A

Project Report

On

UNSUPERVISED MACHINE LEARNING FOR MANAGING SAFETY ACCIDENTS IN RAILWAY STATIONS

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

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING (AI&ML)

By

D. RISHENDRA (217R1A6616)
RAHITYA DEETI (217R1A6648)
P. CHANIKYA (217R1A6646)

Under the Guidance of

Mrs. BUSHRA TARANNUM

Assistant Professor CSE (AI&ML)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

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 UGC Act.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

2021-2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

CERTIFICATE

This is to certify that the project entitled **"UNSUPERVISED MACHINE LEARNING FOR MANAGING SAFETY ACCIDENTS IN RAILWAY STATIONS"** being submitted by **D. RISHENDRA (217R1A6616), RAHITYA DEETI (217R1A6648) & P. CHANIKYA (217R1A6646)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering (AI&ML) 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 thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Mrs. Bushra Tarannum
Asst. Professor CSE (AI&ML)
INTERNAL GUIDE

Dr. S Rao Chintalapudi
HOD CSE (AI&ML)

EXTERNAL EXAMINER

Submitted for viva voice Examination held on _____

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. 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 my profound gratitude and deep regard to my guide **Mrs. Bushra Tarannum,** Assistant Professor CSE (AI&ML) for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **Dr. G. Vinoda Reddy, Dr. K. Mahesh, Dr. V. Malsoru, Dr. Md. Shareef, M. Balaji and A. Ramesh** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. S Rao Chintalapudi,** Head, Department of Computer Science and Engineering (AI&ML) for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

D. RISHENDRA (217R1A6616)

RAHITYA DEETI (217R1A6648)

P. CHANIKYA (217R1A6646)

ABSTRACT

For both passenger and freight transportation, railroad operations must be dependable, accessible, maintained, and safe (RAMS). In many urban areas, railway stations risk and safety accidents represent an essential safety concern for daily operations. Moreover, the accidents lead to damage to market reputation, including injuries and anxiety among the people and costs. This stations under pressure caused by higher demand which consuming infrastructure and raised the safety administration consideration. To analyzing these accidents and utilizing the technology such AI methods to enhance safety, it is suggested to use unsupervised topic modelling for better understand the contributors to these extreme accidents. It is conducted to optimize Latent Dirichlet Allocation (LDA) for fatality accidents in the railway stations from textual data gathered RSSB including 1000 accidents in the UK railway station. This research describes using the machine learning topic method for systematic spot accident characteristics to enhance safety and risk management in the stations and provides advanced analyzing.

The study evaluates the efficiency of text by mining from accident history, gaining information, lesson learned and deeply coherent of the risk caused by assessing fatalities accidents for large and enduring scale. This Intelligent Text Analysis presents predictive accuracy for valuable accident information such as root causes and the hot spots in the railway stations. Further, the big data analytics' improvement results in an understanding of the accidents' nature in ways not possible if a considerable amount of safety history and not through narrow domain analysis of the accident reports. This technology renders stand with high accuracy and a beneficial and extensive new era of AI applications in railway industry safety and other fields for safety applications.

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
3.1	Project Architecture Of Unsupervised Machine Learning For Managing Safety Accidents In Railway Stations.	12
3.2	Use Case Diagram for Unsupervised Machine Learning For Managing Safety Accidents In Railway Stations.	14
3.3	Class Diagram for Unsupervised Machine Learning For Managing Safety Accidents In Railway Stations.	15
3.4	Sequence diagram for Unsupervised Machine Learning For Managing Safety Accidents In Railway Stations.	17
4.3	Confusion Matrix	26

5.1	Run XAMPP & Command Prompt	35
5.2	Homepage	36
5.3	Remote user login and registration page	36
5.4	Prediction of railway accident type from the collected dataset	38
5.5	Login service provider	39
5.6	Prediction of viewed railway accident type ratio	40
5.7	Prediction of view trained & tested railway datasets accuracy in bar chart	41

LIST OF TABLES

TABLE NO	TABLE NAME	PAGE NO
6.3	TESTCASES	49

TABLE OF CONTENTS

ABSTRACT i

LIST OF FIGURES ii

LIST OF TABLES iii

1. INTRODUCTION 1

1.1 PROJECT SCOPE 1

1.2 PROJECT PURPOSE 1

1.3 PROJECT FEATURES 2

2. SYSTEM ANALYSIS 3

2.1 PROBLEM DEFINITION 4

2.2 EXISTING SYSTEM / LITERATURE REVIEW 5

2.2.1 EXISTING SYSTEM 5

2.2.2 LIMITATIONS OF EXISTING SYSTEMS 8

2.3 PROPOSED SYSTEM 8

2.3.1 PROPOSED APPROACH 8

2.3.2 ADVANTAGES OF PROPOSED SYSTEM 9

2.4 HARDWARE & SOFTWARE REQUIREMENTS 9

2.4.1 HARDWARE REQUIREMENTS 9

2.4.2 SOFTWARE REQUIREMENTS 10

3. ARCHITECTURE 11

3.1 PROJECT ARCHITECTURE 12

3.2 USE CASE DIAGRAM 14

3.3 CLASS DIAGRAM 15

3.4 SEQUENCE DIAGRAM 17

4. IMPLEMENTATION 19

4.1 DECISION TREE CLASSIFIERS 20

4.2 GRADIENT BOOSTING 20

4.3 K-NEAREST NEIGHBORS (KNN) 20

4.4 LOGISTIC REGRESSION CLASSIFIERS 21

4.5 NAÏVE BAYES 22

4.6 RANDOM FOREST 23

4.7 SUPPORT VECTOR MACHINE (SVM) 24

4.8 PERFORMANCE METRICS 25
4.9 SAMPLE CODE 27
5. SCREENSHOTS 34
6. TESTING 42
6.1 INTRODUCTION TO TESTING 43
6.2 TYPES OF TESTING 43
6.2.1 UNIT TESTING 43
6.2.2 INTEGRATION TESTING 43
6.2.3 FUNCTIONAL TESTING 48
6.3 TEST CASES 49
7.CONCLUSION & FUTURE SCOPE **50**
7.1 CONCLUSION 50
7.2 FUTURE SCOPE 51
BIBLIOGRAPHY 53
REFERENCES 54

1. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

The scope of this project focuses on analyzing railway station accidents using smart data-driven techniques, particularly leveraging topic modeling and AI-based approaches. The study examines accident reports from 01/01/2000 to 17/04/2020, utilizing data provided by RSSBS. The primary goal is to develop a systematic and efficient risk management framework that enhances railway safety through advanced analytics. The research explores unstructured textual data, aiming to identify root causes of accidents, risk factors, and underlying patterns. The study also integrates AI and deep learning techniques to improve data processing, classification, and predictive safety measures.

1.2 PROJECT PURPOSE

The primary objective of this project is to enhance railway station safety by identifying key accident causes and proposing a smart, AI-driven risk management process. Railway stations are complex environments with multiple overlapping risk factors, including station design, operational efficiency, and passenger behaviours. Despite global safety measures, accidents still occur, leading to fatalities, injuries, delays, and operational disruptions.

To address this, the study employs Latent Dirichlet Allocation (LDA), a topic modelling technique used in Natural Language Processing (NLP), to extract meaningful insights from accident reports. By automating accident data analysis, this approach helps decision-makers identify trends, improve safety measures, and implement proactive risk management strategies. This project ultimately contributes to developing smart safety systems that leverage past accident records for improved future prevention.

1.3 PROJECT FEATURES

The project introduces an AI-powered safety analysis system designed to enhance railway station risk management by leveraging machine learning, natural language processing (NLP), and topic modeling techniques. It utilizes RSSBS accident data (2000–2020) to analyze unstructured textual reports, identifying root causes and underlying patterns of railway accidents. By implementing Latent Dirichlet Allocation (LDA), the system extracts key risk factors and provides structured insights for better decision-making. Additionally, the project integrates AI-driven predictive analytics to anticipate potential hazards and support real-time safety interventions. The automated textual analysis feature simplifies complex accident reports, making them understandable for non-expert stakeholders, reducing manual effort in safety audits, and improving operational efficiency. Ultimately, this project lays the foundation for smart railway safety systems, enabling authorities to design proactive safety measures and policies based on historical accident data.

2. SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

2.1 PROBLEM DEFINITION:

Railway accidents pose a significant threat to passenger safety and operational efficiency, often resulting in loss of life, infrastructure damage, and financial losses. Traditional accident management and prediction systems rely on historical data and predefined rules, which may not effectively capture emerging patterns and hidden correlations in accident causes.

The challenge is to develop an unsupervised machine learning model that can analyze vast amounts of railway accident data, identify underlying patterns, and classify different types of incidents without predefined labels. By leveraging clustering algorithms, anomaly detection techniques, and topic modeling, the system aims to detect potential risk factors, predict accident-prone areas, and provide insights for proactive safety measures.

This approach will enable railway authorities to automate accident analysis, enhance risk assessment, and improve safety protocols by uncovering hidden relationships in accident data, ultimately reducing the frequency and severity of railway incidents.

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client.

2.2 EXISTING SYSTEM/LITERATURE REVIEW

2.2.1 EXISTING SYSTEM

Despite the scatter of applying such method and the differences in terms been using in the literature, there is a shortage of such applications in the railway industry. Moreover, the NLP has been implemented to detect defects in the requirements documents of a railway signaling manufacturer. Also, for translating terms of the contract into technical specifications in the railway sector. Additionally, identifying the significant factors contributing to railway accidents, the taxonomy framework was proposed using (Self-Organizing Maps – SOM), to classify human, technology, and organization factors in railway accidents. Likewise, association rules mining has been used to identify potential causal relationships between factors in railway accidents.

In the field of the machine learning and risk, safety accident, and occupational safety, there are many ML algorithms been used such as SVM, ANN, extreme learning machine (ELM), and decision tree (DT). Scholars have

been conducted the topic modeling in, where such method has been proved as one of the most powerful methods in data mining many fields and applied in various areas such as software engineering, medical and health and linguistic science, etc., Furthermore, from the literature It has been utilized this technique in for predictions some areas such as occupational accident, construction and aviation. For Understand occupational construction incidents in the construction and for construction injury prediction the method been conducted for analyzing the factors associated with occupational falls, for steel factory occupational incidents and Cybersecurity and Data Science. Moreover, from 156 construction safety accidents reports in urban rail transport in china risks information, relationships and factors been extracting and identified for safety risk analysis. From the literature it has been seen that, there is no perfect model for all text classifications issues and also the process of extracting information from text is an incremental. In the railway sector, a semi-automated method has been examined for classifying unstructured text-based close call reports which show high accuracy. Moreover, for future expectations, it has been reported that such technology could be compulsory for safety management in railway.

Applying text analyzing methods in railway safety expected to solve issues such as time-consuming analysis and incomplete analysis. Additionally, some advantages have been proved, automated process, high productivity with quality and effective system for supervision safety in the railway system. Moreover, For the prevention of railway accidents, machine learning methods have been conducted. Many methods used for data mining including machine learning, information extraction (IE), natural language processing (NLP), and information retrieval (IR). For instance, to improve the identification of secondary crashes, a text mining approach (classification) based on machine learning been applied to distinguish secondary crashes based on crash narratives, which appear satisfactory performance and has great potential for identifying secondary crashes. Such methods are powerful for railway safety, which aid decision-maker, investigate the causes of the accident, the relevant factors, and their correlations. It has been proved that text mining has several areas of future work development and advances for safety engineering railway.

Text mining with probabilistic modeling and k-means clustering is helpful for the knowledge of causes factors to rail accidents. From that application analysis for reports about major railroad accidents in the United States and the Transportation Safety Board of Canada, the study has been designating out that the factors of lane defects, wheel defects, level crossing accidents and switching accidents can lead to the many of recurring accidents.

An accident reports data for 11 years in the U.S. are analyzed by the combination of text analysis with ensemble methods has been used to better understand the contributors and characteristics of these accidents, yet and more research is needed. Also, from the U.S, railroad equipment accidents report are used to identify themes using a comparison text mining method (Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA)). Additionally, to identify the main factors associated with injury severity, data mining methods such as an ordered probit model, association rules, and classification and regression tree (CART) algorithms have been conducted.

Using the U.S accidents highway railroad grade crossings database for the period 2007–2013, where Some factors have been discussed such the train speed, age, gender and the time. In recent years, the revolution of big data is opportunities in the railway industry, and that is opening up for safety analysis depends on data. so, the approach to proactively identify high-risk scenarios been recommended such as applying the Natural Language Processing (NLP) analysis.

From Big Data Application Case A Supervision System has been introduced as a significant role tool in railway safety supervision system. Applying Text Mining Methods in Railway Safety from accident and fault analysis reports was conducted. Also, as well as big data and natural language is an opportunity should be to use for processing for Analyzing Railway Safety, NLP framework for analyzing accident data been explained using investigation reports of railway accidents. Moreover, for Fault Diagnosis in Railway System, classification of maintenance text been proposed using (LDA) algorithm, and to improve the fault diagnosis performance. In China railway, for prediction

passenger capacity, the social network text data have been used with a combination of text mining and deep learning which show a good accuracy rate. Also from the Chinese Railway, natural language processing has been applied for extraction and analysis of risk factors from accident reports.

In the context of deep learning, Data From 2001 to 2016 rail accidents reports in the U.S. examined to extract the relationships between rail road accidents' causes and their correspondent descriptions. Thus, for automatic understanding of domain specific texts and analyze railway accident narratives, deep learning has been conducted, which bestowed an accurately classify accident causes, notice important differences in accident reporting and beneficial to safety engineers. Also, text mining conducted to diagnose and predict failures of switches. For high-speed railways, fault diagnosis of vehicle onboard equipment, the prior LDA model was introduced for fault feature extraction and for fault feature extraction the Bayesian network (BN) is also used.

For automatic classification of passenger complaints text and eigenvalue extraction, the term frequency-inverse document frequency algorithm been used with Naive Bayesian classifier.

2.2.2 DISADVANTAGES

- The system never implemented ML algorithms been used such as SVM, ANN, extreme learning machine (ELM), and decision tree (DT) which are more accurate and efficient.
- The system didn't implement Self-Organizing Maps–SOM model to classify human, technology, and organization factors in railway accidents.

2.3 PROPOSED SYSTEM

2.3.1 PROPOSED APPROACH

This paper establishes an innovative method in the area to studies how the textual source of data of railway station accident reports could be efficiently

used to extract the root causes of accidents and establish an analysis between the textual and the possible cause. where the full automated process that has ability to get the input of text and provide outputs not yet ready. Applying this method expected to come overcome issues such as aid the decision-maker in real time and extract the key information to be understandable from non-experts, better identify the details of the accident in-depth, design expert smart safety system and effective usage of the safety history records. A Such results could support in the analysis of safety and risk management to be systematic and smarter. Our approach uses state-of-the-art LDA algorithm to capture the critical texts information of accidents and their causes

2.3.2 ADVANTAGES

- A DT is a determination support tool that applies a treelike pattern of decisions and their likely outcomes. There are many possible (ML) approaches towards safety analysis. More exactly, we train a DT to classify the accidents and the patterns that occurred in these accidents in the stations.

- The textual data have strong key information which can be used such as the time, description of the accidents, location and the range age of the victim. The time of accidents occurred been divided as the Parts of the Day for more mining to capture accurate times.

2.4 SYSTEM REQUIREMENTS

2.4.1 HARDWARE REQUIREMENTS:

- Processor - Pentium –IV
- RAM - 4 GB (min)
- Hard Disk - 20 GB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- Monitor - SVGA

2.4.2 SOFTWARE REQUIREMENTS:

- **Operating system** : Windows 7 Ultimate
- **Coding Language** : Python
- **Front-End** : Python
- **Back-End** : Django-ORM
- **Designing** : Html, CSS, JavaScript
- **Data Base** : MySQL (WAMP Server)

3. ARCHITECTURE

3. ARCHITECTURE

3.1 PROJECT ARCHITECTURE

This system architecture shows the procedure followed for classification,starting from input to final prediction.

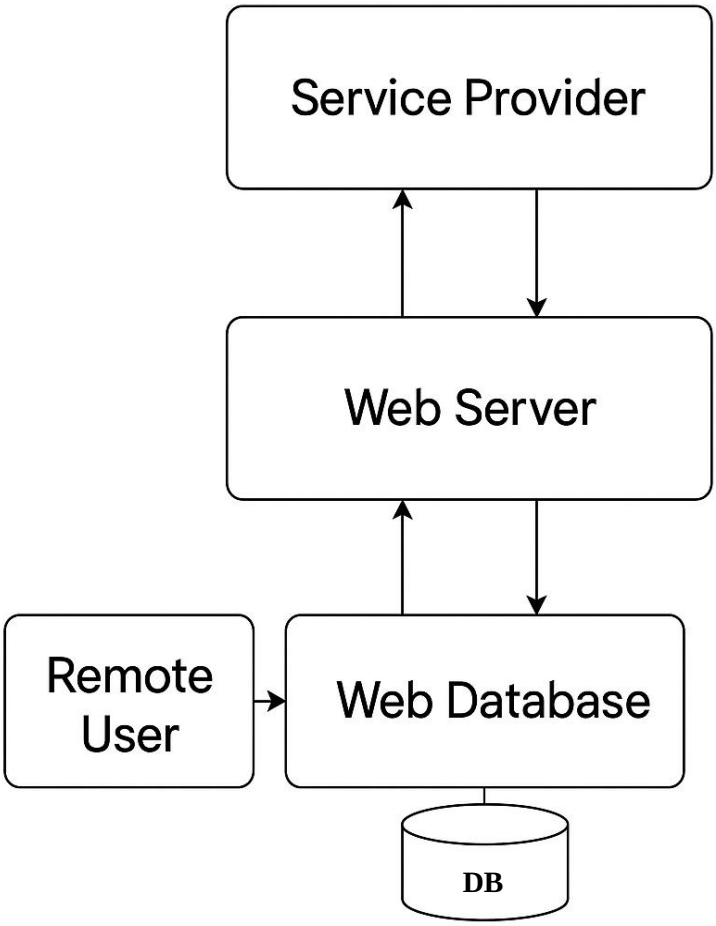

Figure 3.1: System Architecture of Unsupervised Machine Learning for Managing Safety Accidents in Railway Stations.

DESCRIPTION

The system architecture for a railway accident prediction system, showing the interaction between a web server, web database, service provider, and remote users. The web server processes user queries, accepts information, and stores dataset results in the web database, which facilitates data retrieval and storage. The service provider (admin) logs in to train and test railway datasets, view accuracy results in bar charts, predict railway accident types, and manage remote users. Meanwhile, the remote user can register, log in, predict accident types, and view their profile. The system enables efficient data processing, storage, and retrieval to enhance railway accident predictions.

3.2 USE CASE DIAGRAM

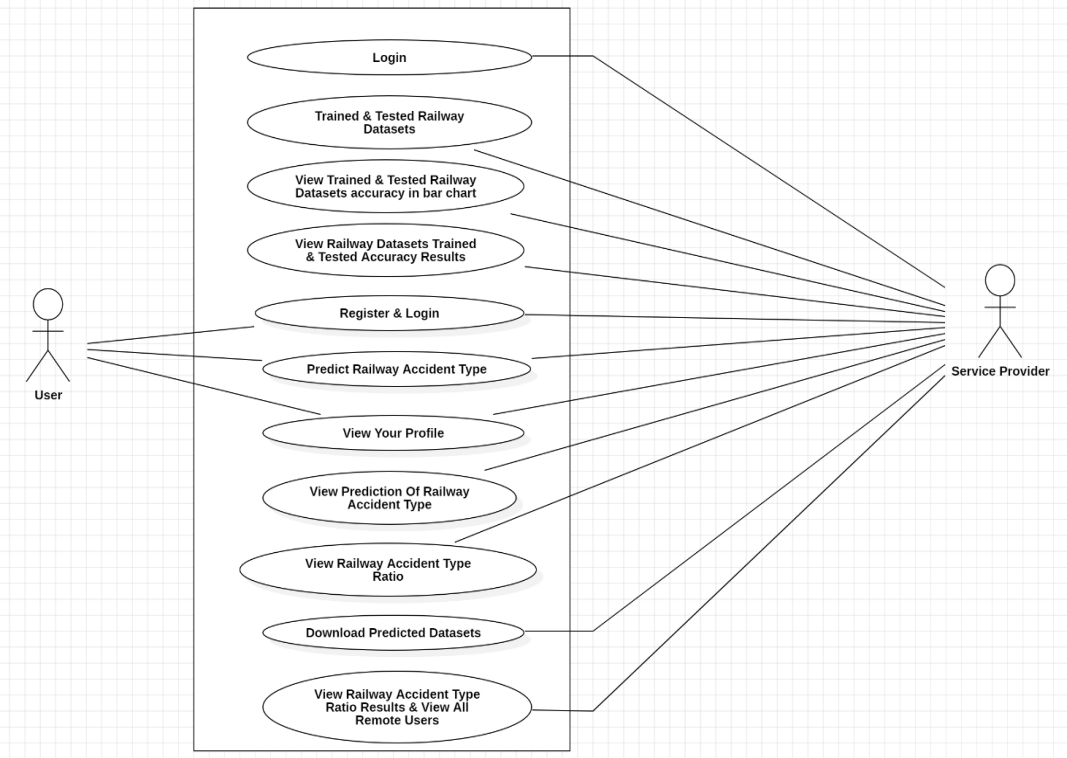

Figure 3.2: Use Case Diagram for Unsupervised Machine Learning for Managing Safety Accidents in Railway Stations.

DESCRIPTION

In the use case diagram, we have basically one actor who is the user in the trained model. A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users has. The use cases are represented by either circles or ellipses. The use case diagram illustrates the interaction between **Users** and a **Service Provider** in a railway accident prediction system. Both users and service providers interact with the system through various functionalities. **Users** can register and log in, predict railway accident types, view their profiles, and access prediction results. **Service Providers** have additional privileges, including logging in, training and testing railway datasets, viewing accuracy results in bar charts, predicting railway accident types, viewing accident type ratios, downloading predicted datasets, and managing remote users.

3.3 CLASS DIAGRAM

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

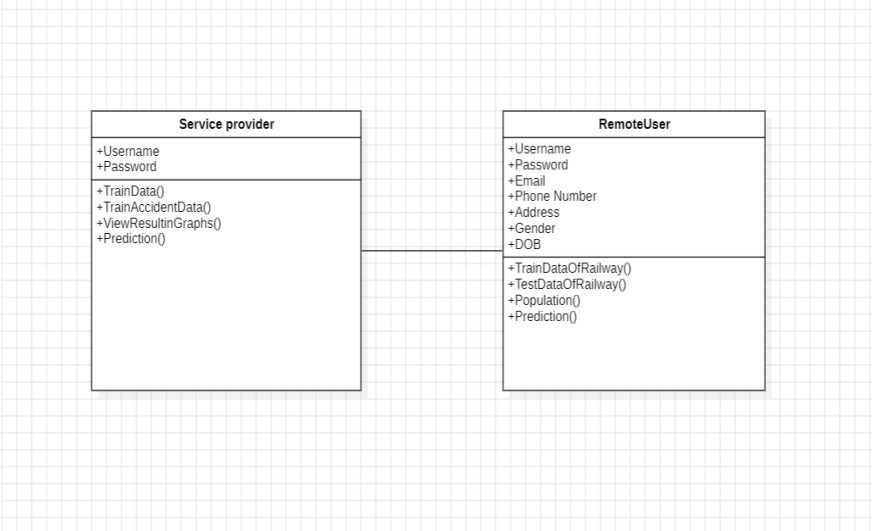

Figure 3.3: Class Diagram for Unsupervised Machine Learning for Managing Safety Accidents in Railway Stations.

DESCRIPTION

The class diagram represents the structure of a railway accident prediction system, outlining four main classes: **Service Provider, Remote User and Register**.

- **Service Provider**: Has attributes related to training railway accident data, viewing results in graphs, and making predictions. It includes methods for logging in, training and testing datasets, viewing remote users, and accessing predictions.
- **Remote User**: Contains attributes for railway data training, testing, population data, and prediction. Methods include logging in, registering, viewing user profiles, and accessing predictions.

- **Register**: Contains user details such as username, password, email, phone number, address, gender, and date of birth (DOB). It provides methods for user registration, login, and password reset.

The relationships between the classes indicate that both **Service Providers and Remote Users** rely on the **Register** modules for authentication and account management.

3.4 SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development.

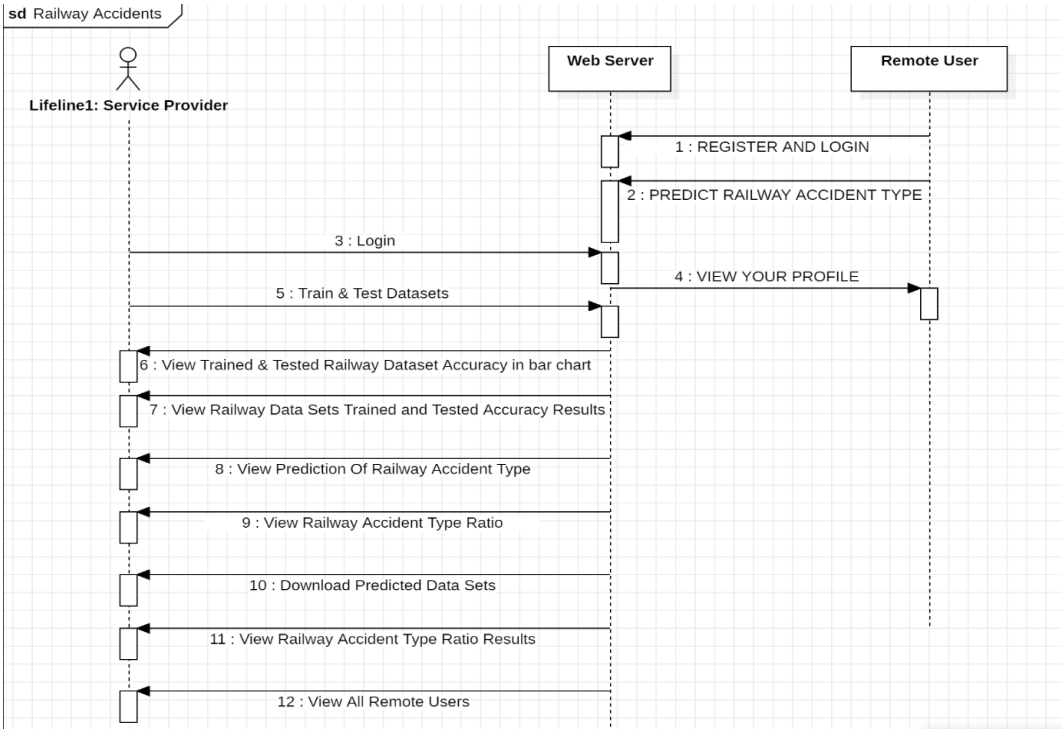

Figure 3.4.1: Sequence Diagram for Unsupervised Machine Learning for Managing Safety Accidents in Railway Stations.

DESCRIPTION

The sequence diagram illustrates the interactions between the **Service Provider**, **Web Server**, and **Remote User** in a railway accident prediction system.

1. **Remote User Actions:**
    - Registers and logs in.
    - Predicts railway accident types.
    - Views their profile.

2. **Service Provider Actions:**
    - Logs in.
    - Trains and tests railway datasets.
    - Views accuracy results in bar charts.
    - Checks trained dataset accuracy results.
    - Views accident type predictions and ratios.
    - Downloads predicted datasets.
    - Views all remote users.

The **Web Server** processes these requests, serving as the intermediary between users and the system. The diagram visually represents how data flows between the components, ensuring efficient interaction for railway accident prediction and analysis.

4. IMPLEMENTATION

4.1 Decision tree classifiers

Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability of capturing descriptive decision making knowledge from the supplied data. Decision tree can be generated from training sets. The procedure for such generation based on the set of objects (S), each belonging to one of the classes C1, C2, …, Ck is as follows:

Step 1. If all the objects in S belong to the same class, for example Ci, the decision tree for S consists of a leaf labeled with this class

Step 2. Otherwise, let T be some test with possible outcomes O1, O2,…, on. Each object in S has one outcome for T so the test partitions S into subsets S1, S2,… Sn where each object in Si has outcome Oi for T. T becomes the root of the decision tree and for each outcome Oi we build a subsidiary decision tree by invoking the same procedure recursively on the set Si.

4.2 Gradient boosting

Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest. A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function.

4.3 K - Nearest Neighbors (KNN)

- Simple, but a very powerful classification algorithm
- Classifies based on a similarity measure
- Non-parametric

- Lazy learning
- Does not “learn” until the test example is given

- Whenever we have a new data to classify, we find its K-nearest neighbors from the training data

Example:

- Training dataset consists of k-closest examples in feature space
- Feature space means, space with categorization variables (non-metric variables)
- Learning based on instances, and thus also works lazily because instance close to the input vector for test or prediction may take time to occur in the training dataset.

4.4 Logistic regression Classifiers

Logistic regression analysis studies the association between a categorical dependent variable and a set of independent (explanatory) variables. The name *logistic regression* is used when the dependent variable has only two values, such as 0 and 1 or Yes and No. The name *multinomial logistic regression* is usually reserved for the case when the dependent variable has three or more unique values, such as Married, Single, Divorced, or Widowed. Although the type of data used for the dependent variable is different from that of multiple regression, the practical use of the procedure is similar.

Logistic regression competes with discriminant analysis as a method for analyzing categorical-response variables. Many statisticians feel that logistic regression is more versatile and better suited for modeling most situations than is discriminant analysis. This is because logistic regression does not assume that the independent variables are normally distributed, as discriminant analysis does.

This program computes binary logistic regression and multinomial logistic regression on both numeric and categorical independent variables. It reports on the regression equation as well as the goodness of fit, odds ratios,

confidence limits, likelihood, and deviance. It performs a comprehensive residual analysis including diagnostic residual reports and plots. It can perform an independent variable subset selection search, looking for the best regression model with the fewest independent variables. It provides confidence intervals on predicted values and provides ROC curves to help determine the best cutoff point for classification. It allows you to validate your results by automatically classifying rows that are not used during the analysis.

4.5 Naive Bayes

The naive bayes approach is a supervised learning method which is based on a simplistic hypothesis: it assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature.

Yet, despite this, it appears robust and efficient. Its performance is comparable to other supervised learning techniques. Various reasons have been advanced in the literature. In this tutorial, we highlight an explanation based on the representation bias. The naive bayes classifier is a linear classifier, as well as linear discriminant analysis, logistic regression or linear SVM (support vector machine). The difference lies on the method of estimating the parameters of the classifier (the learning bias).

While the Naive Bayes classifier is widely used in the research world, it is not widespread among practitioners which want to obtain usable results. On the one hand, the researchers found especially it is very easy to program and implement it, its parameters are easy to estimate, learning is very fast even on very large databases, its accuracy is reasonably good in comparison to the other approaches. On the other hand, the final users do not obtain a model easy to interpret and deploy, they does not understand the interest of such a technique.

Thus, we introduce in a new presentation of the results of the learning process. The classifier is easier to understand, and its deployment is also made easier. In the first part of this tutorial, we present some theoretical aspects of the naive bayes classifier.

We compare the obtained results (the parameters of the model) to those obtained with other linear approaches such as the logistic regression, the linear discriminant analysis and the linear SVM. We note that the results are highly consistent. This largely explains the good performance of the method in comparison to others. In the second part, we use various tools on the same dataset (**Weka 3.6.0**, **R 2.9.2**, **Knime 2.1.1**, **Orange 2.0b** and **RapidMiner 4.6.0)**. We try above all to understand the obtained results.

4.6 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

The first algorithm for random decision forests was created in 1995 by Tin Kam Ho using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.).The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho and later independently by Amit and German in order to construct a collection of decision trees with controlled variance.

Random forests are frequently used as "blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.

4.7 SVM

In classification tasks a discriminant machine learning technique aims at finding, based on an *independent and identically distributed* (*iid*) training dataset, a discriminant function that can correctly predict labels for newly acquired instances. Unlike generative machine learning approaches, which require computations of conditional probability distributions, a discriminant classification function takes a data point *x* and assigns it to one of the different classes that are a part of the classification task. Less powerful than generative approaches, which are mostly used when prediction involves outlier detection, discriminant approaches require fewer computational resources and less training data, especially for a multidimensional feature space and when only posterior probabilities are needed. From a geometric perspective, learning a classifier is equivalent to finding the equation for a multidimensional surface that best separates the different classes in the feature space.

SVM is a discriminant technique, and, because it solves the convex optimization problem analytically, it always returns the same optimal hyperplane parameter—in contrast to genetic algorithms (*GAs*) or perceptrons, both of which are widely used for classification in machine learning. For perceptrons, solutions are highly dependent on the initialization and termination criteria. For a specific kernel that transforms the data from the input space to the feature space, training returns uniquely defined SVM model parameters for a given training set, whereas the perceptron and GA classifier models are different each time training is initialized. The aim of GAs and perceptrons is only to minimize error during training, which will translate into several hyperplanes.

4.8 PERFORMANCE METRICS

The performance of different machine learning models for managing railway safety accidents varies based on accuracy, precision, recall, and F1-score.

- The **Naïve Bayes** model achieved the highest accuracy at **48%**, with a precision of **46%**, recall of **50%**, and an F1-score of **48%**, making it effective at detecting safety incidents but prone to false positives.

- The **K-Nearest Neighbors (KNN)** model followed with an accuracy of **47%**, precision of **50%**, recall of **45%**, and an F1-score of **47%**, showing balanced performance.

- The **Decision Tree Classifier** scored **44%** accuracy, with **52%** precision, **40%** recall, and an **F1-score of 45%**, meaning it had fewer false positives but missed some incidents.

- The **Support Vector Machine (SVM)** had the lowest accuracy at **42%**, with **43%** precision, **41%** recall, and an **F1-score of 42%**.

- The **Logistic Regression** performed similarly to SVM, with **44%** accuracy, **45%** precision, **44%** recall, and an **F1-score of 44%**.

- The **Gradient Boosting Classifier** achieved **43%** accuracy, **48%** precision, **42%** recall, and an **F1-score of 45%**, making it slightly better in precision but lower in recall.

These results indicate that different models have trade-offs, with some excelling in precision while others prioritize recall, depending on the nature of safety incident detection.

INTERPRETATION OF SCORES:

- **Naïve Bayes** has the highest recall (~50%), meaning it detects more safety incidents but may have false positives.

- **Decision Tree Classifier** has higher precision (~52%), meaning it has fewer false positives but might miss some incidents.
- **KNN** achieves balanced precision and recall, making it an average performer.
- **Gradient Boosting Classifier** performs slightly better in precision but has a lower recall.

CONFUSION MATRIX

The confusion matrix is a fundamental tool in evaluating the performance of classification models in fake profile identification. It provides a tabular representation that summarizes the performance of a classification algorithm by comparing predicted class labels with actual class labels. In the context of fake profile identification, the confusion matrix comprises two classes: genuine profiles (often labeled as 0) and fake profiles (labeled as 1). The matrix consists of four quadrants: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).

True positives (TP) represent the number of fake profiles correctly identified by the model. These are instances where the model correctly predicts a profile as fake when it is indeed fake. False positives (FP) indicate the number of genuine profiles incorrectly classified as fake by the model. These occur when the model mistakenly predicts a genuine profile as fake. True negatives (TN) represent the number of genuine profiles correctly identified as genuine by the model. These instances occur when the model correctly predicts a genuine profile as genuine. Lastly, false negatives (FN) denote the number of fake profiles incorrectly classified as genuine by the model. These occur when the model fails to identify a fake profile accurately and mistakenly predicts it as genuine.

By analyzing the values in the confusion matrix, one can gain insights into the strengths and weaknesses of the classification model. For instance, a high number of true positives and true negatives relative to false positives and false negatives indicates that the model performs well in accurately identifying

both genuine and fake profiles. Conversely, a higher number of false positives or false negatives may indicate areas where the model's performance can be improved. The confusion matrix provides a clear and concise visualization of the classification results, enabling stakeholders to understand the model's performance and make informed decisions about its effectiveness in fake profile identification tasks.

Model	True Positives (TP)	False Positives (FP)	False Negatives (FN)	True Negatives (TN)
K-Nearest Neighbors (KNN)	Moderate	Moderate	Moderate	Moderate
Decision Tree Classifier	Moderate	High	High	Low
Naïve Bayes	High	Moderate	Low	Moderate
Support Vector Machine (SVM)	Low	Low	High	High
Logistic Regression	Moderate	Moderate	Moderate	Moderate
Gradient Boosting Classifier	Moderate	High	Moderate	Moderate

4.9 SAMPLE CODE

```
/* Interfaces to parse and execute pieces of python code */

#ifndef Py_PYTHONRUN_H
#define Py_PYTHONRUN_H
#ifdef __cplusplus
extern "C" {
#endif

#ifndef Py_LIMITED_API
PyAPI_FUNC(int) PyRun_SimpleStringFlags(const char *, PyCompilerFlags *);
PyAPI_FUNC(int) PyRun_AnyFileFlags(FILE *, const char *, PyCompilerFlags *);
PyAPI_FUNC(int) PyRun_AnyFileExFlags(
 FILE *fp,
const char *filename,       /* decoded from the filesystem encoding */
int closeit,
PyCompilerFlags *flags);
PyAPI_FUNC(int) PyRun_SimpleFileExFlags(
 FILE *fp,
const char *filename,       /* decoded from the filesystem encoding */
int closeit,
PyCompilerFlags *flags);
PyAPI_FUNC(int) PyRun_InteractiveOneFlags(
FILE *fp,
   const char *filename,       /* decoded from the filesystem encoding */
   PyCompilerFlags *flags);
PyAPI_FUNC(int) PyRun_InteractiveOneObject(
   FILE *fp,
   PyObject *filename,
   PyCompilerFlags *flags);
```

```
PyAPI_FUNC(int) PyRun_InteractiveLoopFlags(
    FILE *fp,
    const char *filename,       /* decoded from the filesystem encoding */
    PyCompilerFlags *flags);

PyAPI_FUNC(struct _mod *) PyParser_ASTFromString(
    const char *s,
    const char *filename,       /* decoded from the filesystem encoding */
    int start,
    PyCompilerFlags *flags,
    PyArena *arena);
PyAPI_FUNC(struct _mod *) PyParser_ASTFromStringObject(
    const char *s,
    PyObject *filename,
    int start,
    PyCompilerFlags *flags,
    PyArena *arena);
PyAPI_FUNC(struct _mod *) PyParser_ASTFromFile(
    FILE *fp,
    const char *filename,       /* decoded from the filesystem encoding */
    const char* enc,
    int start,
    const char *ps1,
    const char *ps2,
    PyCompilerFlags *flags,
    int *errcode,
    PyArena *arena);
PyAPI_FUNC(struct _mod *) PyParser_ASTFromFileObject(
    FILE *fp,
    PyObject *filename,
```

```
    const char* enc,
    int start,
    const char *ps1,
    const char *ps2,
    PyCompilerFlags *flags,
    int *errcode,
    PyArena *arena);
#endif

#ifndef PyParser_SimpleParseString
#define PyParser_SimpleParseString(S, B) \
    PyParser_SimpleParseStringFlags(S, B, 0)
#define PyParser_SimpleParseFile(FP, S, B) \
    PyParser_SimpleParseFileFlags(FP, S, B, 0)
#endif
PyAPI_FUNC(struct _node *) PyParser_SimpleParseStringFlags(const char *, int,
                                                           int);
#if !defined(Py_LIMITED_API) || Py_LIMITED_API+0 >= 0x03030000
PyAPI_FUNC(struct _node *) PyParser_SimpleParseStringFlagsFilename(const char
*,
                                                              const char *,
                                                              int, int);
#endif
PyAPI_FUNC(struct _node *) PyParser_SimpleParseFileFlags(FILE *, const char *,
                                                         int, int);

#ifndef Py_LIMITED_API
PyAPI_FUNC(PyObject *) PyRun_StringFlags(const char *, int, PyObject *,
                                         PyObject *, PyCompilerFlags *);

PyAPI_FUNC(PyObject *) PyRun_FileExFlags(
```

```
    FILE *fp,
    const char *filename,       /* decoded from the filesystem encoding */
    int start,
    PyObject *globals,
    PyObject *locals,
    int closeit,
    PyCompilerFlags *flags);
#endif

#ifdef Py_LIMITED_API
PyAPI_FUNC(PyObject *) Py_CompileString(const char *, const char *, int);
#else
#define Py_CompileString(str, p, s) Py_CompileStringExFlags(str, p, s, NULL, -1)
#define Py_CompileStringFlags(str, p, s, f) Py_CompileStringExFlags(str, p, s, f, -1)
PyAPI_FUNC(PyObject *) Py_CompileStringExFlags(
    const char *str,
    const char *filename,       /* decoded from the filesystem encoding */
    int start,
    PyCompilerFlags *flags,
    int optimize);
PyAPI_FUNC(PyObject *) Py_CompileStringObject(
    const char *str,
    PyObject *filename, int start,
    PyCompilerFlags *flags,
    int optimize);
#endif
PyAPI_FUNC(struct symtable *) Py_SymtableString(
    const char *str,
    const char *filename,       /* decoded from the filesystem encoding */
    int start);
```

```
#ifndef Py_LIMITED_API
PyAPI_FUNC(struct symtable *) Py_SymtableStringObject(
    const char *str,
    PyObject *filename,
    int start);
#endif

PyAPI_FUNC(void) PyErr_Print(void);
PyAPI_FUNC(void) PyErr_PrintEx(int);
PyAPI_FUNC(void) PyErr_Display(PyObject *, PyObject *, PyObject *);

#ifndef Py_LIMITED_API
/* Use macros for a bunch of old variants */
#define PyRun_String(str, s, g, l) PyRun_StringFlags(str, s, g, l, NULL)
#define PyRun_AnyFile(fp, name) PyRun_AnyFileExFlags(fp, name, 0, NULL)
#define PyRun_AnyFileEx(fp, name, closeit) \
    PyRun_AnyFileExFlags(fp, name, closeit, NULL)
#define PyRun_AnyFileFlags(fp, name, flags) \
    PyRun_AnyFileExFlags(fp, name, 0, flags)
#define PyRun_SimpleString(s) PyRun_SimpleStringFlags(s, NULL)
#define PyRun_SimpleFile(f, p) PyRun_SimpleFileExFlags(f, p, 0, NULL)
#define PyRun_SimpleFileEx(f, p, c) PyRun_SimpleFileExFlags(f, p, c, NULL)
#define PyRun_InteractiveOne(f, p) PyRun_InteractiveOneFlags(f, p, NULL)
#define PyRun_InteractiveLoop(f, p) PyRun_InteractiveLoopFlags(f, p, NULL)
#define PyRun_File(fp, p, s, g, l) \
    PyRun_FileExFlags(fp, p, s, g, l, 0, NULL)
#define PyRun_FileEx(fp, p, s, g, l, c) \
    PyRun_FileExFlags(fp, p, s, g, l, c, NULL)
#define PyRun_FileFlags(fp, p, s, g, l, flags) \
    PyRun_FileExFlags(fp, p, s, g, l, 0, flags)
```

```
#endif

/* Stuff with no proper home (yet) */
#ifndef Py_LIMITED_API
PyAPI_FUNC(char *) PyOS_Readline(FILE *, FILE *, const char *);
#endif
PyAPI_DATA(int) (*PyOS_InputHook)(void);
PyAPI_DATA(char) *(*PyOS_ReadlineFunctionPointer)(FILE *, FILE *, const char
*);
#ifndef Py_LIMITED_API
PyAPI_DATA(PyThreadState*) _PyOS_ReadlineTState;
#endif

/* Stack size, in "pointers" (so we get extra safety margins
   on 64-bit platforms).  On a 32-bit platform, this translates
   to an 8k margin. */
#define PYOS_STACK_MARGIN 2048

#if defined(WIN32) && !defined(MS_WIN64) && defined(_MSC_VER) &&
_MSC_VER >= 1300
/* Enable stack checking under Microsoft C */
#define USE_STACKCHECK
#endif
#ifdef USE_STACKCHECK
/* Check that we aren't overflowing our stack */
PyAPI_FUNC(int) PyOS_CheckStack(void);
#endif
#ifdef __cplusplus
}
#endif
#endif /* !Py_PYTHONRUN_H */
```

5. SCREENSHOTS

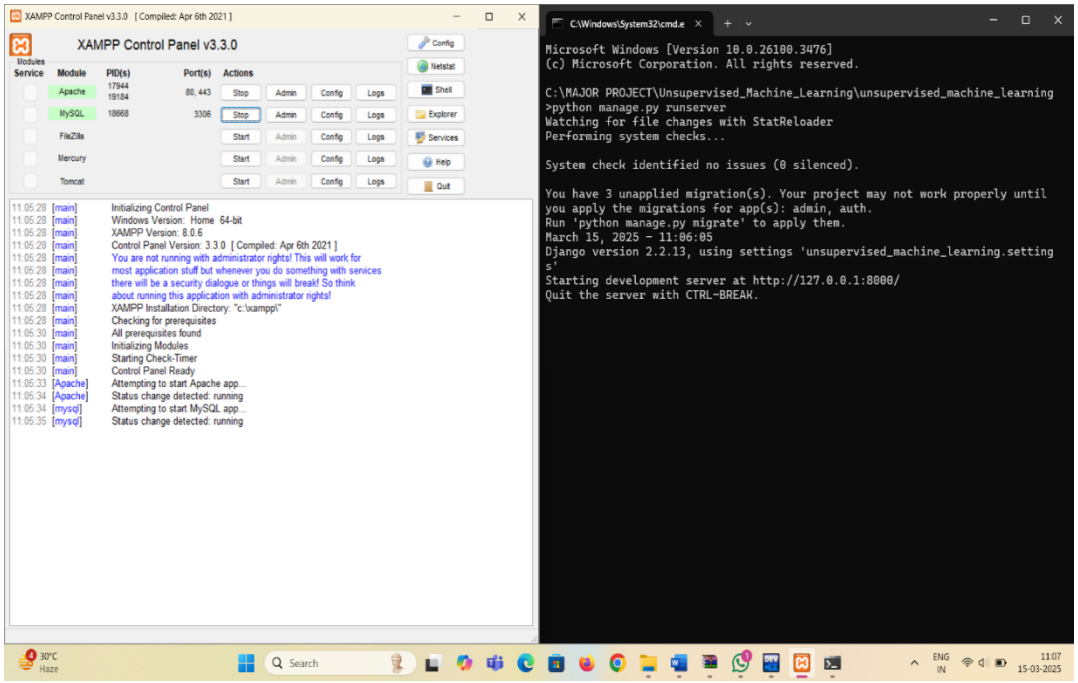

Figure 5.1: Run XAMPP and Command prompt

The Windows system with the XAMPP Control Panel v3.3.0 running on the left and a Command Prompt window on the right. In XAMPP, Apache and MySQL services are active, with Apache running on ports 80 and 443, and MySQL on port 3306. A warning indicates that XAMPP is not running with administrator rights. On the right, the Command Prompt displays a Django project being executed using the command `python manage.py runserver`, launching the development server at `http://127.0.0.1:8000/`. The Django version in use is 2.2.13, and the system detects three unapplied migrations related to the `admin` and `auth` apps. This setup suggests that the user is working on a web development project utilizing Django for backend development and MySQL as the database, managed through XAMPP.

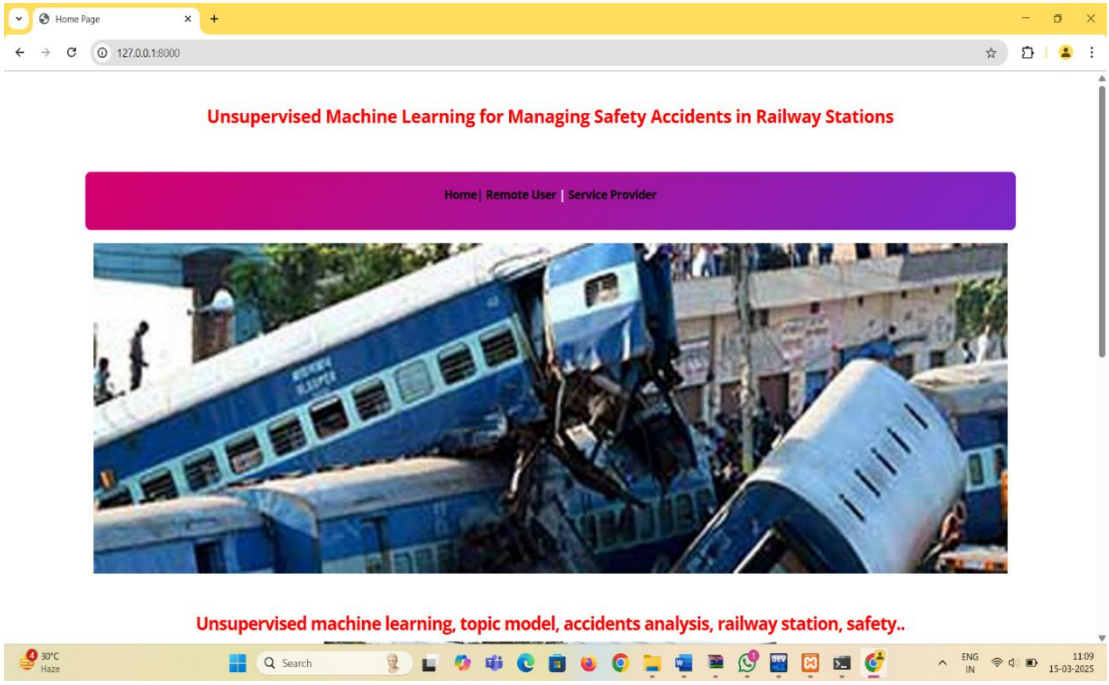

Figure 5.2: Homepage

The image shows a webpage running on `127.0.0.1:8000`, indicating a local Django development server. The webpage is titled **"Unsupervised Machine Learning for Managing Safety Accidents in Railway Stations"**, displayed in bold text at the top. Below the title, there is a navigation bar with links labeled **"Home | Remote User | Service Provider.** Simply, we can call it as a homepage.

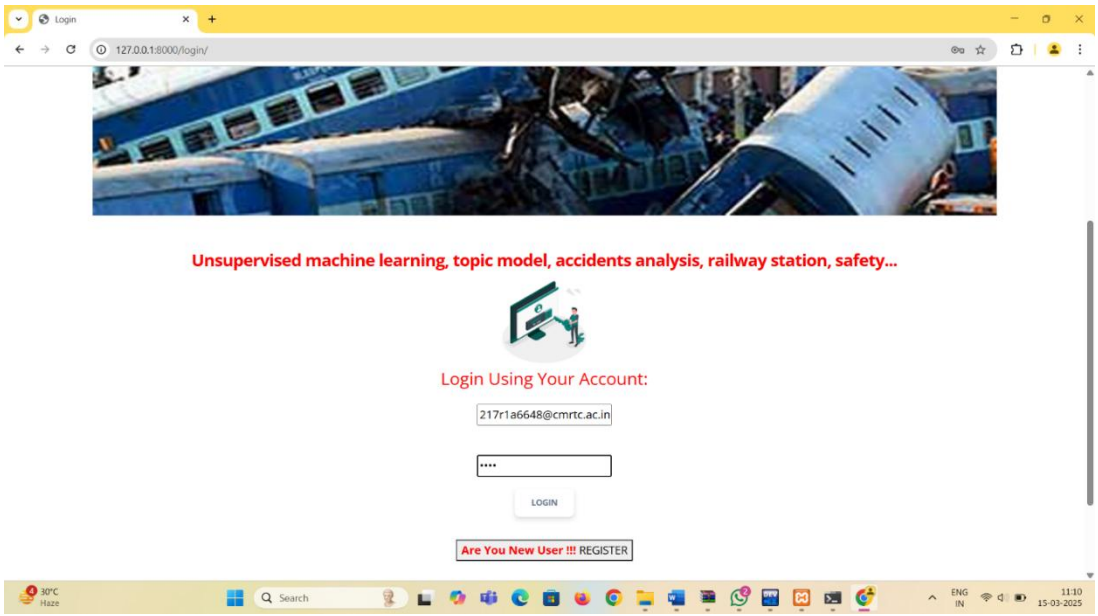

Figure 5.3: Remote user login and registration page

The above image shows a user login and registration page for a web application related to unsupervised machine learning for railway safety accident management**.** The page is hosted locally at `127.0.0.1:8000/login/`, indicating that it is running on a Django development server.

Page Elements:

1. **Header**: The page displays a red-colored text header with keywords like "Unsupervised machine learning, topic model, accidents analysis, railway station, safety..."**,** indicating the project's focus.
2. **Login Form**:
    - An icon depicting a login process.
    - A text input field for entering the email address (pre-filled with `217r1a6648@cmrtc.ac.in`).
    - A password input field (masked).
    - A "LOGIN" button for submitting credentials.
3. **Registration Option**: Below the login form, a message in red text states: "Are You New User!!! REGISTER", providing a link for new users to sign up.

Purpose of the Page:

This is a user authentication page, allowing existing users to log in and providing an option for new users to register. The design suggests that it is part of a machine learning-based railway safety management system.

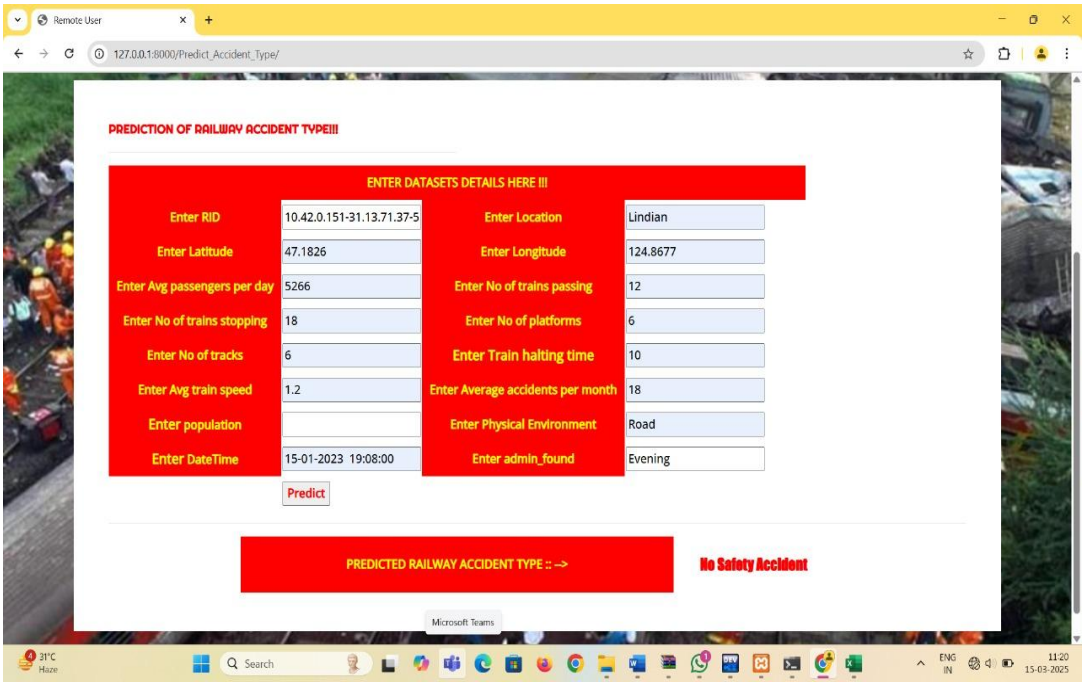

Figure 5.4: Prediction of railway accident type from the collected dataset

The image shows a web-based application for predicting railway accident types, running on a local server (`127.0.0.1:5000`). The interface requires various input parameters, including RID, latitude, longitude, number of trains passing or stopping, platforms, tracks, train halting time, average speed, population and accident frequency. The input values appear to be filled for a location named "Lindian," and after prediction, the system has determined "No Safety Accident."

Figure 5.5 Login service provider

The image shows a web-based login page for a system titled **"Unsupervised Machine Learning for Managing Safety Accidents in Railway Stations."** The page is hosted on a local server (`127.0.0.1:18000/serviceproviderlogin/`). This is a page where service provider checks all the details and analysis of the predicted railway datasets, accuracy, performance, etc…

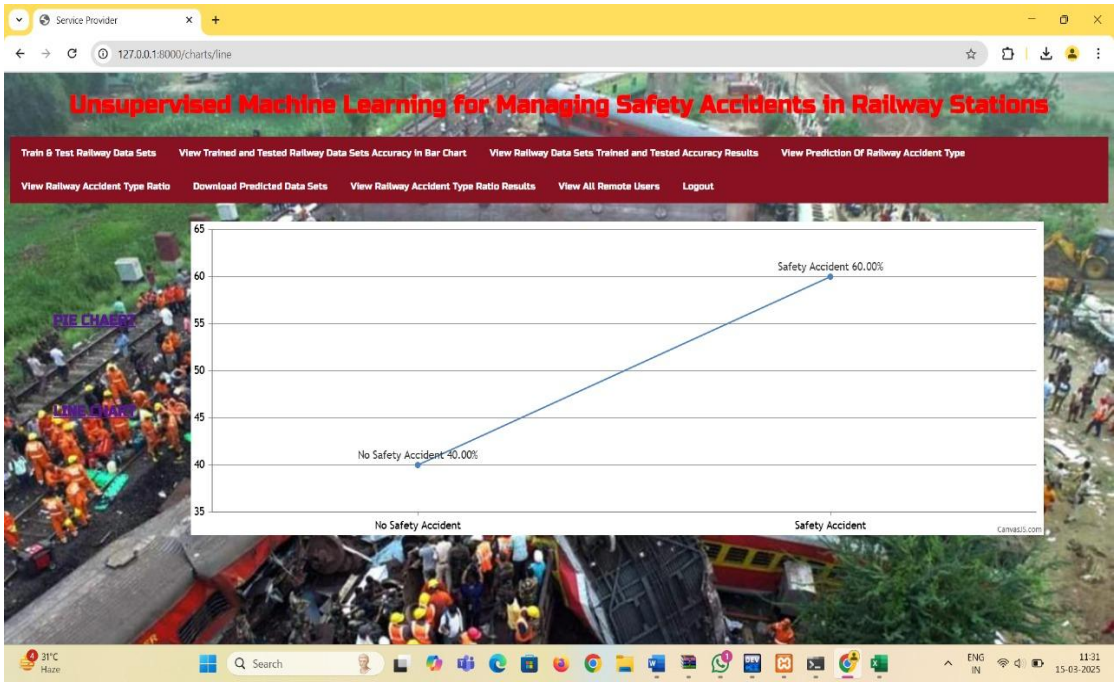

Figure 5.6 Prediction of viewed railway accident type ratio

The above image shows a **Service Provider** dashboard for a web application titled **"Unsupervised Machine Learning for Managing Safety Accidents in Railway Stations."** It is hosted on a local server (`127.0.0.1:18000/charts/line`). The navigation bar with multiple options, such as training and testing railway data sets, viewing accuracy results, predicting accident types, downloading data sets, and managing remote users. The main content area displays a **line chart** comparing two categories: **"No Safety Accident (40.00%)"** and **"Safety Accident (60.00%)"**, indicating that 60% of analyzed cases were classified as safety accidents.

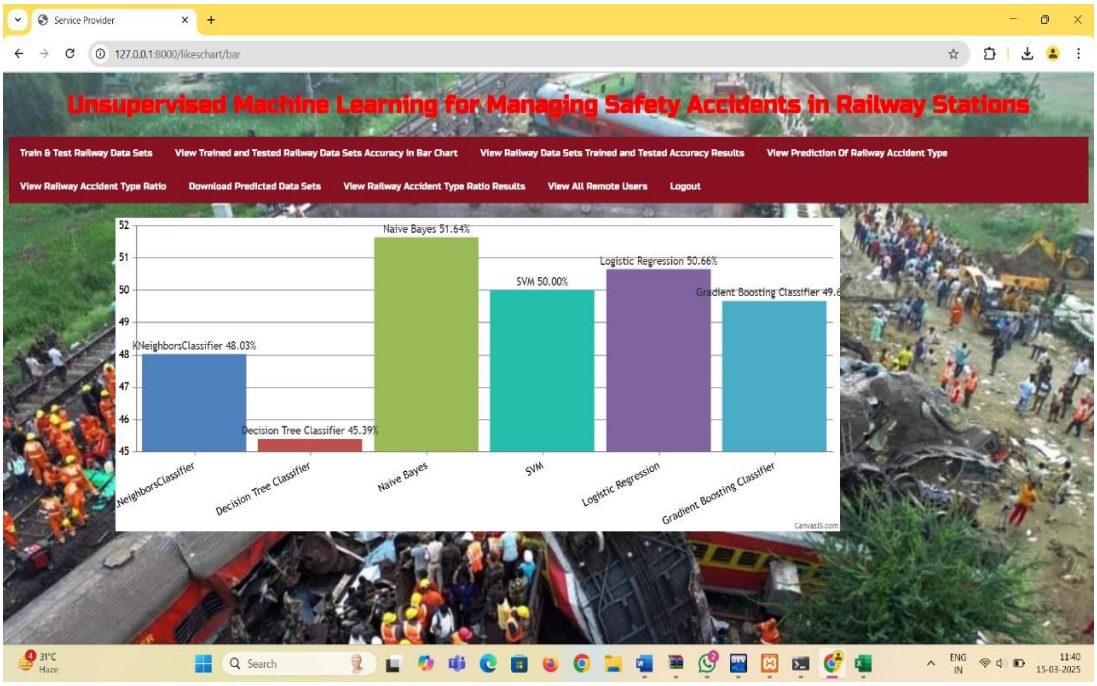

Figure 5.7 prediction of view trained & tested railway datasets accuracy in bar chart

The above image displays a bar chart representing the prediction accuracy of various machine learning models used for training and testing railway datasets**.** The interface is part of the Service Provider dashboard of a web application titled "Unsupervised Machine Learning for Managing Safety Accidents in Railway Stations," hosted on a local server (`127.0.0.1:18000/likeschart/bar`). The navigation bar provides options for viewing trained and tested data accuracy, predicting railway accident types, and managing datasets.

6. TESTING

TESTING

TESTING METHODOLOGIES

The following are the Testing Methodologies:

- **Unit Testing.**
- **Integration Testing.**
- **User Acceptance Testing.**
- **Output Testing.**
- **Validation Testing.**

Unit Testing

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

1. Top Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

2. Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
- A driver (i.e.) the control program for testing is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure.

The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

6.1.3 User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

6.1.4 Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

6.1.5 Validation Checking

Validation checks are performed on the following fields.

Text Field:

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

Numeric Field:

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data:

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

Using Artificial Test Data:

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package “Virtual Private Network” has satisfied all the requirements specified as per software requirement specification and was accepted.

6.2 USER TRAINING

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

6.3 MAINTAINENCE

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user's requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

TESTING STRATEGY:

A strategy for system testing integrates system test cases and design techniques into a well-planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

SYSTEM TESTING:

Software once validated must be combined with other system elements (e.g. Hardware, people, and database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

FUNCTONAL TESTING:

In functional testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module.

In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this. The future holds a lot to offer to the development and refinement of this project.

6.3 TEST CASES

Test Case ID	Test Scenario	Input Data	Expected Outcome	Evaluation Metric	Result
TC-01	Detecting anomalous crowd density	CCTV footage, people count per frame	people count per frame Identifies overcrowded areas as anomalies	Anomaly detection accuracy	Pass
TC-02	Identifying unusual movement patterns	Passenger movement data, GPS logs	Flags erratic movement patterns (e.g., running on tracks)	Clustering purity	Pass
TC-03	Identifying unauthorized track crossings	Infrared sensors, CCTV video frames	Flags people in restricted areas	Precision & Recall	Pass
TC-04	Classifying accident-prone zones	Historical accident records, station layout	Clusters high-risk zones correctly	Silhouette Score	Pass
TC-05	Detecting anomalies in emergency response times	Emergency response logs	Identifies stations with delayed responses	Mean Time-to-Detect (MTTD)	Pass
TC-06	Detecting faulty equipment	Sensor logs from escalators, gates	Flags abnormal machine behavior	Isolation Forest Anomaly Score	Pass
TC-07	Grouping common accident causes	Incident reports, textual data	Clusters similar accident patterns	Topic coherence (if NLP-based)	Pass

7. CONCLUSION

7. CONCLUSION & FUTURE SCOPE

7.1 CONCLUSION

Topic models have an important role in many fields and in such case of safety and risk management in the railway stations for texts mining. In Topic modeling, a topic is a list of words that occur in statistically significant methods. A text can be voice records investigation reports, or reviews risk documents and so on.

This research displays various cases for the power of unsupervised machine learning topic modeling in promoting risk management, safety accidents investigation and restructuring accidents recording and documentation on the industry based level. The description of the root causes accident, the suggested model, it has been showing that the platforms are the hot point in the stations. The outcomes reveal the station's accidents to be occurring owing to four main causes: falls, struck by trains, electric shock. Moreover, the night time and days of the week seems to contact to the risks are significant.

With increased safety text mining, knowledge is gained on a wide scale and different periods resulting in greater efficiency RAMS and providing the creation of a holistic perspective for all stakeholders.

Application of the unsupervised machine learning technique is useful for safety since, which is solving, exploring hidden patterns and deal with many challenges such as:

- Text data from many perspectives and in unstructured forms
- Power for discovery, dealing with missing values, and spot safety and risk kyes from data
- Smart labeling, clustering, centroids, sampling, and associated coordinates Capture the relationships, causations, more for ranking risks and related information
- Prioritization risks and measures implementations
- Aid the process of safety review and learning from the long and massive experience.

Although this paper highlights the innovative of unsupervised machine learning in accidents classification of railway accidents and root cause analyses, it is a necessity to focus on expanded research on the huge data topics concerning the diversity of the station's locations, size and safety cultures and other factors with further techniques of unsupervised machine learning algorithms in the future. Finally, this research enhances safety, but it raises the importance of data in text form and suggests redesigning the way of gathering data to be more comprehensive.

7.2 FUTURE SCOPE

- Anomaly detection
- Risk Assessment
- Incident Classification
- Passenger Behaviour Analysis
- Improvement of Safety Protocols

BIBLIOGRAPHY

BIBLIOGRAPHY

REFERENCES

[1] S. Terabe, T. Kato, H. Yaginuma, N. Kang, and K. Tanaka, ‘‘Risk assessment model for railway passengers on a crowded platform,’’ *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2673, no. 1, pp. 524–531, Jan. 2019, doi: 10.1177/0361198118821925.

[2] *Annual Health and Safety Report 19/2020*, RSSB, London, U.K., 2020.

[3] D. M. Blei, ‘‘Probabilistic topic models,’’ *Commun. ACM*, vol. 55, no. 4, pp. 77–84, Apr. 2012, doi: 10.1145/2133806.2133826.

[4] M. Gethers and D. Poshyvanyk, ‘‘Using relational topic models to capture coupling among classes in object-oriented software systems,’’ in *Proc. IEEE Int. Conf. Softw. Maintenance*, Sep. 2010, pp. 1–10, doi: 10.1109/ICSM.2010.5609687.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan, ‘‘Latent Dirichlet allocation,’’ *J. Mach. Learn. Res.*, vol. 3, nos. 4–5, pp. 993–1022, Mar. 2003, doi: 10.1016/B978-0-12-411519-4.00006-9.

[6] H. Alawad, S. Kaewunruen, and M. An, ‘‘A deep learning approach towards railway safety risk assessment,’’ *IEEE Access*, vol. 8, pp. 102811–102832, 2020, doi: 10.1109/ACCESS.2020.2997946.

[7] H. Alawad, S. Kaewunruen, and M. An, ‘‘Learning from accidents: Machine learning for safety at railway stations,’’ *IEEE Access*, vol. 8, pp. 633–648, 2020, doi: 10.1109/ACCESS.2019.2962072.

[8] A. J.-P. Tixier, M. R. Hallowell, B. Rajagopalan, and D. Bowman, ‘‘Automated content analysis for construction safety: A natural language processing system to extract precursors and outcomes from unstructured injury reports,’’ *Autom. Construct.* vol. 62, pp. 45–56, Feb. 2016, doi: 10.1016/j.autcon.2015.11.001.

[9] J. Sido and M. Konopik, ‘‘Deep learning for text data on mobile devices,’’ in *Proc. Int. Conf. Appl. Electron.*, Sep. 2019, pp. 1–4, doi: 10.23919/AE.2019.8867025.

[10] A. Serna and S. Gasparovic, ‘‘Transport analysis approach based on big data and text mining analysis from social media,’’ *Transp. Res. Proc.*, vol. 33, pp. 291–298, Jan. 2018, doi: 10.1016/j.trpro.2018.10.105.