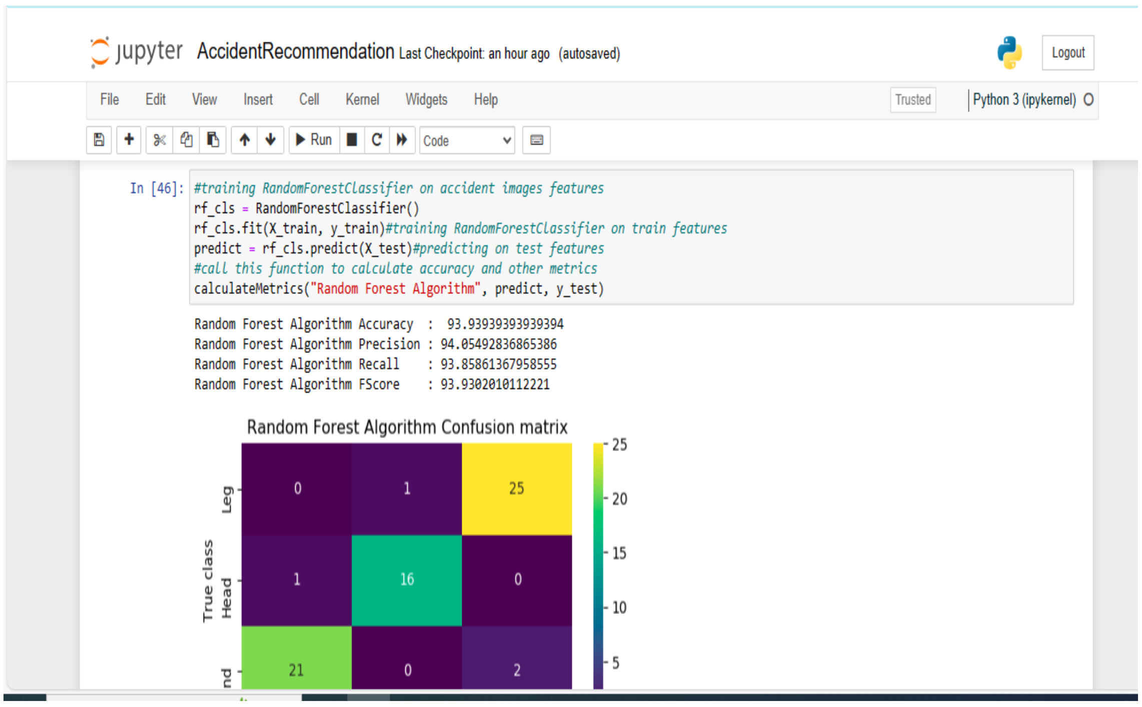
jupyter AccidentRecommendation Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Decision Tree Algorithm Confusion matrix
True class
Leg
Head
Hand
3
7
16
4
13
0
15
3
5
16
14
12
10
8
6
4
2

Figure 5.5: Decision Tree model training results of Road Accident Severity and Hospital Recommendation using Deep Learning.

DESCRIPTION:

The image shows the training results of the Decision Tree model, which achieved an accuracy of only **66%**. The lower performance indicates the model’s limited ability to capture complex features in image data. The confusion matrix reveals a higher number of misclassifications, making it less suitable for precise injury classification. This result highlights the need for more advanced models like Convolutional Neural Network.

5.6 RANDOM FOREST MODEL PERFORMANCE:


jupyter AccidentRecommendation Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Random Forest Algorithm Confusion matrix
True class
Leg
Head
nd
0 1 25
1 16 0
21 0 2
25 20 15 10 5

```
In [46]: #training RandomForestClassifier on accident images features
         rf_cls = RandomForestClassifier()
         rf_cls.fit(X_train, y_train)#training RandomForestClassifier on train features
         predict = rf_cls.predict(X_test)#predicting on test features
         #call this function to calculate accuracy and other metrics
         calculateMetrics("Random Forest Algorithm", predict, y_test)

         Random Forest Algorithm Accuracy  :  93.93939393939394
         Random Forest Algorithm Precision : 94.05492836865386
         Random Forest Algorithm Recall    : 93.8586136795855
         Random Forest Algorithm FScore    : 93.930201011221
```

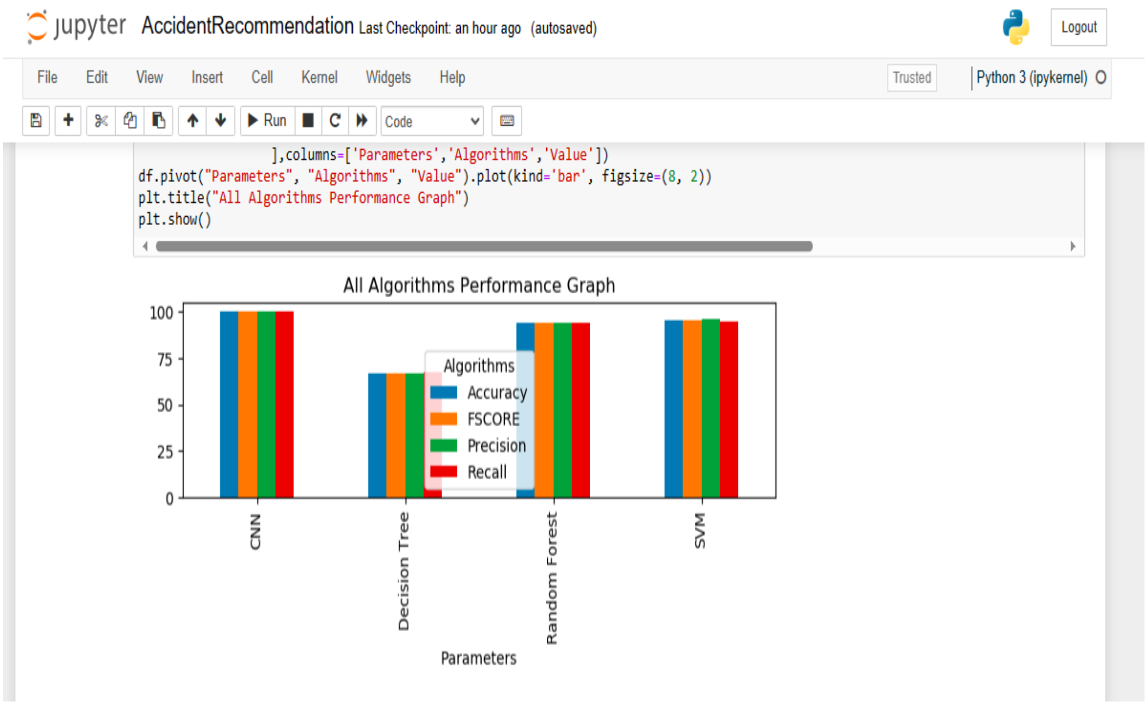
Figure 5.6: Random Forest model training results of Road Accident Severity and Hospital Recommendation using Deep Learning.

DESCRIPTION:

The image Displays the Random Forest model's training results on the accident injury dataset. It achieved an accuracy of **93%**, making it more effective than Decision Tree but slightly below CNN and SVM. The metrics and confusion matrix demonstrate that the model can handle some data complexity, offering reliable classification for most cases.

.

5.7 PERFORMANCE COMPARISON BAR GRAPH:



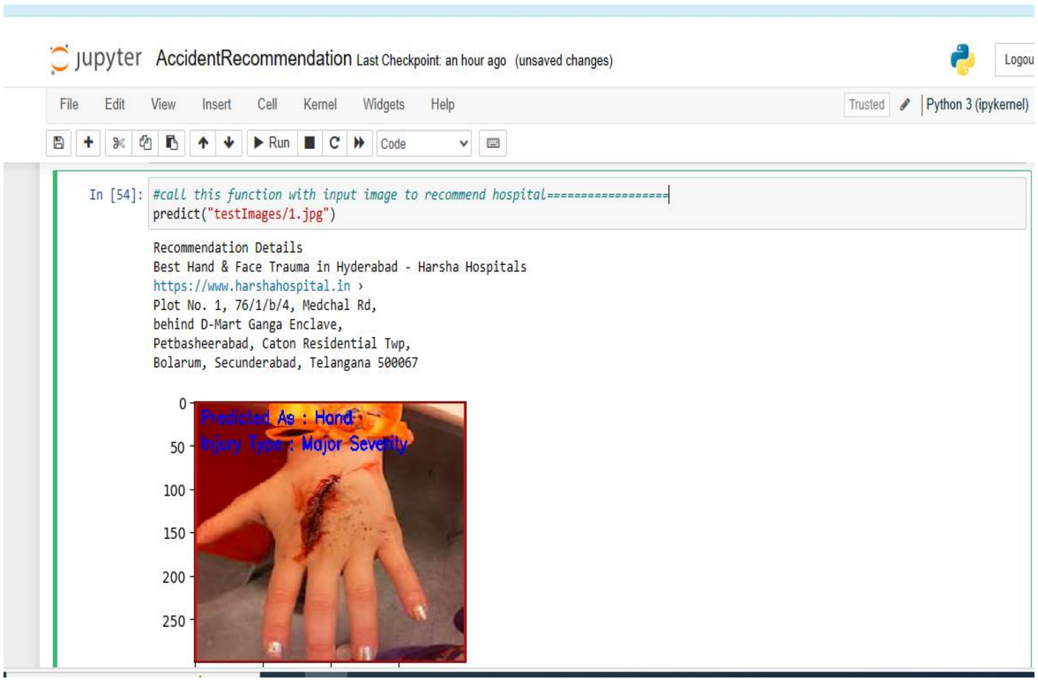
```
                   ],columns=['Parameters','Algorithms','Value'])
df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar', figsize=(8, 2))
plt.title("All Algorithms Performance Graph")
plt.show()
```

Figure 5.7: Comparison chart showing model performances of Road Accident Severity and Hospital Recommendation using Deep Learning.

DESCRIPTION:

The image represents a comparative bar chart that visualizes the performance metrics—accuracy, precision, recall, and F1-score—for CNN, SVM, Decision Tree, and Random Forest. The graph clearly highlights CNN as the most accurate and reliable model, achieving nearly perfect results across all metrics. SVM and Random Forest follow closely, whereas the Decision Tree shows significantly lower performance. This graphical comparison offers a quick and intuitive understanding of how each algorithm fares in injury classification.

5.8 SAMPLE PREDICTION OUTPUT -1:


jupyter AccidentRecommendation Last Checkpoint: an hour ago (unsaved changes)
Logou
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Predicted As : Hand
Injury Type : Major Severity

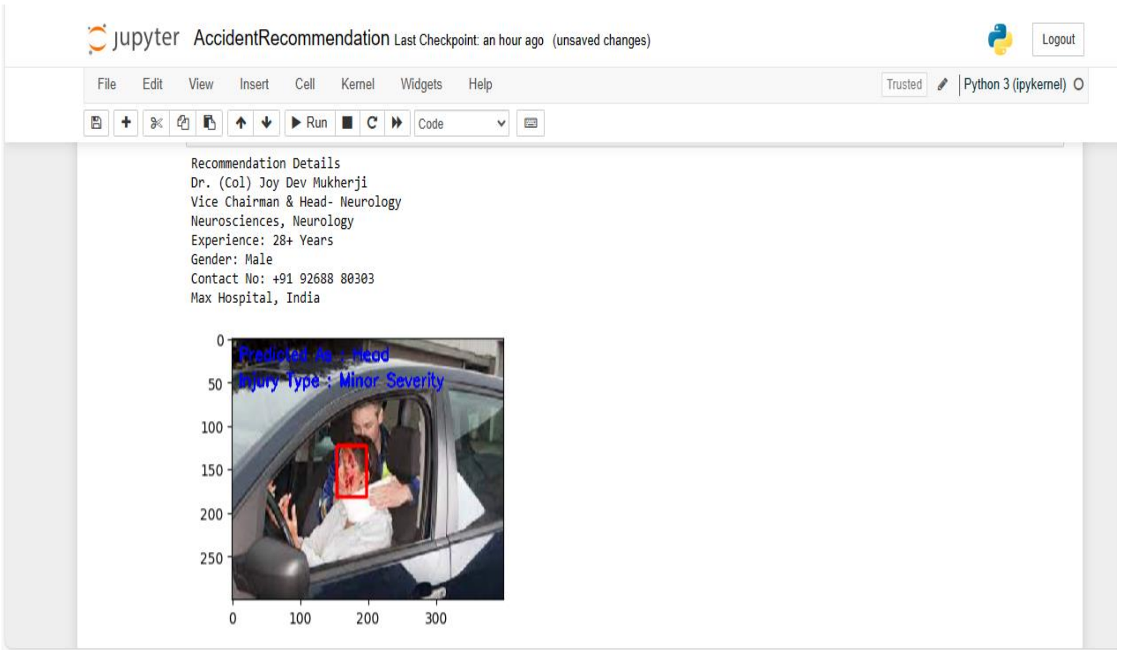
```
In [54]: #call this function with input image to recommend hospital=================
         predict("testImages/1.jpg")

         Recommendation Details
         Best Hand & Face Trauma in Hyderabad - Harsha Hospitals
         https://www.harshahospital.in ›
         Plot No. 1, 76/1/b/4, Medchal Rd,
         behind D-Mart Ganga Enclave,
         Petbasheerabad, Caton Residential Twp,
         Bolarum, Secunderabad, Telangana 500067
```

Figure 5.8: Predicted result for test image showing Hand injury with Major severity of Road Accident Severity and Hospital Recommendation using Deep Learning.

DESCRIPTION:

This image shows the output of the system predicting a major hand injury from the test image. The injury is clearly marked with a bounding box, and based on the severity, the system recommends Harsha Hospitals, a reputed center for hand and face trauma located in Hyderabad. The full address and website details are provided for immediate access. This image validates the CNN model's efficiency in injury classification and relevant hospital recommendation.

5.9 SAMPLE PREDICTION OUTPUT - 2:


jupyter AccidentRecommendation Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Predicted As : Head
Injury Type : Minor Severity
0
50
100
150
200
250
0
100
200
300

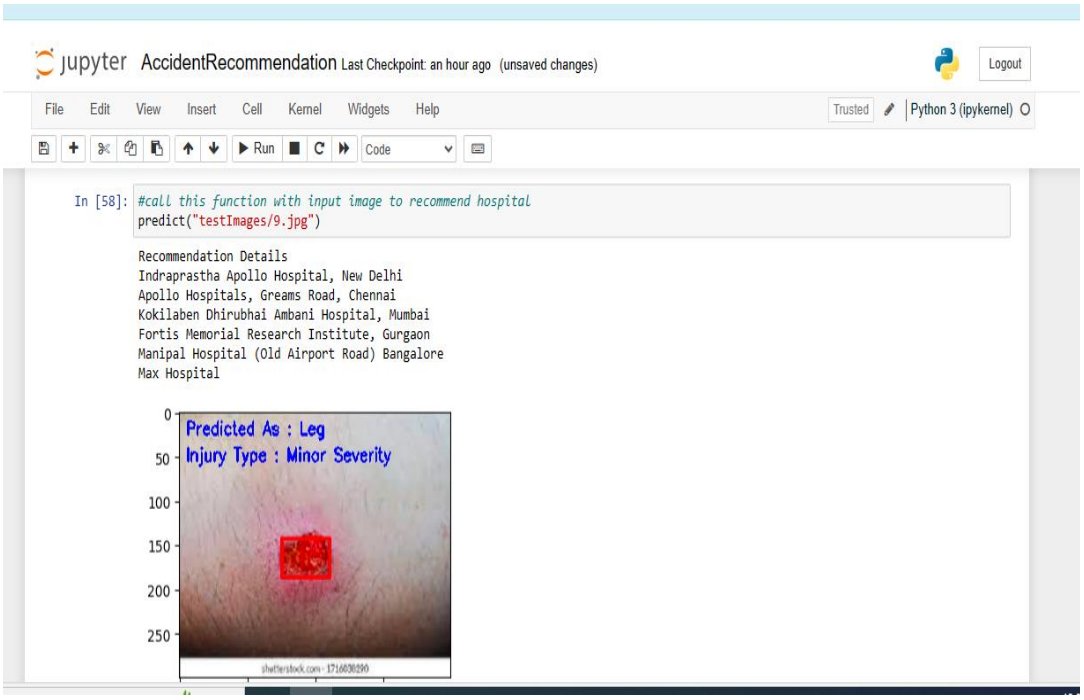
```
Recommendation Details
Dr. (Col) Joy Dev Mukherji
Vice Chairman & Head- Neurology
Neurosciences, Neurology
Experience: 28+ Years
Gender: Male
Contact No: +91 92688 80303
Max Hospital, India
```

Figure 5.9: Predicted result for test image showing Head injury with Minor severity of Road Accident Severity and Hospital Recommendation using Deep Learning.

DESCRIPTION:

The image represents the prediction outcome of a head injury scenario, where the system classified the injury as minor severity. Based on this classification, the system recommends consultation with Dr. (Col) Joy Dev Mukherji, a senior neurologist from Max Hospital, India. Along with the prediction, detailed contact and professional information of the specialist is provided. The image clearly highlights the injured region with a bounding box, supporting visual confirmation of the model’s accuracy.

5.10 SAMPLE PREDICTION OUTPUT - 3:


jupyter AccidentRecommendation Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Predicted As : Leg
Injury Type : Minor Severity

```
In [58]: #call this function with input image to recommend hospital
         predict("testImages/9.jpg")

         Recommendation Details
         Indraprastha Apollo Hospital, New Delhi
         Apollo Hospitals, Greams Road, Chennai
         Kokilaben Dhirubhai Ambani Hospital, Mumbai
         Fortis Memorial Research Institute, Gurgaon
         Manipal Hospital (Old Airport Road) Bangalore
         Max Hospital
```

Figure 5.10 : Predicted result for test image showing Head injury with Minor severity of Road Accident Severity and Hospital Recommendation using Deep Learning.

DESCRIPTION:

The image displays the result of a sample prediction using the CNN-based system. The test image was classified as a leg injury with minor severity. Based on this prediction, the system automatically provided a list of nearby hospitals suitable for treating such injuries, including well-known names like Apollo, Max, and Fortis. The predicted image section highlights the injury area using a red bounding box. This output demonstrates the system’s ability to process new images accurately and suggest appropriate medical facilities in real-time.

6. VALIDATION

6. VALIDATION

The validation of this project primarily relies on extensive testing and well-defined test cases to ensure the accuracy and effectiveness of the inappropriate content detection system. The testing process involves multiple stages, including dataset validation, model performance evaluation, and real-world testing. By implementing a structured validation approach, we can ensure that the system consistently delivers high accuracy in detecting inappropriate content while minimizing false positives and false negatives.

6.1 INTRODUCTION

In this project, which focuses on predicting road accident severity and recommending hospitals using deep learning, validation helps assess how effectively the system performs when exposed to new or unseen data. Various machine learning and deep learning algorithms, including Convolutional Neural Networks (CNN), Support Vector Machines (SVM), Random Forest, and Decision Tree, were applied to classify accident injuries and identify their severity levels. To validate the performance of each model, we used a set of evaluation metrics such as accuracy, precision, recall, F1-score, and confusion matrix. These metrics offer insight into how well each model can detect different types of injuries (head, hand, leg) and classify them as major or minor. Additionally, the system’s ability to recommend suitable hospitals based on the injury classification was also validated. The results obtained from this validation help to prove that the proposed approach not only outperforms traditional methods but also offers real-time, accurate predictions that can aid in faster decision-making during emergencies.

6.2 TEST CASES

6.2.1 UPLOADING DATASET

Test case ID	Test case name	Purpose	Test Case	Output
1	Uploads Dataset.	Use it for content prediction.	The user uploads the Dataset, on which the content is detected.	Dataset successfully loaded.

6.2.2 CLASSIFICATION

Test case ID	Test case name	Purpose	Input	Output
1	Classification -1	To check if the model detects minor severity	Image input showing light external injuries	Predicted as Minor Severity
2	Classification -2	To check if the model detects major severity	Image input showing severe injury	Predicted as Major Severity

7. CONCLUSION AND FUTURE ASPECTS

7. CONCLUSION AND FUTURE ASPECTS

In conclusion, the project has successfully achieved its objectives, showcasing significant progress and outcomes. The implementation and execution phases were meticulously planned and executed, leading to substantial improvements and insights. Looking ahead, the future aspects of the project hold immense potential. Future developments will focus on expanding the scope, integrating new technologies, and enhancing sustainability. These advancements will not only strengthen the existing framework but also open new avenues for growth and innovation, ensuring the project remains relevant and impactful in the long term. This strategic approach will drive continuous improvement and success.

7.1 PROJECT CONCLUSION

In conclusion, the road accident severity and hospital recommendation project represent a significant step forward in leveraging technology to improve emergency response and healthcare delivery in the context of road accidents. By harnessing deep learning techniques such as Convolutional Neural Networks (CNNs) for injury severity detection and hospital recommendation, we have developed a robust system capable of automating critical aspects of post-accident management. Through the implementation of our system, we address several shortcomings of the existing manual assessment methods, including subjectivity, inconsistency, and time-consuming processes. By automating injury severity detection, we ensure swift and accurate assessment, enabling emergency responders and healthcare professionals to make informed decisions promptly. Additionally, our system provides consistent and objective hospital recommendations based on the severity of injuries, ensuring that accident victims receive timely access to appropriate medical care.

Overall, this project proves that AI-driven solutions are not only feasible but also highly effective in addressing real-world challenges in public health and safety. It sets a foundation for future work that could include expanding the model’s capabilities, improving prediction accuracy, and connecting directly with ambulance and hospital systems. Through this project, a crucial step has been taken towards building a technology-assisted emergency ecosystem.

7.2 FUTURE ASPECTS

The project "Road Accident Severity and Hospital Recommendation using Deep Learning" holds immense potential for future expansion and real-world implementation. One major enhancement could be the integration with real-time accident detection systems using IoT-enabled traffic cameras or vehicle sensors, enabling automatic injury analysis and instant hospital recommendations. The classification model, currently limited to major and minor injuries, can be extended to multiple severity levels for more precise triaging. Furthermore, connecting the system with live hospital databases will allow dynamic decision-making based on bed availability, distance, and specialization. A user-friendly mobile or web application can also be developed for the public to report accidents and receive immediate assistance. Integration with GPS and navigation systems will help direct victims or emergency responders to the recommended hospital in real-time. Future versions may incorporate advanced deep learning architectures like Vision Transformers for improved accuracy and include vital signs from wearable sensors for holistic injury assessment. To ensure safe deployment, data privacy and security measures such as encryption and compliance with healthcare regulations will be crucial. With collaboration from government and healthcare providers, this system could become a core part of national emergency response strategies and smart city infrastructure.

8. BIBLIOGRAPHY

8. BIBLIOGRAPHY

8.1 REFERENCES

1. Sameen, M.I., & Pradhan, B. (2016). Assessment of the effects of expressway geometric design features on the frequency of accident crash rates using high-resolution laser scanning data and GIS. *Geomat. Nat. Hazards Risk*, 1–15.

2. Pei, X., Wong, S., & Sze, N.-N. (2011). A joint-probability approach to crash prediction models. *Accident Analysis & Prevention*, 43, 1160–1166.

3. Fogue, M., Garrido, P., Martinez, F.J., Cano, J.-C., Calafate, C.T., & Manzoni, P. (2014). A system for automatic notification and severity estimation of automotive accidents. *IEEE Transactions on Mobile Computing*, 13, 948–963.

4. Pawlus, W., Reza, H., & Robbersmyr, K.G. (2011). Application of viscoelastic hybrid models to vehicle crash simulation. *International Journal of Crashworthiness*, 16, 195–205.

5. Karimi, H.R., Pawlus, W., & Robbersmyr, K.G. (2012). Signal reconstruction, modeling, and simulation of a vehicle full-scale crash test based on Morlet wavelets. *Neurocomputing*, 93, 88–99.

6. Abdelwahab, H., & Abdel-Aty, M. (2001). Development of artificial neural network models to predict driver injury severity in traffic accidents at signalized intersections. *Transportation Research Record: Journal of the Transportation Research Board*, 1746, 6–13.

8.2 GITHUB LINK

https://github.com/Chinthakindi-Rushmitha/Road_Accident_Severity_and_Hospital_Recommendation_using_DL