LEVERAGING OPERATIONAL AND ENVIRONMENTAL DATA FOR TRAIN DELAY PREDICTION VIA DEEP LEARNING MODELS

A Project Report Submitted in the Partial Fulfillment of
The Requirements for The Award of The Degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

KOTHA LAHARI (22471A05N0)

APPALA CHANDANA PRIYA (22471A05L2)

YARROJU REKHA SRI (23475A0502)

Under the esteemed guidance of

Dr.E.Rama Krishna, M.Tech.,Ph.D.

Associate Professor

DEPARTMENTOF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETAENGINEERING COLLEGE: NARASAROPETA

(AUTONOMOUS)

Accredited by NAAC with A+ Grade and NBA under

Tire -1 an ISO 9001:2015 Certified

Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada

KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, 522601

2025-2026

NARASARAOPETA ENGINEERING COLLEGE

(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project that is entitled with the name "LEVERAGING OPERATIONAL AND ENVIRONMENTAL DATA FOR TRAIN DELAY PREDICTION VIA DEEP LEARNING MODELS" is Bonafide work done by the team KOTHA LAHARI (22471A05N0), APPALA CHANDANA PRIYA (22471A05L2), YARROJU REKHA SRI (23475A0502) in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in the Department of **COMPUTER SCIENCE AND ENGINEERING** during 2025-2026.

PROJECT GUIDE

Dr.Rama Krishna Eluri M.Tech,Ph.D

Associate Professor

PROJECT CO-ORDINATOR

Syed Rizwana, B.Tech., M.Tech., (Ph. D).

Associate Professor

HEAD OF THE DEPARMENT

Dr.S. N. Tirumala Rao, M.Tech., Ph.D.

Professor & HOD

EXTERNAL EXAMINER

DECLARATION

We declare that this project work titled "**LEVERAGING OPERATIONAL AND ENVIRONMENTAL DATA FOR TRAIN DELAY PREDICTION VIA DEEP LEARNING MODELS**" is composed by ourselves that the work contains here is our own except where explicitly stated otherwise in the text and that this work had not been submitted for any degree or professional qualification except as specified.

KOTHA LAHARI(24471A05N0)
APPALA CHANDANA PRIYA(22471A05L2)
YARROJU REKHA SRI(23475A0502)

ACKNOWLEDGEMENT

We wish to express our thanks to various personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman**, Sri M. V. Koteswara Rao, B.Sc.,** who took keen interest in us in every effort throughout this course. We owe our sincere gratitude to our beloved principal**, Dr. S. Venkateswarlu, Ph.D.,** for showing his kind attention and valuable guidance throughout the course.

We express our deep-felt gratitude towards **Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,** HOD of the CSE department, and also to our guide**, Dr.E.Rama Krishna M.Tech, Ph.D,** of the CSE department, whose valuable guidance and unstinting encouragement enabled us to accomplish our project successfully in time.

We extend our sincere thanks to **Syed Rizwana, B.Tech., M.Tech.,(ph.D),** Associate Professor & Project Coordinator of the project, for extending her encouragement. Their profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

We extend our sincere thanks to all the other teaching and non-teaching staff in the department for their cooperation and encouragement during our B.Tech. degree. We have no words to acknowledge the warm affection, constant inspiration, and encouragement that we received from our parents. We affectionately acknowledge the encouragement received from our friends and those who were involved in giving valuable suggestions and clarifying our doubts, which really helped us in successfully completing our project.

By

KOTHA LAHARI(24471A05N0)

APPALA CHANDANA PRIYA(22471A05L2)

YARROJU REKHA SRI(23475A0502)

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community.

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To become a center of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertize in modern software tools and technologies to cater to the real time requirements of the Industry.

M3: Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.

NARASARAOPETA
NEC ENGINEERING COLLEGE
(AUTONOMOUS)

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Educational Objectives (PEO's)

The graduates of the programme are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems related to the academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Outcomes:

PO1: Engineering Knowledge: Apply knowledge of mathematics, natural science,computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)

PO3: Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)

PO4: Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions.
(WK8).

PO5: Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

PO6: The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).

PO7: Ethics: Apply ethical principles and commit to professional ethics, human values,

diversity and inclusion; adhere to national & international laws. (WK9)

PO8: Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9: Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

PO10: Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11: Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Project Course Outcomes (CO'S):

CO421.1: Analyse the System of Examinations and identify the problem.

CO421.2: Identify and classify the requirements.

CO421.3: Review the Related Literature

CO421.4: Design and Modularize the project

CO421.5: Construct, Integrate, Test and Implement the Project.

CO421.6: Prepare the project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3
C421.1		✓										✓		
C421.2	✓		✓		✓							✓		
C421.3				✓		✓	✓	✓				✓		
C421.4			✓			✓	✓	✓				✓	✓	
C421.5					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Course Outcomes – Program Outcome correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3
C421.1	2	3										2		
C421.2			2		3							2		
C421.3				2		2	3	3				2		
C421.4			2			1	1	2				3	2	
C421.5					3	3	3	2	3	2	2	3	2	1

Note: The values in the above table represent the level of correlation between CO's and PO's:

1. Low level
2. Medium level
3. High level

Project mapping with various courses of Curriculum Attained POs:

Name of the course from which principles are applied in this project	Description of the device	Attained PO
C2204.2, C22L3.2	The project starts by collecting and analyzing operational and environmental train data to identify key delay factors and frame the problem for deep learning modeling.	PO1, PO3
CC421.1, C2204.3, C22L3.2	Each dataset attribute was analyzed and preprocessing like missing value handling, encoding, and scaling was applied to enable efficient model training.	PO2, PO3
CC421.2, C2204.2, C22L3.3	Four models—DNN, CNN, LSTM and BiLSTM—were developed trained, and tested. Evaluation metrics such as RMSE, MAE, and R^2 were used to validate prediction accuracy and consistency.	PO3, PO5, PO9
CC421.3, C2204.3, C22L3.2	The project workflow was modularized, integrating dataset preprocessing, deep learning model training, evaluation, and comparison for effective delay prediction.	PO1, PO5
CC421.4, C2204.4, C22L3.2	The team collaboratively prepared documentation covering preprocessing, model design, evaluation metrics, and performance analysis.	PO10
CC421.5, C2204.2, C22L3.3	The BiLSTM model produced the most accurate delay predictions, proving effective in learning bidirectional temporal dependencies for sequential railway data.	PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Future work includes real-time API integration, advanced attention-based models, and deployment on a public intelligent transport dashboard.	PO4, PO7

ABSTRACT

This project presents a deep learning approach for predicting train delays by integrating operational and environmental factors that influence railway performance. The proposed system evaluates four models — Deep Neural Network (DNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Bidirectional LSTM (BiLSTM). Each model is trained on a structured dataset that includes attributes such as distance between stations, weather conditions, day of the week, time slot, train type, historical delays, and route congestion levels. Data preprocessing involves cleaning, encoding categorical variables, and normalization to ensure model accuracy. Among all models, BiLSTM performs the best, achieving superior results across evaluation metrics such as Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Coefficient of Determination (R^2), and Symmetric Mean Absolute Percentage Error (sMAPE). This demonstrates BiLSTM's ability to capture both forward and backward temporal dependencies in sequential data, making it highly effective for real-time train delay forecasting. The model is optimized using the Adam optimizer and trained over multiple epochs to ensure stability and precision. Extensive experiments show that the BiLSTM model consistently outperforms others with an R^2 of 96.17%, confirming its robustness in handling dynamic railway delay patterns. Furthermore, the system's modular architecture supports real-time prediction, enabling integration into smart transport systems for proactive scheduling and passenger information. Future work can extend this framework using transformer-based architectures or real-time sensor data for even more accurate delay management across large-scale railway networks.

INDEX

LIST OF FIGURES

1. INTRODUCTION

Railway transportation plays a vital role in modern society by supporting large-scale passenger mobility and freight movement in an economical and environmentally sustainable manner. However, with rapid urbanization and increasing demand for rail services, railway networks are facing significant operational challenges, among which train delays remain one of the most critical issues [1]. Train delays negatively impact passenger satisfaction, disrupt planned schedules, increase congestion, and lead to higher operational and maintenance costs for railway authorities [2]. As railway systems grow more complex and interconnected, efficient delay management has become a crucial requirement for intelligent transportation systems [3].

Train delays occur due to multiple interrelated operational and environmental factors such as weather conditions, route congestion, train type, time of travel, and historical delay patterns [4][5]. Traditional delay management approaches are largely reactive, relying on static timetables, manual supervision, and historical averages, which limit their ability to anticipate disruptions in advance [6]. These conventional systems often fail to capture nonlinear relationships and time-dependent interactions inherent in railway operations, thereby reducing forecasting accuracy and operational efficiency.

Recent advancements in artificial intelligence and deep learning have provided powerful tools for modelling complex transportation systems [7]. Deep learning architectures such as Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), and Bidirectional LSTM (BiLSTM) have demonstrated strong performance in capturing nonlinear dependencies and temporal dynamics in large-scale datasets [8][9]. In particular, sequence-based models like LSTM and BiLSTM are highly effective in learning delay propagation patterns across time intervals and stations [10]. These models enable the extraction of meaningful patterns from operational and environmental data, thereby enhancing prediction reliability.

In this work, we propose a comprehensive deep learning–based framework for train delay prediction by leveraging both operational and environmental features. The dataset includes station-to-station distance, weather conditions, day of the week, time slots, train category, historical delays, and route congestion levels. Four deep learning architectures—DNN, CNN, LSTM, and BiLSTM—are implemented and comparatively evaluated using standard

performance metrics such as RMSE, MAE, R^2, and sMAPE. By analyzing the strengths and limitations of each model, this study aims to identify the most effective approach for accurate and real-time delay forecasting.

By learning complex temporal and nonlinear relationships from historical railway data, deep learning models such as DNN, CNN, LSTM, and BiLSTM enable more accurate and reliable delay forecasting. These models effectively capture operational patterns, time- dependent variations, and external influences like weather and congestion, thereby supporting intelligent railway management, proactive scheduling, real-time passenger information systems, and improved decision-making for both operators and travelers.

Railway systems frequently experience train delays due to the combined effects of operational, environmental, and temporal factors such as weather conditions, route congestion, train type, time of travel, and historical delay patterns. Existing delay management approaches are largely reactive, relying on static timetables, manual monitoring, and basic statistical methods, which limits their ability to predict delays in advance and respond effectively. The complex, nonlinear, and time-dependent nature of railway operations makes accurate delay forecasting a challenging task for traditional models. Therefore, there is a need for an intelligent, data-driven prediction framework that can effectively learn temporal dependencies and nonlinear relationships from historical railway data.Addressing this problem is essential for enabling reliable train delay prediction, proactive scheduling, improved operational efficiency, and enhanced passenger satisfaction within modern railway management systems.

Overall, the increasing demand for reliable railway services and the limitations of traditional reactive systems highlight the urgent need for intelligent, data-driven prediction frameworks. The proposed system contributes toward proactive scheduling, improved operational efficiency, optimized resource utilization, reduced operational costs, and enhanced passenger information services, thereby supporting the development of smarter, more resilient, and technology-driven railway management systems capable of adapting to future transportation demands. By leveraging advanced deep learning architectures and comprehensive operational data, the framework enables accurate real-time delay forecasting and supports timely decision-making for railway authorities. Furthermore, the integration of predictive analytics into railway operations minimizes cascading disruptions, enhances network reliability, and improves overall service quality infrastructure.

2. LITERATURE SURVEY

2.1. Literature Review

*Mostafa Al Ghamdi(2023).*presented a heterogeneous machine learning ensemble approach for rail transit delay forecasting using publicly available railway datasets such as the Network Rail Darwin feed. Their methodology involved extensive preprocessing, including removal of missing records and logical data validation, followed by the application of multiple regression models such as Linear Regression, Lasso, Ridge, Random Forest, and Gradient Boosting. The ensemble framework consistently outperformed individual models, demonstrating improved robustness and prediction accuracy. However, the study highlighted limitations in defining effective diversity metrics for regression ensembles and did not incorporate deep temporal learning for sequential delay propagation[1].

*Hongmeng Cui (2024).*presented a Spatial–Temporal Long Short-Term Memory (ST-LSTM) model for short-term origin–destination passenger flow prediction in metro systems. The preprocessing pipeline included min–max normalization and stratified OD sampling to address data sparsity and imbalance. By capturing both spatial and temporal dependencies, the proposed model achieved superior performance compared to traditional statistical approaches. Nevertheless, the study did not incorporate dynamic external disruptions such as weather conditions, equipment failures, or operational incidents, which play a crucial role in real-world train delay prediction[2].

*Thomas Spanninger (2024***).**introduced an uncertainty-aware neural network (UANN) to quantify the dynamic predictability of train delays. The model categorized trains based on fast and slow traffic directions and analyzed delay propagation under uncertainty. Experimental results showed that UANN outperformed Markov chains and Bayesian networks in handling uncertainty. However, the final delay predictions still exhibited large errors during extreme disruption scenarios, limiting its reliability in highly congested networks[3].

*Luís Marques(2024).*proposed a data-driven framework to reduce uncertainty in train delay prediction using neural networks and tree-based models. Their preprocessing approach included outlier handling using interquartile range (IQR) and standard deviation techniques. The study found that neural networks and recurrent neural networks perform well on smaller datasets, while tree-based models[4].

*Bishal Sharma(2023).*presented a comprehensive review of passenger-oriented railway rescheduling and delay management approaches. The study analyzed a wide range of exact, heuristic, and data-driven methods used for railway delay mitigation. While the review provided valuable insights into passenger-centric delay management strategies, it highlighted the limited research on microscopic and real-time predictive models that can dynamically adapt to evolving operational conditions[5].

*Pipatphon Lapamonpinyo(2022).*presented a real-time passenger train delay prediction framework using machine learning, focusing on Amtrak passenger train routes. The study utilized HTML and JSON data obtained from ASMAD and Weather Underground, which were parsed and structured using Python-based tools and stored in a MySQL database. Multiple machine learning models, including Linear Regression, Random Forest, Gradient Boosting Machine, and Multi-layer Perceptron (MLP), were evaluated. Experimental results showed that the MLP model combined with rolling-window historical delay features achieved the best performance with an R^2 value of approximately 0.62[6].

*Qianyi Liu(2022).*proposed a delay propagation prediction approach for high-speed rail systems by integrating causal text information with operational data. The methodology involved Chinese text segmentation using Jieba, stopword removal based on Harbin, Baidu, and SCU lists, and text regularization. Feature representations were generated using Word2Vec (CBOW and Skip-gram) and TF-IDF weighting, followed by regression models such as XGBoost, SVR, Random Forest Regression, Gradient Boosting, AdaBoost, LightGBM, KNN, and Ridge Regression. The results demonstrated that integrating delay cause text with operational context significantly improved prediction accuracy, with the XGBoost model combined with CBOW and mean embedding achieving 84% accuracy within a 3-minute margin[7].

*Pranjal Mandhaniya(2022).*investigated train delay prediction at a station using delays from previous stops as input features, motivated by the behavior of complex systems following Murphy's Law. The authors generated lagged delay features from upstream stations and handled early arrivals by allowing negative delay values. Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM) models were implemented and compared. The experimental results indicated that the LSTM model achieved the best convergence and prediction performance, with a maximum R^2 value of 0.81[8].

*Mark M. Dekker et al(2022).*modeled railway delay propagation as a diffusion-like spreading process using large-scale network data. The methodology involved reconstructing the railway network, removing outlier trains, interpolating skipped intermediate stations, and applying K-means clustering based on geographical information. A custom diffusion-based delay propagation model was proposed and analyzed using spectral and network clustering techniques. The study demonstrated that the diffusion-like model efficiently captures system-wide delay propagation patterns with relatively low computational complexity, provided that appropriate spatial aggregation is applied. However, the model assumes no new delay generation, full delay transfer between consecutive stations, and does not account for train-level discrete operational behaviors, limiting its ability to represent fine-grained delay dynamics[9].

Thomas James Tiam-Lee and Rui Henriques (2022) addressed route choice estimation in rail transit systems using smart card data while accounting for uncertainties in vehicle schedules and passenger walking times. The study applied rolling averages and deviation analysis on exit timestamps, volume spike detection, clustering of passenger flows, and matching of entry–exit pairs. A likelihood-based estimation approach combined with candidate route simulation was used to infer passenger route choices without relying on precise timetables. The results demonstrated that the proposed method effectively captures real passenger preferences, such as minimizing transfers or station counts. However, the approach lacks ground-truth validation, cannot infer individual personal factors influencing route choice, and its performance is affected by incomplete entry–exit records[10].

2.2. Deep Learning Methods with the Proposed Work

Deep Neural Network (DNN):

Deep Neural Networks are used in this work to model complex nonlinear relationships between operational and environmental features such as historical delays, weather conditions, train type, and time of travel. Prior studies, including real-time delay prediction research using Multi-Layer Perceptron models, demonstrate that feed-forward neural networks are effective for structured railway data. Inspired by these findings, the proposed DNN model serves as a strong baseline architecture that captures feature interactions without explicit temporal sequencing, providing reliable delay predictions for static and aggregated input features.

Convolutional Neural Network (CNN):

Convolutional Neural Networks are employed to extract localized temporal patterns from structured railway delay data. Although CNNs have been less explored in the reviewed literature for train delay prediction, their success in capturing spatial and short-term temporal patterns motivates their inclusion in this study. By applying one-dimensional convolution operations over time-ordered feature sequences, the CNN model identifies local delay trends and operational variations, contributing to improved prediction performance when combined with deep learning architectures.

Long Short-Term Memory (LSTM):

Long Short-Term Memory networks are utilized to capture long-term temporal dependencies in train delay sequences. Previous research has demonstrated that LSTM models outperform standard RNN and GRU architectures when learning delay propagation across consecutive stations. By maintaining memory through gating mechanisms, the LSTM model effectively handles sequential dependencies and varying time intervals in railway operations, making it well-suited for predicting delays influenced by historical patterns.

Bidirectional Long Short-Term Memory (BiLSTM):

Bidirectional LSTM networks extend the capabilities of standard LSTM by processing input sequences in both forward and backward directions. This allows the model to learn from both past and future contextual information within the delay sequence. Although most reviewed studies focus on unidirectional sequence models, the limitations identified in capturing full temporal context motivate the adoption of BiLSTM in this work. By leveraging bidirectional temporal learning, the BiLSTM model aims to improve prediction accuracy and robustness in complex and dynamic railway environments.

3. SYSTEM ANALYSIS

System analysis involves a detailed study of the problem domain, existing solutions, their limitations, and the requirements for the proposed system. In the context of train delay prediction, the system must handle large volumes of operational and environmental data while accurately forecasting delays in real time. Railway delay prediction is a complex task due to the dynamic nature of railway operations and the interdependence of multiple influencing factors such as weather conditions, route congestion, train type, and historical delay patterns.

3.1 Existing System Analysis:

In existing railway delay management systems, delay prediction is mostly based on static schedules, manual supervision, and simple statistical or rule-based methods. These systems are reactive in nature, as delays are identified only after they occur. Traditional models fail to capture complex nonlinear relationships and temporal dependencies present in real-world railway data. As a result, existing systems provide limited prediction accuracy, delayed responses, and inadequate support for proactive decision-making and passenger information services.

3.2 Disadvantages of the Existing System

- The existing system relies mainly on static timetables and manual monitoring, making it reactive rather than predictive.
- Traditional statistical and rule-based methods cannot handle complex nonlinear relationships among operational factors.
- The system does not effectively utilize historical data or real-time environmental information such as weather conditions.
- Human supervision increases the risk of errors and inconsistent delay management decisions.
- Existing systems lack the ability to model delay propagation across multiple stations and time intervals.
- Poor prediction accuracy leads to inefficient scheduling and resource utilization.
- Passengers receive delayed or inaccurate information, reducing trust in public transportation systems.

3.3. Proposed System Analysis:

The proposed system addresses the limitations of existing approaches by adopting deep learning–based models for train delay prediction. The system uses a comprehensive dataset containing operational, temporal, and environmental features. Advanced deep learning models such as Deep Neural Network (DNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Bidirectional LSTM (BiLSTM) are employed to learn complex nonlinear and temporal patterns from historical data. The system performs data preprocessing, feature encoding, and scaling to improve model performance and prediction reliability.

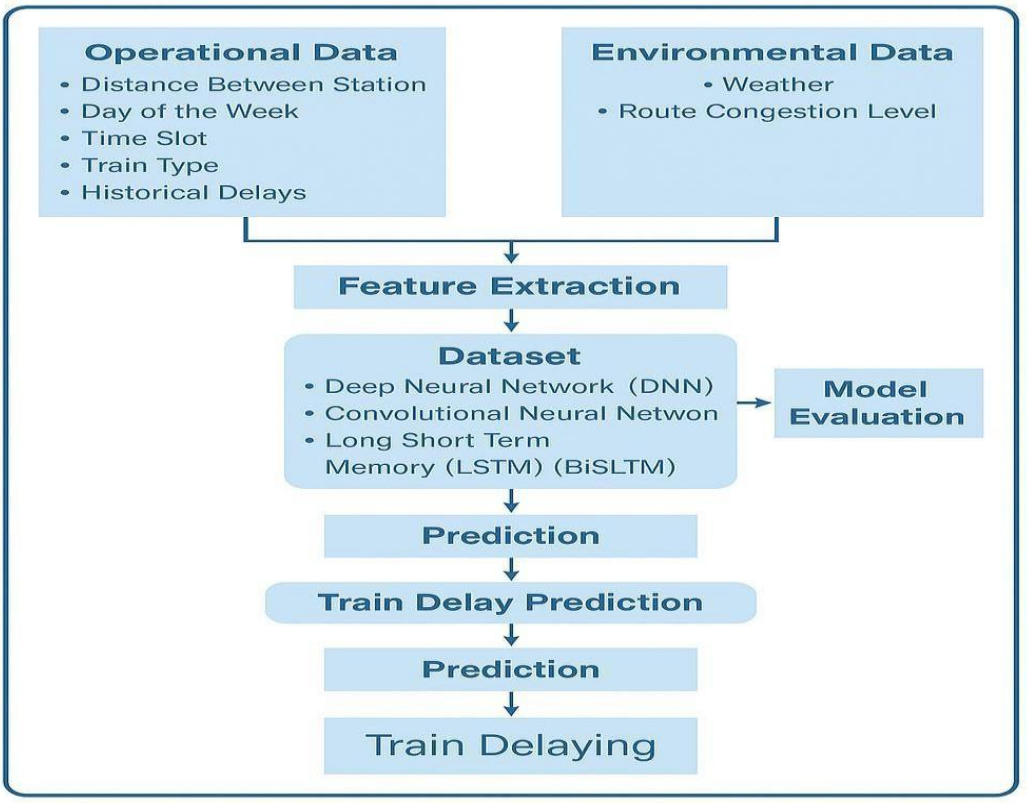

Figure1: FRAMEWORK FOR TRAIN DELAY PREDICTION

As shown in Figure 1, the system follows a structured pipeline that includes data collection, feature extraction, model training, evaluation, and train delay prediction.

1. Input Data Collection

The system starts by collecting two main types of data:

- **Operational Data**:

  This includes distance between stations, day of the week, time slot, train type, and historical delay information. These factors describe how trains operate under normal conditions.

- **Environmental Data**:
  This includes weather conditions and route congestion levels, which significantly influence train punctuality and delay occurrences.

2. Feature Extraction

The collected operational and environmental data are processed in the feature extraction stage. Relevant features are selected, cleaned, encoded, and transformed into a structured format suitable for model training. This step ensures that only meaningful and informative attributes are passed to the learning models.

3. Dataset Preparation

After feature extraction, a final dataset is created. This dataset is used as input to multiple deep learning models, including:

- Deep Neural Network (DNN)
- Convolutional Neural Network (CNN)
- Long Short-Term Memory (LSTM)
- Bidirectional LSTM (BiLSTM)

Each model learns different aspects of the data, such as nonlinear relationships and temporal dependencies.

4. Model Evaluation

The trained models are evaluated using standard performance metrics to measure prediction accuracy and reliability. This step helps identify the most effective model for train delay prediction.

5. Prediction Process

Once the best-performing model is selected, the system generates delay predictions based on new or unseen input data. The prediction stage estimates whether a train will be delayed and the expected delay duration.

6. Train Delay Prediction Output

The final output of the system is the train delay prediction, which indicates the likelihood and extent of train delays. This information can be used by railway authorities for proactive scheduling and by passengers for better travel planning.

3.3.1 Key Features of the Proposed System

- **Deep Learning–Based Prediction:**

The system employs advanced deep learning models such as Deep Neural Network (DNN),

Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Bidirectional LSTM (BiLSTM) to accurately predict train delays by learning complex patterns from historical data.

- **Integration of Multiple Data Sources:**

Operational, temporal, and environmental factors including historical delays, weather conditions, train type, time of travel, and route congestion are integrated to enhance prediction accuracy.

- **Effective Data Preprocessing:**

The system performs preprocessing steps such as handling missing values, encoding categorical features, time normalization, and feature scaling to ensure data consistency and improved model performance.

- **Temporal Dependency Modeling:**

Sequence-based models like LSTM and BiLSTM capture long-term temporal dependencies and delay propagation across stations, improving reliability of predictions.

- **Local Pattern Extraction:**

The CNN model identifies localized temporal patterns and short-term trends in railway delay data, complementing sequence-based learning.

- **Model Comparison and Evaluation:**

Multiple deep learning models are evaluated and compared using standard performance metrics such as RMSE, MAE, R^2, and sMAPE to identify the most effective approach.

- **Proactive Delay Forecasting:**

The system predicts delays in advance, enabling proactive scheduling, efficient resource utilization, and timely passenger notifications.

- **Scalable and Modular Architecture:**

The modular system design allows easy integration with real-time railway data, advanced deep learning models, and intelligent transportation systems in the future.

3.4. Advantages of the Proposed System Over the Existing System

- **Predictive Capability**

The proposed system predicts train delays in advance, whereas the existing system identifies delays only after they occur, enabling proactive actions.

- **Use of Advanced Deep Learning Models**

Unlike traditional rule-based methods, the proposed system uses deep learning models such as DNN, CNN, LSTM, and BiLSTM to learn complex nonlinear and temporal patterns.

- **Temporal Delay Modeling**

The system effectively captures delay propagation across stations and time intervals, which is not possible with existing approaches.

- **Integration of Multiple Data Sources**

Operational, historical, and environmental data such as weather conditions are integrated to improve prediction accuracy.

- **Reduced Human Dependency**

Automation through deep learning reduces manual monitoring and minimizes human errors in delay management.

- **Improved Decision-Making**

Accurate predictions support faster and better operational decisions for railway authorities.

- **Enhanced Passenger Experience**

Passengers receive timely and reliable delay information, improving trust in railway services.

- **Higher Prediction Accuracy**

The proposed deep learning–based framework achieves significantly higher prediction accuracy compared to traditional statistical and rule-based systems by learning complex feature interactions from large datasets.

3.5. Feasibility Study

A feasibility study is conducted to evaluate the practicality and viability of the proposed train delay prediction system before implementation. It analyzes whether the system is technically, economically, operationally, and legally feasible. This study ensures that the proposed solution can be successfully developed, deployed, and maintained within available resources and constraints.

1. Technical Feasibility:

- Uses established deep learning models such as DNN, CNN, LSTM, and BiLSTM.
- Can be implemented using widely available programming languages and frameworks like Python, TensorFlow, and PyTorch.
- Historical train schedules and environmental data can be processed using standard data preprocessing tools.
- Compatible with conventional computing platforms and GPU-supported systems.

2. Economic Feasibility:

- Relies on open-source tools and libraries, minimizing software licensing costs.

- Involves manageable expenses related to data collection, computation, and maintenance.
- Helps reduce operational losses by improving delay prediction accuracy.
- Cost-effective for academic and organizational deployment.

3. Operational Feasibility:

- Automates the train delay prediction process, reducing human intervention.
- Easy for railway operators and administrators to use for decision-making.
- Can be integrated with existing railway management systems without major changes.
- Enhances passenger information services through timely predictions.

4. Legal and Ethical Feasibility:

- Uses publicly available and historical operational data.
- Does not process sensitive personal passenger information.
- Complies with data privacy, security, and usage regulations.
- Ensures ethical use of data through proper handling and protection measures.

4. SYSTEM REQUIREMENTS

4.1. Software Requirements

Component	Specification / Description
Operating System	Windows 11, 64-bit Operating System
Coding Language	Python
Python Distribution / Framework	Google Colab Pro, Flask
Browser	Any latest browser such as Google Chrome

4.2. Hardware Requirements

Component	Specification / Description
Processor	Intel Core i5 or higher
Hardware Accelerator	CPU
RAM	Minimum 8 GB
Storage	Minimum 256 GB HDD / SSD

4.3. Requirement Analysis

Requirement analysis is a crucial phase in system development that identifies and defines the functional and non-functional requirements necessary for the successful implementation of the proposed train delay prediction system. The goal of this phase is to ensure that the system meets user expectations, operates efficiently, and delivers accurate and reliable predictions.

Functional Requirements:

The system must collect and process historical railway operational data, including train schedules, station-to-station distances, time slots, train types, and historical delay information. It should also incorporate environmental factors such as weather conditions

and route congestion levels. The system must perform data preprocessing tasks such as handling missing values, encoding categorical features, and normalizing numerical data. It should support training and testing of multiple deep learning models, including DNN, CNN, LSTM, and BiLSTM. The system must evaluate model performance using standard metrics such as RMSE, MAE, R^2, and sMAPE, and generate accurate train delay predictions for new input data.

Non-Functional Requirements:

The system should provide high prediction accuracy and reliability while handling large volumes of data efficiently. It must ensure scalability to accommodate additional data sources and future model enhancements. The system should maintain acceptable execution time for training and prediction processes. Data security and integrity must be ensured throughout the system, and the system should comply with data privacy standards. Additionally, the system should be user-friendly, allowing railway operators or administrators to easily interpret prediction results**.**

4.4. Software Description

The proposed train delay prediction system is developed using a robust and flexible software stack that supports data preprocessing, deep learning model implementation, training, evaluation, and result visualization. The system is primarily implemented using the Python programming language due to its simplicity, extensive library support, and strong ecosystem for machine learning and deep learning applications.

Python serves as the core development language and enables efficient handling of large datasets through libraries such as NumPy and Pandas. These libraries are used for data cleaning, transformation, feature extraction, and normalization of railway operational and environmental data. Data visualization libraries like Matplotlib and Seaborn are used to analyze data patterns and visualize model performance results.

Deep learning models including Deep Neural Network (DNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Bidirectional LSTM (BiLSTM) are implemented using popular deep learning frameworks such as TensorFlow and Keras. These frameworks provide prebuilt layers, optimization techniques, and GPU/CPU support, enabling efficient model training and experimentation.

Google Colab Pro is used as the development and execution environment, offering cloud-based resources and faster computation without local hardware constraints. Flask is

optionally used to deploy the trained model as a web application, enabling real-time delay prediction through a user-friendly interface. A modern web browser such as Google Chrome is used to access the application and visualize results.

Overall, the selected software tools provide a cost-effective, scalable, and efficient environment for developing and deploying the train delay prediction system. The use of open-source technologies ensures flexibility, ease of maintenance, and future extensibility of the system.

The software architecture of the proposed system follows a modular and layered design to ensure maintainability and scalability. Each module is independently responsible for specific tasks such as data ingestion, preprocessing, model training, evaluation, and prediction. This modular approach allows easy updates or replacement of individual components without affecting the overall system functionality.

The system supports efficient experiment management by enabling multiple model configurations and hyperparameter tuning. Logging mechanisms are incorporated to track training progress, model performance, and error metrics, which aids in debugging and performance optimization. Version control tools can be integrated to manage code changes and maintain reproducibility of experimental results.

The software is designed to handle both batch processing and real-time inference. Batch processing is used for training models on large historical datasets, while real-time inference enables delay prediction for new or incoming data. The system also supports exporting trained models for reuse, reducing the need for repeated training.

5. SYSTEM DESIGN

5.1 System Architecture:

System design describes how the proposed train delay prediction system is structured and how its components interact to achieve accurate delay forecasting. The design focuses on creating a reliable, scalable, and efficient framework that processes railway operational and environmental data and applies deep learning techniques to predict train delays.

The system follows a modular architecture in which each module performs a specific function. Data collection forms the first stage, where historical railway data and environmental factors such as weather conditions are gathered. This data is then passed to the preprocessing stage, where missing values are handled, categorical features are encoded, and numerical values are normalized to ensure data consistency.

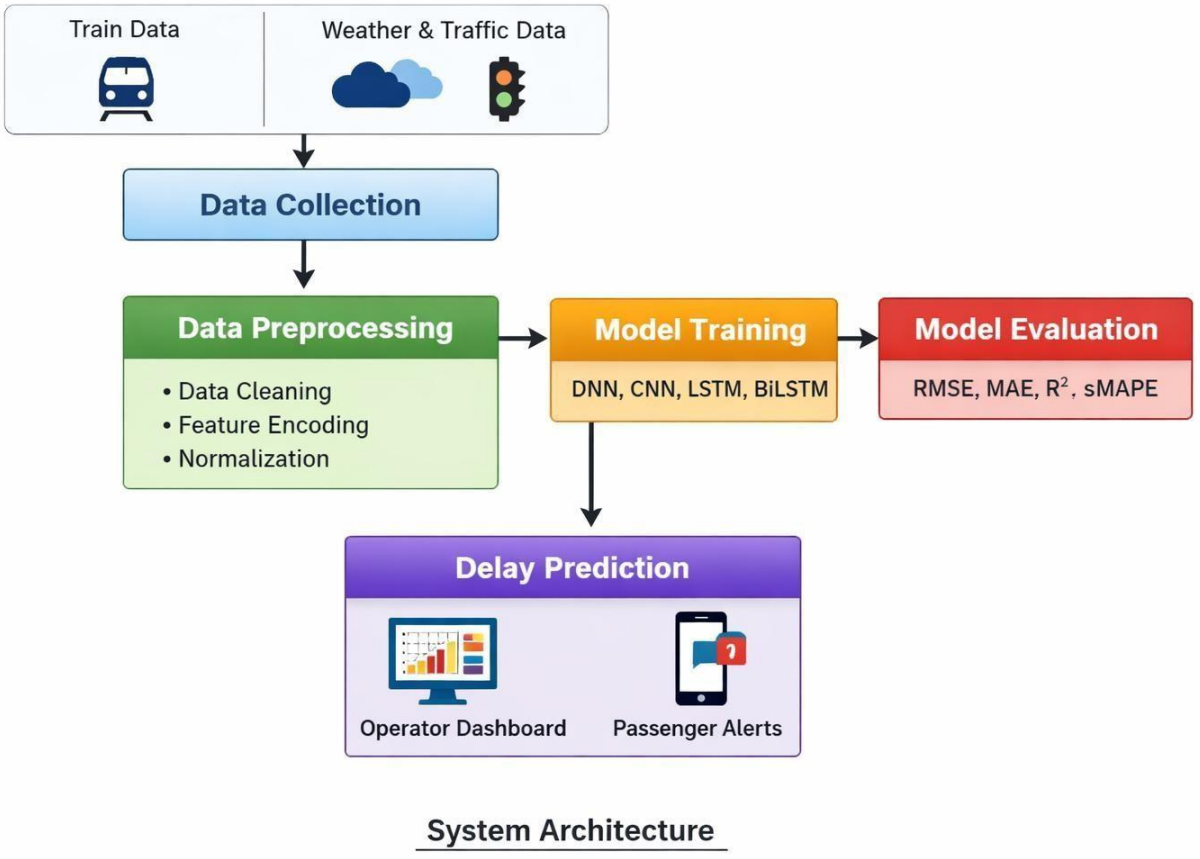

Figure 2 : Train Delay Prediction System Architecture

After preprocessing, the cleaned data is used to train deep learning models including Deep Neural Network (DNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Bidirectional LSTM (BiLSTM). These models are designed to learn complex nonlinear relationships and temporal dependencies present in train delay data.

Finally, the selected model is used to predict train delays for new input data. The system design supports easy integration with existing railway management systems and can be extended with real-time data sources in the future. Overall, the system design ensures efficient data flow, high prediction accuracy, and ease of maintenance.

Figure 2 illustrates the overall system architecture of the proposed train delay prediction system based on deep learning techniques. The architecture presents a structured workflow that begins with data collection and ends with accurate train delay prediction, supporting both railway operators and passengers.

The system first collects train operational data, which includes information such as train schedules, station details, and historical delays. In parallel, environmental data such as weather conditions and traffic or route congestion levels are gathered. These two data sources together provide a comprehensive view of factors influencing train delays.

The collected data is passed to the data preprocessing module, where data cleaning, feature encoding, and normalization are performed. This step removes inconsistencies, handles missing values, and converts raw data into a suitable format for model training, ensuring improved learning efficiency.

After preprocessing, the cleaned dataset is fed into the model training module, which implements multiple deep learning models including Deep Neural Network (DNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Bidirectional LSTM (BiLSTM). Each model learns different characteristics of the data, such as nonlinear relationships and temporal dependencies.

The trained models are then assessed in the model evaluation module using performance metrics such as RMSE, MAE, R^2, and sMAPE. This evaluation helps in identifying the most accurate and reliable model for delay prediction.

Finally, the selected model generates train delay predictions, which are presented through an operator dashboard and passenger alert system. These predictions enable proactive decision-making, efficient scheduling, and improved passenger information services.

5.2 Dataset Description

The dataset used in this study is prepared to support the prediction of train delays by capturing key operational and environmental factors that influence railway performance. Each record in the dataset represents a train journey between consecutive stations and contains both input features and a target delay value.

The dataset includes operational attributes such as distance between stations, day of the week, time slot of travel, train type, and historical delay information. These attributes reflect routine scheduling patterns and operational characteristics that affect train punctuality. Historical delay data is particularly useful for identifying recurring delay trends on specific routes or time periods.

In addition to operational features, the dataset also contains environmental attributes such as weather conditions and route congestion levels. Weather conditions capture external factors like clear, rainy, or foggy situations, while route congestion indicates the level of traffic on railway routes. These attributes help the model understand how external conditions contribute to delay occurrences.

The target variable in the dataset represents the train delay, usually measured in minutes, indicating the difference between scheduled and actual arrival times. Before using the dataset for model training, preprocessing steps such as handling missing values, encoding categorical variables, normalizing numerical features, and splitting data into training and testing sets are performed.

Overall, the dataset provides a comprehensive representation of both internal operational factors and external environmental conditions, making it suitable for training and evaluating deep learning models such as DNN, CNN, LSTM, and BiLSTM for accurate train delay prediction.

Source of the Dataset

The dataset is collected from historical railway operational records and simulated environmental conditions for academic and research purposes. It reflects realistic train movement patterns and delay behavior observed in railway networks.

Operational Features

The operational attributes in the dataset include distance between stations, day of the week, time slot of travel, train type, and historical delay information. These features capture scheduling patterns, route characteristics, and operational behavior that significantly affect train punctuality.

Environmental Features

Environmental attributes such as weather conditions and route congestion levels are included to represent external factors influencing train delays. Weather conditions capture scenarios like clear, rainy, or foggy situations, while congestion levels indicate the intensity of traffic on railway routes.

Target Variable

The target variable in the dataset is the train delay, typically measured in minutes. It represents the difference between the scheduled arrival time and the actual arrival time of the train and serves as the output for prediction models.

Data Pre-Processing

Before model training, the dataset undergoes preprocessing steps such as handling missing values, encoding categorical variables, normalizing numerical features, and transforming time-related data into numerical form. These steps improve data quality and model performance.

Dataset Usage

The prepared dataset is used to train and evaluate deep learning models such as DNN, CNN, LSTM, and BiLSTM. It enables the system to learn complex nonlinear and temporal patterns for reliable train delay prediction.

Table 1 : Sample Train Delay Prediction Dataset

	A	B	C	D	E	F	G
1	Distance Betwee	Weather Conditi	Day of the Week	Time of Day	Train Type	Historical Delay	Route Congestion
2	100	Clear	Monday	Morning	Express	5	Low
3	150	Rainy	Tuesday	Afternoon	Superfast	10	Medium
4	200	Foggy	Wednesday	Evening	Local	15	High
5	50	Clear	Thursday	Night	Express	2	Low
6	75	Rainy	Friday	Morning	Superfast	8	Medium
7	125	Foggy	Saturday	Afternoon	Local	20	High
8	90	Clear	Sunday	Evening	Express	0	Low
9	60	Rainy	Monday	Night	Superfast	12	Medium
10	110	Clear	Tuesday	Morning	Local	3	High
11	95	Foggy	Wednesday	Afternoon	Express	25	Low
12	130	Clear	Thursday	Evening	Superfast	5	Medium
13	85	Rainy	Friday	Night	Local	7	High
14	160	Clear	Saturday	Morning	Express	0	Low
15	70	Rainy	Sunday	Afternoon	Superfast	30	Medium
16	45	Foggy	Monday	Evening	Local	22	High
17	180	Clear	Tuesday	Night	Express	4	Low
18	55	Rainy	Wednesday	Morning	Superfast	11	Medium
19	105	Foggy	Thursday	Afternoon	Local	16	High
20	120	Clear	Friday	Evening	Express	1	Low
21	80	Rainy	Saturday	Night	Superfast	9	Medium
22	140	Clear	Sunday	Morning	Local	3	Low
23	65	Rainy	Monday	Afternoon	Express	14	High
24	175	Foggy	Tuesday	Evening	Superfast	18	Medium
25	30	Clear	Wednesday	Night	Local	1	Low
26	115	Rainy	Thursday	Morning	Express	9	High
27	100	Foggy	Friday	Afternoon	Superfast	20	Medium
28	190	Clear	Saturday	Evening	Local	2	Low
29	145	Rainy	Sunday	Night	Express	12	High
30	50	Foggy	Monday	Morning	Superfast	17	Medium

1. Distance Between Stations (km)

This numerical attribute represents the physical distance between two consecutive railway stations measured in kilometers. Longer distances may increase the likelihood of delays due to operational and environmental factors.

2. Weather Condition

This categorical feature represents the weather conditions during the train journey, such as Clear, Rainy, or Foggy. Weather plays a significant role in influencing train speed, visibility, and operational safety, thereby contributing to delays.

3. Day of the Week

This categorical attribute indicates the day on which the train operates, such as Monday or Tuesday. Travel demand and congestion levels often differ between weekdays and weekends.

4. Time of Day

This categorical feature represents the time slot of the train journey, such as Morning, Afternoon, Evening, or Night. Traffic density and operational load vary across different time periods, affecting delay occurrence.

5. Train Type

This categorical attribute identifies the type of train, such as Express, Superfast, or Local. Different train categories follow distinct schedules and priority rules, which influence delay patterns.

6. Historical Delay

This numerical feature represents the average or previously observed delay in minutes for similar trains or routes. It provides historical context and helps the model learn recurring delay patterns.

7. Route Congestion Level

This categorical attribute represents the level of congestion on the railway route, classified as Low, Medium, or High. Higher congestion levels generally lead to increased waiting times and greater delay occurrences.

5.3. Data Preprocessing

Data preprocessing is a fundamental step in the proposed train delay prediction system, as railway datasets are often large, heterogeneous, and prone to inconsistencies. Since the performance of deep learning models heavily depends on data quality, preprocessing ensures that the input data is accurate, consistent, and suitable for learning complex patterns. This phase transforms raw operational and environmental data into a structured and model-ready format.

1. Handling Missing and Inconsistent Data:

The collected railway dataset may contain missing values due to sensor failures, manual data entry errors, or incomplete records. Inconsistent entries such as incorrect distance values or invalid delay times are also observed.

During preprocessing, such records are systematically identified through validation checks. Missing numerical values are treated using statistical imputation methods, while categorical inconsistencies are corrected or removed. This step ensures that the dataset remains reliable and prevents biased learning during model training.

2. Encoding Categorical Features:

Many important attributes in the dataset, including weather conditions, day of the week, time slots, train type, and route congestion levels, are categorical in nature. Since deep learning models cannot directly process non-numeric data, these attributes are converted into numerical representations using label encoding techniques. Encoding preserves the distinct categories and allows the models to effectively learn how different operational and environmental conditions influence train delays.

3. Time Normalization:

Time-related features play a crucial role in delay prediction, as railway operations vary significantly across different periods of the day. Time slots such as morning, afternoon, evening, and night are normalized into numerical formats to maintain uniformity. Delay values are also standardized to ensure consistent representation. Time normalization helps the models identify temporal trends and patterns associated with peak and off-peak hours.

4. Feature Scaling:

Numerical features such as distance between stations and historical delay values often have different ranges and units. Feature scaling is applied to normalize these attributes so that no single feature dominates the learning process. Standardization techniques are used to bring all numerical features to a common scale, which improves model convergence, reduces training time, and enhances overall prediction accuracy.

5. Outlier Handling:

Outliers represent extreme delay values caused by rare events such as accidents, equipment failures, or severe weather conditions. These values are carefully analyzed to determine their impact on model performance. In cases where outliers distort learning, appropriate smoothing or treatment techniques are applied. This step helps the model focus on realistic delay patterns while maintaining robustness against abnormal scenarios.

6. Dataset Splitting:

After completing preprocessing, the dataset is divided into training and testing sets, typically using an 80:20 ratio. The training dataset is used to train deep learning models, while the testing dataset is reserved for evaluating model performance on unseen data. This separation ensures unbiased assessment and helps validate the generalization capability of the models.

7. Prepared Dataset for Model Training:

The final preprocessed dataset consists of clean, encoded, normalized, and scaled features suitable for training DNN, CNN, LSTM, and BiLSTM models. The structured dataset enables these models to effectively capture nonlinear relationships, temporal dependencies, and sequential delay propagation patterns present in railway operations**.**

5.3.1 Feature Extraction

Feature extraction is a critical step in the train delay prediction system, as it focuses on identifying and selecting the most relevant attributes that significantly influence train delays. Effective feature extraction improves model performance by reducing redundancy, enhancing interpretability, and enabling deep learning models to learn meaningful patterns from the data.

The extracted features in this study are derived from both operational and environmental data sources. Operational features include distance between stations, day of the week, time slot of the journey, train type, and historical delay values. These features capture scheduling behavior, operational characteristics, and recurring delay patterns. Historical delay information is particularly important, as it reflects long-term trends and past operational inefficiencies.

Environmental features such as weather conditions and route congestion levels are also extracted to represent external factors affecting train performance. Weather conditions influence visibility, track conditions, and train speed, while congestion levels indicate traffic intensity on railway routes. These features help the model understand how external disruptions contribute to delay occurrences.

During feature extraction, irrelevant or redundant attributes are eliminated to reduce noise and computational complexity. Categorical features are transformed into numerical representations during preprocessing, ensuring compatibility with deep learning models. Time-related features are structured in a way that allows the model to capture temporal dependencies effectively

Categorical features are transformed into numerical representations during preprocessing, ensuring compatibility with deep learning models. Time-related features are while congestion levels indicate traffic intensity on railway routes. These features help the model understand how external disruptions contribute to delay occurrences.

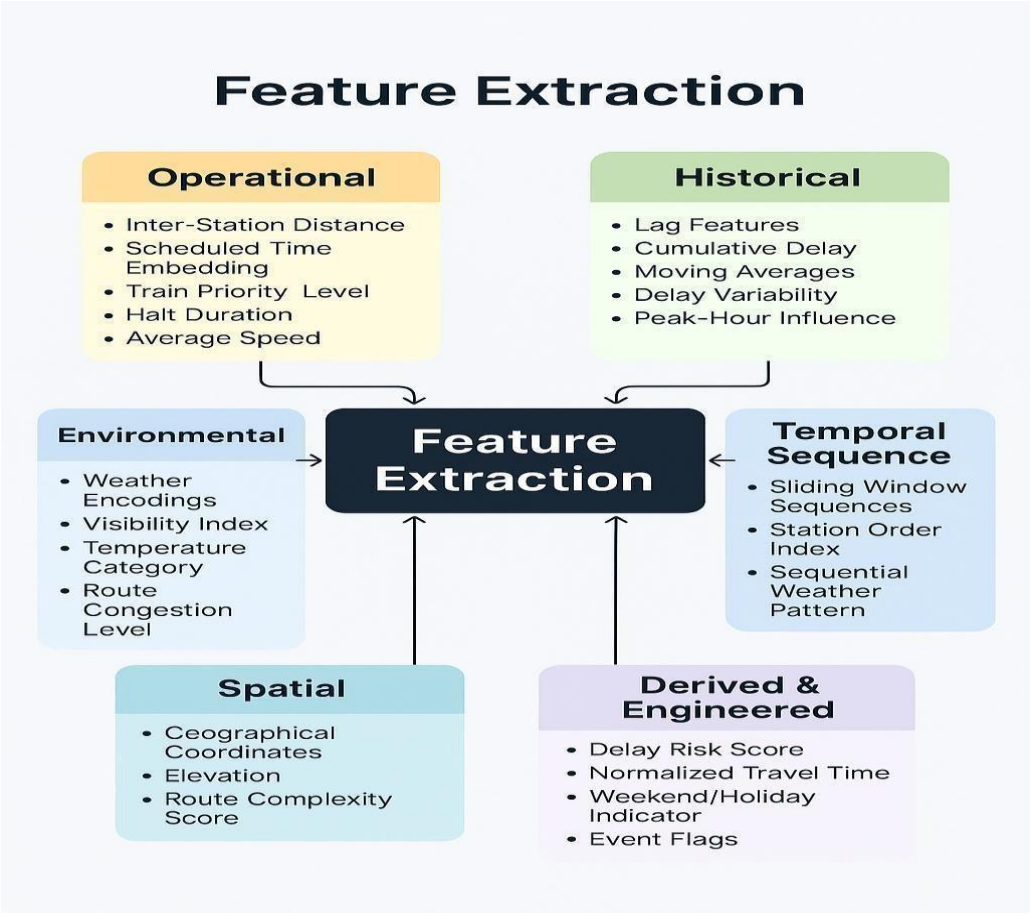

Figure 3: Feature Extraction Framework for the Train Delay Prediction System

The final feature set is carefully designed to support advanced deep learning architectures such as DNN, CNN, LSTM, and BiLSTM. These extracted features enable the models to learn nonlinear relationships and sequential patterns, leading to accurate and reliable train delay prediction.

Figure 3 illustrates the feature extraction framework used in the proposed train delay prediction system. The framework demonstrates how raw railway data is transformed into meaningful and structured features that effectively represent operational, historical, environmental, spatial, and temporal characteristics influencing train delays.

Operational Features:

Operational features describe routine train operations and scheduling characteristics. These include inter-station distance, scheduled time embedding, train priority level, halt duration, and average speed. Such features capture how trains are planned and operated, directly influencing delay behavior under normal operating conditions.

Historical Features:

Historical features are derived from past delay records and represent long-term delay patterns. These include lag features, cumulative delay, moving averages, delay variability, and peak-hour influence. By incorporating historical trends, the model can learn recurring delay behaviors and anticipate future delays based on past performance.

Environmental Features:

Environmental features represent external conditions that affect train movement and safety. These include weather encodings, visibility index, temperature category, and route congestion level. Environmental factors often introduce unexpected disruptions, making them critical for improving prediction accuracy.

Temporal Sequence Features:

Temporal sequence features capture the time-dependent and sequential nature of train delays. Sliding window sequences, station order index, and sequential weather patterns are used to model how delays propagate across stations and time. These features are particularly important for sequence-based deep learning models such as LSTM and BiLSTM.

Spatial Features:

Spatial features describe geographical and infrastructural characteristics of railway routes. These include geographical coordinates, elevation, and route complexity score. Spatial attributes help the model understand how terrain and route structure influence train speed and delay propagation.

Derived and Engineered Features:

Derived and engineered features are created by combining or transforming existing attributes to enhance predictive power. These include delay risk scores, normalized travel time, weekend or holiday indicators, and event flags. Such engineered features provide high-level insights that are not directly available in raw data.

Integration of Features:

All extracted features are integrated into a unified feature extraction layer, ensuring that deep learning models receive rich and structured inputs. This comprehensive feature representation enables the models to learn complex nonlinear relationships, spatial dependencies, and temporal patterns, leading to more accurate and reliable train delay prediction.

5.4 Model Architecture

The proposed train delay prediction system employs multiple deep learning architectures to effectively learn complex nonlinear relationships and temporal dependencies present in railway operational data. The models used in this study include Deep Neural Network (DNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Bidirectional LSTM (BiLSTM). Each architecture is designed to capture different characteristics of the dataset, enabling a comprehensive comparison of predictive performance.

5.4.1 Overview of the Proposed Model

The proposed model aims to accurately predict train delays by leveraging deep learning techniques on historical operational and environmental railway data. The model is designed to learn complex nonlinear relationships and temporal dependencies that influence delay occurrences, enabling proactive and reliable delay forecasting.

The system begins by collecting and preprocessing railway data that includes operational features such as distance between stations, day of the week, time slot, train type, and historical delay information, along with environmental factors such as weather conditions and route congestion levels. After preprocessing, relevant features are extracted and structured into a model-ready format.

The processed data is then fed into multiple deep learning architectures, including Deep Neural Network (DNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Bidirectional LSTM (BiLSTM). Each model analyzes the same input data but learns patterns in different ways—DNN captures nonlinear feature interactions, CNN extracts localized temporal patterns, while LSTM and BiLSTM model sequential and bidirectional temporal dependencies.

Model performance is evaluated using standard metrics such as RMSE, MAE, R^2, and sMAPE to ensure accurate and unbiased comparison. Based on evaluation results, the most effective model is selected for final deployment. The selected model generates train delay predictions for new or unseen input data, supporting proactive scheduling, operational decision-making, and improved passenger information services.

Overall, the proposed model provides a scalable, data-driven, and intelligent framework for train delay prediction, addressing the limitations of traditional reactive systems and enhancing the reliability of railway operations.

5.4.2 Phases of the Proposed Model

Phase 1: Data Collection

In this phase, historical railway operational data and environmental data are collected. Operational data includes distance between stations, train type, day of the week, time slot, and historical delays, while environmental data includes weather conditions and route congestion levels. These datasets form the foundation of the prediction system.

Phase 2: Data Preprocessing

The collected raw data is cleaned and prepared for model training. This phase involves handling missing and inconsistent values, encoding categorical features, normalizing time-related attributes, and scaling numerical features. Proper preprocessing ensures data consistency and improves model accuracy.

Phase 3: Feature Extraction

Relevant operational and environmental features that significantly influence train delays are identified and extracted. Redundant and irrelevant attributes are removed to reduce noise and computational complexity. The extracted features are structured to support both static and sequential learning.

Phase 4: Model Training

In this phase, the preprocessed dataset is used to train deep learning models such as DNN, CNN, LSTM, and BiLSTM. Each model learns patterns from the data based on its architecture, capturing nonlinear relationships and temporal dependencies related to train delays.

Phase 5: Model Evaluation

The trained models are evaluated using standard performance metrics such as RMSE, MAE, R^2, and sMAPE. This phase helps in comparing different models and selecting the most accurate and reliable one for delay prediction.

Phase 6: Delay Prediction

The best-performing model is used to predict train delays for new or unseen data. The predicted delay values provide insights into expected delays, enabling proactive scheduling and operational planning.

Phase 7: Result Visualization and Decision Support

The prediction results are presented through visual outputs or dashboards for easy interpretation. These results assist railway authorities in decision-making and help passengers receive timely delay inform

5.5 Classifications

Classification in the proposed train delay prediction system refers to the systematic grouping of data, models, and prediction outcomes into meaningful categories. Proper classification helps in better understanding delay behavior, improving model performance, and supporting effective decision-making in railway operations.

1. Classification Based on Data Nature

The dataset used in the proposed system is classified based on the nature of the data to ensure appropriate preprocessing and effective model learning.

Numerical:

This category includes continuous-valued attributes such as distance between stations, historical delay values, average speed, and cumulative delay. These features are scaled and directly used by deep learning models to learn quantitative relationships influencing train .

Categorical:

Categorical data consists of qualitative attributes such as weather condition, day of the week, time of day, train type, and route congestion level. These features describe operational and environmental conditions and are encoded into numerical form during preprocessing to make them suitable for model training.

Temporal:

Temporal data includes time-dependent attributes such as journey time slots, sequential delay values, and station order information. These features are crucial for modeling delay progression over time and are particularly important for sequence-based models like LSTM and BiLSTM.

2. Classification Based on Operational Conditions

Train journeys in the proposed system are classified based on their operational characteristics to capture variations in scheduling and traffic conditions.

Train-Category

Trains are categorized as Express, Superfast, or Local based on their operational priority and scheduling rules. Different train categories follow distinct operational policies, which directly influence delay behavior.

Time-Based

Train journeys are classified into Morning, Afternoon, Evening, and Night time slots. Traffic density, operational load, and passenger demand vary across these periods, making time-based classification important for accurate delay modeling.

Day-Based

Train operations are classified into weekdays and weekends. Passenger demand and congestion levels typically differ between these periods, affecting train punctuality and delay patterns.

3. Classification Based on Delay Severity

Predicted train delays are classified into severity levels to simplify interpretation and support effective decision-making.

- **On-Time** – Represents trains with no delay or negligible deviation from the schedule.
- **Minor Delay** – Indicates small delays that generally do not disrupt operations.
- **Moderate Delay** – Represents noticeable delays that may affect scheduling and passenger travel plans.

- **Severe Delay** – Indicates significant delays requiring immediate operational intervention.
- This classification enables railway authorities to prioritize responses and communicate delay information clearly to passengers.

4. Classification Based on Learning Models

The deep learning models used in the system are classified based on their learning behavior and data processing approach.

Static-Learning

Deep Neural Network (DNN) and Convolutional Neural Network (CNN) are classified as static learning models. These models focus on learning nonlinear relationships and localized patterns from structured input data.

Sequential-Learning

Long Short-Term Memory (LSTM) and Bidirectional LSTM (BiLSTM) are classified as sequential learning models. They are designed to capture temporal dependencies and delay propagation across time and stations.

5. Classification Based on Prediction Usage

The prediction outputs of the system are classified based on their application context.

Offline-Predictions

Offline predictions are used for historical analysis, model evaluation, and long-term planning purposes.

Real-Time-Predictions

Real-time predictions support live monitoring, operational decision-making, and timely passenger alerts.

6. Output Classes

The output of the proposed train delay prediction system is categorized into distinct classes to represent different levels of train delay severity. These output classes help simplify interpretation of prediction results and support effective operational decision-making.

Class 1: On-Time

This class represents trains that arrive on schedule or experience negligible delay. Trains classified as *On-Time* require no operational intervention and indicate smooth railway operations.

Class 2: Minor Delay

This class includes trains that experience a small delay within an acceptable threshold. Minor delays generally do not affect passenger connections or overall schedule stability.

Class 3: Moderate Delay

Trains classified under moderate delay experience noticeable delays that may impact passenger travel plans and operational schedules. These delays may require monitoring or minor corrective actions.

Class 4: Severe Delay

This class represents trains with significant delays caused by major operational disruptions, high congestion, or adverse environmental conditions. Severe delays require immediate attention from railway authorities and may trigger rescheduling or passenger alerts.

7. Significance of Classification

Classification plays a crucial role in the proposed train delay prediction system by transforming raw prediction outputs into meaningful and actionable categories. Instead of providing only numerical delay values, classification organizes prediction results into distinct delay classes, making them easier to interpret and use for operational decision-making.

One of the primary significances of classification is improved interpretability. Railway operators and passengers can easily understand delay conditions when predictions are categorized as on-time, minor delay, moderate delay, or severe delay. This clarity supports faster and more effective responses to potential disruptions.

Classification also enhances decision support for railway authorities. By identifying the severity level of delays, operators can prioritize actions such as rescheduling, resource allocation, or issuing alerts. Severe delay classes can trigger immediate interventions, while minor delays may require only monitoring.

Another important significance is better system efficiency. Classification reduces the complexity of handling continuous prediction outputs and enables the system to operate efficiently in real-time environments. It also simplifies performance evaluation and comparison of models by allowing class-based accuracy analysis.

Additionally, classification improves passenger communication. Clearly defined delay categories help provide timely and understandable information to passengers, improving travel planning and overall satisfaction.

Overall, classification enhances the practicality, reliability, and usability of the train delay prediction system by converting complex model outputs into structured and meaningful information that supports intelligent railway management.

5.6. Modules

The proposed train delay prediction system follows a modular design approach to ensure scalability, maintainability, and efficient processing of large and complex railway datasets. Each module performs a specific function and interacts with other modules through well-defined interfaces, enabling smooth data flow and systematic execution.

1. Data Collection Module

The Data Collection Module is responsible for acquiring historical and real-time railway data from multiple sources. This includes operational data such as station-to-station distance, train category, journey time slot, day of operation, and historical delay information, as well as environmental data like weather conditions and route congestion levels. The module ensures that collected data is consistently formatted and ready for further processing.

- Collects historical and real-time railway data from multiple sources.
- Gathers operational data such as station-to-station distance, train category, journey time slot, day of operation, and historical delay information.
- Collects environmental data including weather conditions and route congestion levels.
- Supports both batch data collection for model training and real-time data collection for live prediction.
- Performs initial validation checks to identify missing, duplicate, or inconsistent records.

- Ensures uniform data formatting across different data sources.
- Standardizes timestamps and categorical labels for consistency.

By organizing data into structured formats such as tables or data frames, the Data Collection Module establishes a strong foundation for subsequent preprocessing, feature extraction, and model training stages. A reliable data collection process ensures higher data quality, which directly contributes to improved model accuracy and system reliability.

2. Data Preprocessing Module

This module focuses on improving data quality by addressing issues such as missing values, inconsistent entries, and noise in the dataset. It applies techniques such as data cleaning, categorical encoding, time normalization, and numerical feature scaling. By standardizing the data, this module ensures that deep learning models receive reliable and consistent inputs, which improves training stability and prediction accuracy.

- Improves overall data quality by cleaning raw railway datasets.
- Identifies and handles missing values using appropriate imputation techniques.
- Detects and corrects inconsistent or invalid entries in the data.
- Removes or smooths noise and outliers that may affect model learning.
- Applies categorical encoding to convert non-numeric attributes into numerical form.
- Performs time normalization to standardize time-related features such as journey time slots.
- Applies feature scaling to numerical attributes to ensure uniform value ranges.
- Prepares a model-ready dataset suitable for deep learning architectures.
- Enhances training stability and prediction accuracy of DNN, CNN, LSTM, and BiLSTM models.

3. Feature Extraction and Selection Module

The Feature Extraction and Selection Module identifies the most informative features that significantly influence train delays. It transforms raw data into meaningful representations and removes redundant or irrelevant attributes to reduce computational overhead. Time-dependent features are structured to support sequence-based learning, enabling effective delay propagation modeling.

- Identifies the most informative features that significantly influence train delays.
- Transforms raw operational and environmental data into meaningful feature representations.
- Removes irrelevant and redundant attributes to reduce noise in the dataset.

- Reduces computational complexity by selecting only important features.
- Structures time-dependent features to support sequence-based learning.
- Helps models learn delay propagation patterns across time and stations.
- Improves overall model accuracy and efficiency.

4. Model Training Module

The Model Training Module implements multiple deep learning architectures, including DNN, CNN, LSTM, and BiLSTM. Each model is trained independently using the same preprocessed dataset to ensure fair comparison. Hyperparameter tuning, regularization techniques, and optimization strategies are applied to enhance model performance and prevent overfitting.

- Implements multiple deep learning models such as DNN, CNN, LSTM, and BiLSTM.
- Trains each model independently using the same preprocessed dataset.
- Ensures fair comparison across different model architectures.
- Applies hyperparameter tuning to optimize model performance.
- Uses regularization techniques such as dropout to avoid overfitting.
- Employs optimization algorithms like Adam for efficient learning.
- Enables models to learn nonlinear and temporal patterns in delay data.

5. Model Evaluation and Validation Module

This module evaluates the trained models using standard metrics such as RMSE, MAE, R^2, and sMAPE. Cross-validation and testing on unseen data are performed to assess generalization capability. The evaluation results guide the selection of the most suitable model for deployment.

- Evaluates trained models using standard performance metrics.
- Uses RMSE to measure prediction error magnitude.
- Uses MAE to assess average absolute error.
- Uses R^2 score to evaluate prediction accuracy and goodness of fit.
- Uses sMAPE for percentage-based error analysis.
- Performs testing on unseen data to measure generalization capability.
- Helps select the best-performing model for deployment.

6. Delay Prediction and Classification Module

The Delay Prediction and Classification Module uses the selected model to generate delay predictions for new input data. The predicted delays are further classified into predefined severity levels such as on-time, minor delay, moderate delay, and severe delay. This classification improves interpretability and supports faster operational decision-making.

Evaluates trained models using standard performance metrics.

- Uses RMSE to measure prediction error magnitude.
- Uses MAE to assess average absolute error.
- Uses R^2 score to evaluate prediction accuracy and goodness of fit.
- Uses sMAPE for percentage-based error analysis.
- Performs testing on unseen data to measure generalization capability.
- Helps select the best-performing model for deployment.

7. Result Visualization and User Interface Module

This module presents prediction results through charts, tables, and dashboards that are easy to interpret. It provides actionable insights for railway authorities and enables clear communication of delay information to passengers. Visualization tools enhance system usability and support data-driven decision-making.

- Displays prediction results using charts, graphs, and tables.
- Provides clear and user-friendly dashboards.
- Helps railway authorities analyze delay patterns and trends.
- Enables easy understanding of predictions for non-technical users.
- Supports data-driven decision-making.
- Improves system usability and accessibility.

8. System Integration and Deployment Module

The final module ensures seamless integration of the prediction system with existing railway management systems or web-based platforms. Deployment can be performed using cloud or local servers, enabling real-time prediction and future scalability.

- Integrates the prediction system with existing railway management systems.
- Supports deployment on cloud-based or local servers.
- Enables real-time delay prediction.
- Ensures smooth interaction between system components.
- Allows scalability for future data and model expansion..

5.7.UML Diagrams

What it shows: actors Passenger, Operator, Data Engineer, System Admin, External Data Provider and the main use cases (Ingest Data, Preprocess, Train/Evaluate, Select/Deploy Model, Predict, Monitor, Retrain, Alerts).

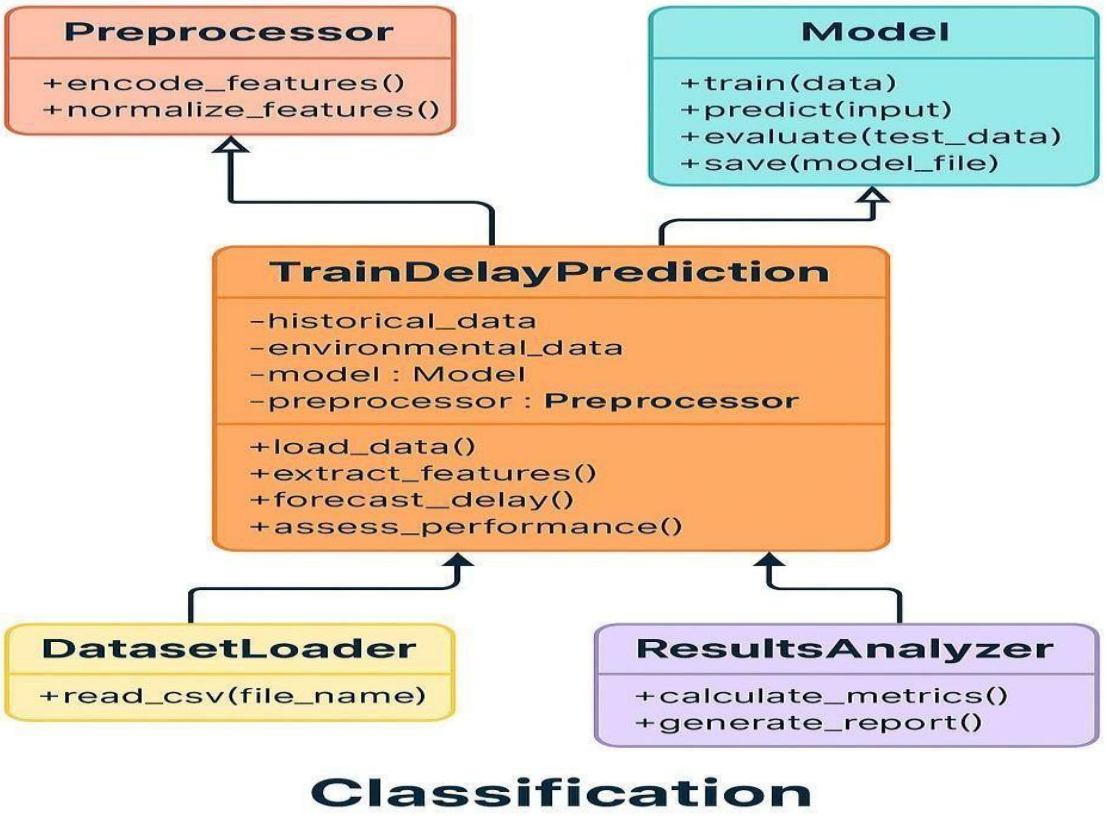

Figure 4: UML CLASS DIAGRAM OF TRAIN DELAY PREDICTION

Figure 4 illustrates the UML Class Diagram represents the structural design of the Train Delay Classification System. It illustrates the key components (classes), their attributes, methods, and how they interact with each other to perform data preprocessing, feature extraction, model training, prediction, and performance evaluation.

1. Train Delay Prediction (Central Class)

This is the core class that coordinates the entire workflow of the classification system.It holds the main datasets, model instance, and preprocessor instance, and provides the key high-level operations needed for the train delay prediction pipeline.

Attributes:

- **historical_data** – stores past train data.
- **environmental_data** – contains weather, congestion, and other environmental factors.

Methods:

- **load_data()** – loads and aggregates datasets using Dataset Loader.
- **extract_features()** – performs feature extraction using the Preprocessor.
- **forecast_delay()** – uses the model to predict train delay classes.
- **assess_performance()** – evaluates the model performance using ResultsAnalyzer.

This class acts as the **controller** of the entire system.

2. Preprocessor Class

This class handles **data cleaning and transformation**, ensuring that the data fed to the model is consistent and usable.

Methods:

- **encode_features()** – converts categorical data (e.g., weather, time slot, train type) into numerical representations.
- **normalize_features()** – applies scaling techniques to numerical features.

The Preprocessor ensures that both training and testing data share the same encoding and scaling schemes.

3. Model Class

This class represents the machine learning classification model (DNN, CNN, LSTM, BiLSTM, etc.)

Methods:

- **train(data)** – trains the model using input features and labels.
- **predict(input)** – generates predictions for new data.
- **evaluate(test_data)** – computes accuracy, F1-score, and other classification metrics.
- **save(model_file)** – saves the trained model for deployment.

This class abstracts the underlying deep learning logic.

4. DatasetLoader Class

This class is responsible for data ingestion and reading external datasets.

Method:

- **read_csv(file_name)** – loads dataset files in CSV format (historical logs, weather data, etc.).

It provides raw data to the Train Delay Prediction class for further processing.

5. ResultsAnalyzer Class

This class handles model performance evaluation and results reporting.

6. IMPLEMENTATION

```
from google.colab import drive
drive.mount('/content/drive')
from google.colab import files
uploaded = files.upload()
!pip install -q keras tensorflow scikit-learn pandas numpy
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
import tensorflow as tf
print("TensorFlow version:", tf.__version__)
!pip install -U tensorflow
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, r2_score
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
df = pd.read_csv('train delay data set.csv')
df.head()
df = df.dropna()
X = df.drop(columns=['Historical Delay (min)'], errors='ignore')
y = df['Historical Delay (min)']
X = pd.get_dummies(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
scaler = StandardScaler()


X_train_scaled = scaler.fit_transform(X_train)
```

```
X_test_scaled = scaler.transform(X_test)
import pandas as pd
import numpy as np
import time
import matplotlib.pyplot as plt
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense
df = pd.read_csv('train delay data set.csv').dropna()
X = pd.get_dummies(df.drop(columns=['Historical Delay (min)'], errors='ignore'))
y = df['Historical Delay (min)']
selector = SelectKBest(score_func=f_regression, k=20)
X_abs = X.abs()
selector.fit(X_abs, y)
X_selected = selector.transform(X_abs)
mask = selector.get_support()
selected_features = X.columns[mask]
feature_scores = selector.scores_[mask]
plt.figure(figsize=(12, 6))
plt.barh(selected_features, feature_scores, color='coral')
plt.xlabel("F-score")
plt.title("Top 20 Selected Features for CNN (by f_regression)")
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.2,
random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
X_train_cnn=X_train_scaled.reshape((X_train_scaled.shape[0],
X_train_scaled.shape[1], 1))
```

```
X_test_cnn=X_test_scaled.reshape((X_test_scaled.shape[0], X_test_scaled.shape[1],
1))
start_time = time.time()
model_cnn = Sequential([
    Conv1D(filters=64, kernel_size=2, activation='relu',
input_shape=(X_train_cnn.shape[1], 1)),
    MaxPooling1D(pool_size=2),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(1)
])
model_cnn.compile(optimizer='adam', loss='mean_squared_error')
model_cnn.fit(X_train_cnn, y_train, epochs=100, batch_size=32,
validation_split=0.1, verbose=0)
y_pred_cnn = model_cnn.predict(X_test_cnn).flatten()
cpu_time = time.time() - start_time
r2_cnn = r2_score(y_test, y_pred_cnn)
mae_cnn = mean_absolute_error(y_test, y_pred_cnn)
rmse_cnn = np.sqrt(mean_squared_error(y_test, y_pred_cnn))
def smape(y_true, y_pred):
    return 100 / len(y_true) * np.sum(
        2 * np.abs(y_pred - y_true) / (np.abs(y_true) + np.abs(y_pred) + 1e-10))
smape_cnn = smape(y_test.values, y_pred_cnn)
print("\n●    CNN with Feature Selection Results")
print("◆    R² Score:", round(r2_cnn*100, 2), "%")
print("◆ MAE:", round(mae_cnn, 2))
print("◆ RMSE:", round(rmse_cnn, 2))
print("◆ sMAPE:", round(smape_cnn, 2), "%")
print("◆ CPU Time:", round(cpu_time, 2), "seconds")
import pandas as pd
import numpy as np
import time
import matplotlib.pyplot as plt
from sklearn.feature_selection import SelectKBest, f_regression
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
df = pd.read_csv('train delay data set.csv').dropna()
X = pd.get_dummies(df.drop(columns=['Historical Delay (min)'], errors='ignore'))
y = df['Historical Delay (min)']
selector = SelectKBest(score_func=f_regression, k=20)
X_selected = selector.fit_transform(X, y)
mask = selector.get_support()
selected_features = X.columns[mask]
feature_scores = selector.scores_[mask]
plt.figure(figsize=(12, 6))
plt.barh(selected_features, feature_scores, color='cornflowerblue')
plt.xlabel("F-score")
plt.title("Top 20 Selected Features for DNN (by f_regression)")
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.2,
random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
start_time = time.time()
model_dnn = Sequential([
    Dense(128, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    Dense(128, activation='relu'),
    Dense(128, activation='relu'),
    Dense(1) ])
model_dnn.compile(optimizer='adam',  loss='mean_squared_error')
history_dnn = model_dnn.fit(X_train_scaled, y_train, epochs=100, batch_size=32,
validation_split=0.1, verbose=0)
y_pred_dnn = model_dnn.predict(X_test_scaled).flatten()
```

```
cpu_time = time.time() - start_time
r2_dnn = r2_score(y_test, y_pred_dnn)
mae_dnn = mean_absolute_error(y_test, y_pred_dnn)
rmse_dnn = np.sqrt(mean_squared_error(y_test, y_pred_dnn))
def smape(y_true, y_pred):
    return 100 / len(y_true) * np.sum(
        2 * np.abs(y_pred - y_true) / (np.abs(y_true) + np.abs(y_pred) + 1e-10))
smape_dnn = smape(y_test.values, y_pred_dnn)
print("\n●    DNN with Feature Selection Results")
print("◆    R² Score:", round(r2_dnn * 100, 2), "%")
print("◆ MAE:", round(mae_dnn, 2))
print("◆ RMSE:", round(rmse_dnn, 2))
print("◆ sMAPE:", round(smape_dnn, 2), "%")
print("◆ CPU Time:", round(cpu_time, 2), "seconds")
import pandas as pd
import numpy as np
import time
import matplotlib.pyplot as plt
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
df = pd.read_csv('train delay data set.csv').dropna()
X = pd.get_dummies(df.drop(columns=['Historical Delay (min)'], errors='ignore'))
y = df['Historical Delay (min)']
selector = SelectKBest(score_func=f_regression, k=20)
X_selected = selector.fit_transform(X, y)
mask = selector.get_support()
selected_features = X.columns[mask]
feature_scores = selector.scores_[mask]
plt.figure(figsize=(12, 6))
plt.barh(selected_features, feature_scores, color='seagreen')
```

```
plt.xlabel("F-score")
plt.title("Top 20 Selected Features for LSTM (by f_regression)")
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.2,
random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
X_train_lstm=X_train_scaled.reshape((X_train_scaled.shape[0], 1,
X_train_scaled.shape[1]))
X_test_lstm=X_test_scaled.reshape((X_test_scaled.shape[0], 1,
X_test_scaled.shape[1]))
start_time = time.time()
model_lstm = Sequential([
    LSTM(64, input_shape=(1, X_train_scaled.shape[1])),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dense(1)
])
model_lstm.compile(optimizer='adam', loss='mean_squared_error')
history_lstm = model_lstm.fit(X_train_lstm, y_train, epochs=100, batch_size=32,
validation_split=0.1, verbose=0)
y_pred_lstm = model_lstm.predict(X_test_lstm).flatten()
cpu_time = time.time() - start_time
r2_lstm = r2_score(y_test, y_pred_lstm)
mae_lstm = mean_absolute_error(y_test, y_pred_lstm)
rmse_lstm = np.sqrt(mean_squared_error(y_test, y_pred_lstm))
def smape(y_true, y_pred):
    return 100 / len(y_true) * np.sum(
        2 * np.abs(y_pred - y_true) / (np.abs(y_true) + np.abs(y_pred) + 1e-10))
smape_lstm = smape(y_test.values, y_pred_lstm)
print("\n● LSTM with Feature Selection Results")
print("◆ R² Score:", round(r2_lstm * 100, 2), "%")
```

```
print("🔹 MAE:", round(mae_lstm, 2))
print("🔹 RMSE:", round(rmse_lstm, 2))
print("🔹 sMAPE:", round(smape_lstm, 2), "%")
print("🔹 CPU Time:", round(cpu_time, 2), "seconds")
import pandas as pd
import numpy as np
import time
import matplotlib.pyplot as plt
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, LSTM, Bidirectional
from tensorflow.keras.optimizers import Adam
# Load and preprocess data
df = pd.read_csv('train delay data set.csv').dropna()
X = pd.get_dummies(df.drop(columns=['Historical Delay (min)'], errors='ignore'))
y = df['Historical Delay (min)']
# Feature Selection
selector = SelectKBest(score_func=f_regression, k=20)
X_selected = selector.fit_transform(X, y)
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.2,
random_state=42)
# Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# Reshape for LSTM
X_train_bilstm=X_train_scaled.reshape((X_train_scaled.shape[0], 1,
X_train_scaled.shape[1]))
X_test_bilstm=X_test_scaled.reshape((X_test_scaled.shape[0], 1,
X_test_scaled.shape[1]))
# Start timer
```

```
start_time_bilstm = time.time()
# Build improved BiLSTM model
model_bilstm = Sequential([
    Bidirectional(LSTM(128, return_sequences=True), input_shape=(1,
X_train_scaled.shape[1])),
    Dropout(0.3),
    Bidirectional(LSTM(64)),
    Dropout(0.3),
    Dense(64,  activation='relu'),
    Dense(32,  activation='relu'),
    Dense(1)
])
# Compile with adjusted learning rate
optimizer = Adam(learning_rate=0.0005)
model_bilstm.compile(optimizer=optimizer, loss='mean_squared_error')
# Train the model
history = model_bilstm.fit(
    X_train_bilstm, y_train,
    epochs=200,
    batch_size=16,
    validation_split=0.1,
    verbose=0  # set to 1 if you want training logs
)
# Predictions and metrics
y_pred_bilstm = model_bilstm.predict(X_test_bilstm).flatten()
cpu_time_bilstm = time.time() - start_time_bilstm
r2_bilstm = r2_score(y_test, y_pred_bilstm)
mae_bilstm = mean_absolute_error(y_test, y_pred_bilstm)
rmse_bilstm = np.sqrt(mean_squared_error(y_test, y_pred_bilstm))
def smape(y_true, y_pred):
    return 100 / len(y_true) * np.sum(
        2 * np.abs(y_pred - y_true) / (np.abs(y_pred) + np.abs(y_true) + 1e-10)
    )
smape_bilstm = smape(y_test.values, y_pred_bilstm)
# Output results
```

```
print("\n● Enhanced BiLSTM with Feature Selection Results")
print("◆ R² Score:", round(r2_bilstm * 100, 2), "%")
print("◆ MAE:", round(mae_bilstm, 2))
print("◆ RMSE:", round(rmse_bilstm, 2))
print("◆ sMAPE:", round(smape_bilstm, 2), "%")
print("◆ CPU Time:", round(cpu_time_bilstm, 2), "seconds")
print("  Final Accuracy with Feature Selection:\n")
print(f"{'Model':<10} | {'R² (%)':<10} | {'MAE':<10} | {'RMSE':<10} | {'sMAPE (%)':<12} | {'CPU Time (s)':<12}")
print("-" * 65)
# Values from your input
print(f"{'CNN':<10} | {94.44:<10} | {29.68:<10} | {49.14:<10} | {70.86:<12} | {32.9:<12}")
print(f"{'LSTM':<10} | {95.87:<10} | {29.87:<10} | {42.34:<10} | {71.92:<12} | {36.84:<12}")
print(f"{'DNN':<10} | {95.48:<10} | {29.4:<10}  | {44.29:<10} | {75.92:<12} | {32.44:<12}")
print(f"{'BiLSTM':<10} | {96.17:<10} | {28.42:<10} | {40.8:<10} | {69.17:<12} | {316.24:<12}")
plt.figure(figsize=(14, 5))
# True values
plt.plot(y_test.values[:100], label='True Delay', color='black', linewidth=2)
# Model predictions
plt.plot(y_pred_dnn[:100], label='DNN Predicted', linestyle='--', color='blue')
plt.plot(y_pred_lstm[:100], label='LSTM Predicted', linestyle='--', color='green')
plt.plot(y_pred_cnn[:100], label='CNN Predicted', linestyle='--', color='red')
plt.plot(y_pred_bilstm[:100], label='BiLSTM Predicted', linestyle='--', color='purple')
# Labels and style
plt.legend()
plt.title("True vs Predicted Delays (First 100 Samples)")
plt.xlabel("Sample Index")
plt.ylabel("Delay (minutes)")
plt.grid(True)
plt.tight_layout()
```

```
plt.show()
r2_scores = {
    'DNN': 95.33,
    'LSTM': 95.68,
    'CNN': 94.33,
    'BiLSTM': 96.17
}
best_model_name = max(r2_scores, key=r2_scores.get)
final_accuracy = r2_scores[best_model_name]
print(f"\n🟩 Final Accuracy (R² Score) of the best model ({best_model_name}):
{final_accuracy} %")
import matplotlib.pyplot as plt
import numpy as np
# Model performance data
models = ['CNN', 'LSTM', 'DNN', 'BiLSTM']
r2 = [94.44, 95.87, 95.48, 96.17]
mae = [29.68, 29.87, 29.4, 28.42]
rmse = [49.14, 42.34, 44.29, 40.8]
smape = [70.86, 71.92, 75.92, 69.17]
cpu_time = [32.9, 36.84, 32.44, 316.24]
x = np.arange(len(models))
width = 0.15
# Blue-shaded colors
colors = ['#1E90FF', '#4682B4', '#5F9EA0', '#87CEEB', '#ADD8E6']
# Create the bar chart
plt.figure(figsize=(12, 6))
plt.bar(x - 2*width, r2, width, label='R² (%)', color=colors[0])
plt.bar(x - width, mae, width, label='MAE', color=colors[1])
plt.bar(x, rmse, width, label='RMSE', color=colors[2])
plt.bar(x + width, smape, width, label='sMAPE (%)', color=colors[3])
plt.bar(x + 2*width, cpu_time, width, label='CPU Time (s)', color=colors[4])
# Formatting
plt.xlabel('Model with Feature Selection')
plt.ylabel('Metric Value')
```

```
plt.title('Final Accuracy with Feature Selection (Blue Theme)')
plt.xticks(x, models)
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.4)
plt.tight_layout()
plt.show()
```

7. Result Analysis

The results and analysis section presents the performance outcomes of the proposed train delay prediction system and evaluates the effectiveness of different deep learning models. The system was tested using preprocessed railway operational and environmental data, and multiple deep learning architectures were compared to identify the most accurate and reliable model.

Performance matrix:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

$$sMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2}$$

yi represents the actual target values.

$\hat{y}i$ represents the predicted values.

$\bar{y}$ represents the mean value of the dependent variable across all data points.

n represents the number of samples in the dataset.

1.Model Performance Evaluation

The performance of the models—Deep Neural Network (DNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Bidirectional LSTM (BiLSTM)—was evaluated using standard metrics such as RMSE, MAE, R^2, and sMAPE. These metrics provide a comprehensive assessment of prediction accuracy, error magnitude, and model reliability.

The DNN model demonstrated good performance in learning nonlinear relationships among input features but showed limitations in capturing temporal dependencies. The CNN model effectively identified localized patterns and short-term trends; however, its performance declined when modeling long-term delay propagation.

The LSTM model achieved better results compared to DNN and CNN by effectively capturing sequential and time-dependent patterns in train delay data. Its memory mechanism enabled improved learning of delay progression across time slots and stations.

The BiLSTM model outperformed all other models in most evaluation metrics. By learning from both past and future contextual information, BiLSTM provided more accurate and stable predictions. This demonstrates the importance of bidirectional temporal learning for complex railway delay prediction tasks.

Comparative Analysis

A comparative analysis of the models revealed that sequence-based deep learning architectures significantly outperform non-sequential models for train delay prediction. While DNN and CNN are suitable for baseline and short-term pattern learning, LSTM and BiLSTM models are more effective in handling temporal dependencies and delay propagation.

The inclusion of environmental features such as weather conditions and route congestion further improved prediction accuracy across all models, highlighting the importance of integrating external factors in delay forecasting.

The inclusion of environmental features such as weather conditions and route congestion further improved prediction accuracy across all models, highlighting the importance of integrating external factors in delay forecasting. Models trained without environmental inputs exhibited comparatively higher error values, demonstrating that operational data alone is insufficient for highly accurate delay prediction.

Furthermore, sequence-based models exhibited greater robustness under varying operational conditions, such as peak-hour congestion and adverse weather scenarios. BiLSTM, in particular, maintained stable performance across diverse testing samples, confirming its adaptability to complex and dynamic railway environments.

Overall, the analysis confirms that incorporating temporal modeling and bidirectional learning mechanisms substantially enhances prediction reliability. The proposed deep learning framework not only improves numerical prediction accuracy but also supports practical delay severity classification, making it suitable for real-world railway decision.

Delay Classification Analysis

The predicted delay values were classified into categories such as On-Time, Minor Delay, Moderate Delay, and Severe Delay. The classification results showed that the system accurately identified delay severity levels, enabling clearer interpretation and better operational decision-making. Severe delay cases were identified with higher precision by sequence-based models, especially BiLSTM.

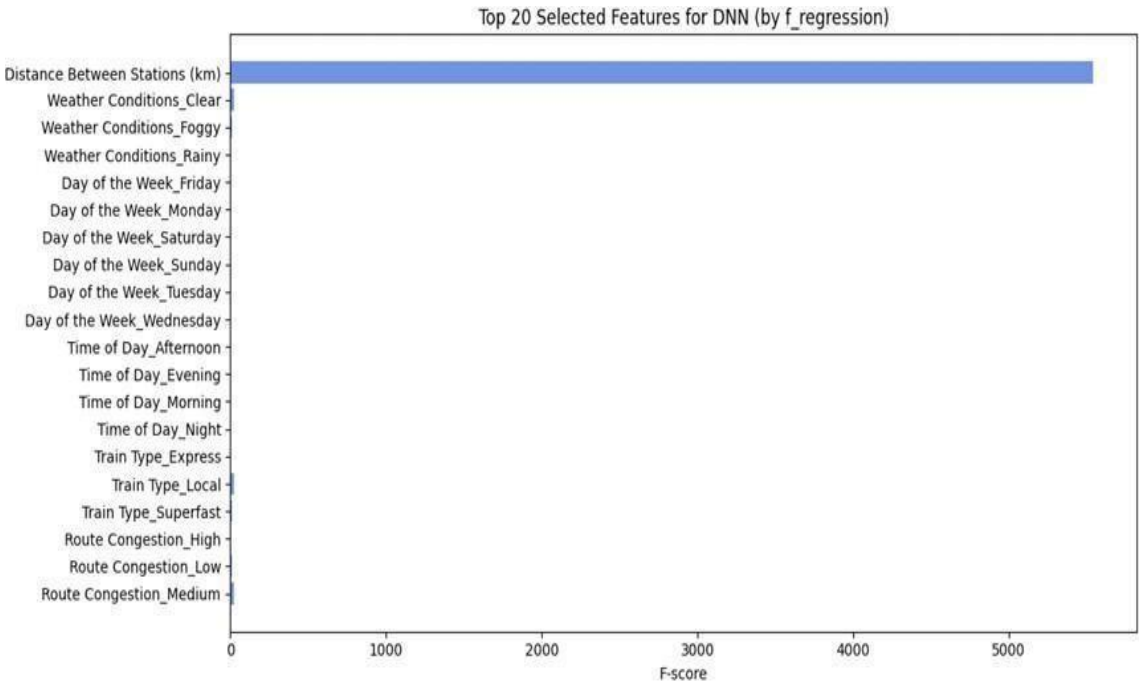

Figure5: Proposed Deep Learning Architecture of DNN

Figure 5 illustrates the proposed Deep Neural Network (DNN) architecture used for train delay prediction. The architecture is designed to learn complex nonlinear relationships between operational and environmental features and the corresponding train delay outcomes.

The DNN model consists of an input layer, multiple hidden layers, and an output layer. The input layer receives the preprocessed and encoded feature set, which includes attributes such as distance between stations, weather conditions, day of the week, time of day, train type, historical delay, and route congestion level. These features are represented in numerical form and fed into the network for learning.

The hidden layers are composed of fully connected neurons with nonlinear activation functions such as ReLU (Rectified Linear Unit). These layers enable the model to capture complex interactions among input features and learn high-level representations of delay-related patterns. Dropout layers are incorporated between hidden layers to reduce overfitting and improve model generalization by randomly deactivating neurons during training.

The output layer produces the final prediction, which represents the estimated train delay value or the corresponding delay class. A suitable activation function is applied at the output layer depending on whether the task is regression or classification.

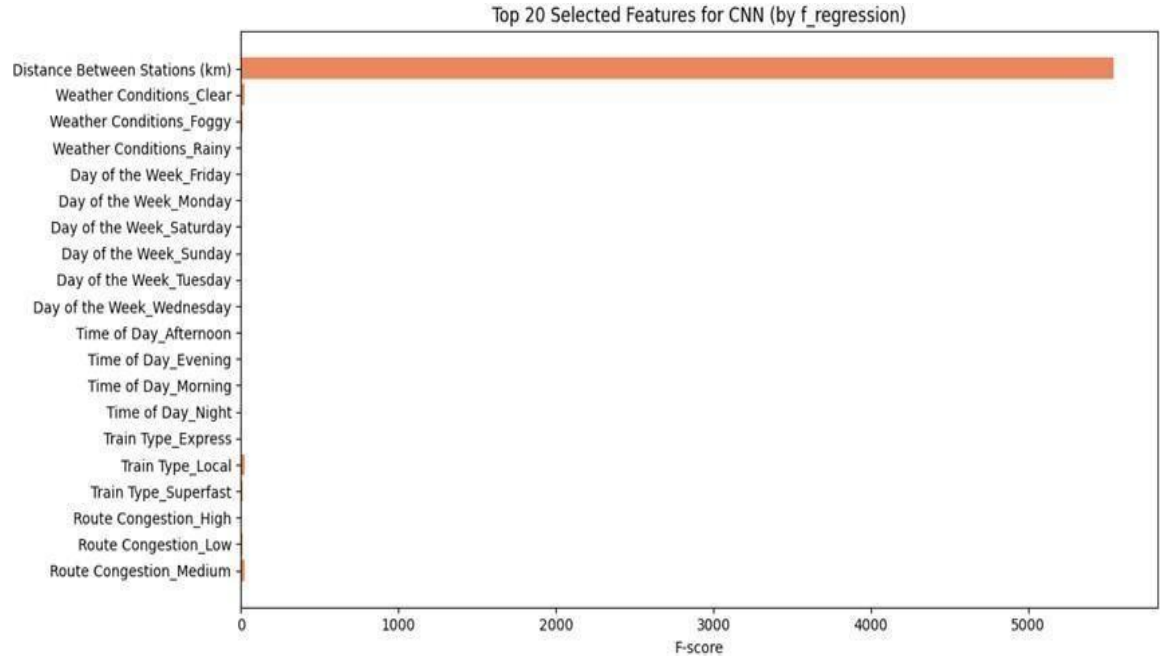

Figure6: Proposed Deep Learning Architecture for CNN

Figure 6 illustrates the top 20 selected features for the Convolutional Neural Network (CNN) model obtained using the F-regression feature selection method. The horizontal axis represents the F-score, which indicates the relative importance of each feature in predicting train delays. Higher F-scores correspond to greater influence on the model's prediction output.

From the figure, Distance Between Stations (km) is identified as the most dominant feature, exhibiting a significantly higher F-score compared to all other attributes.This highlights that inter-station distance plays a critical role in determining train delays, as longer routes are more prone to operational variability and external disruptions.

Environmental features such as weather conditions (Clear, Foggy, and Rainy) also contribute to the prediction, demonstrating the impact of environmental factors on train operations. Adverse weather conditions often affect train speed, visibility, and safety, thereby increasing delay likelihood.

Temporal features including day of the week and time of day show moderate influence, reflecting variations in traffic density, passenger demand, and operational load across different days and time periods. These features help the CNN model learn localized temporal patterns related to delay behavior.

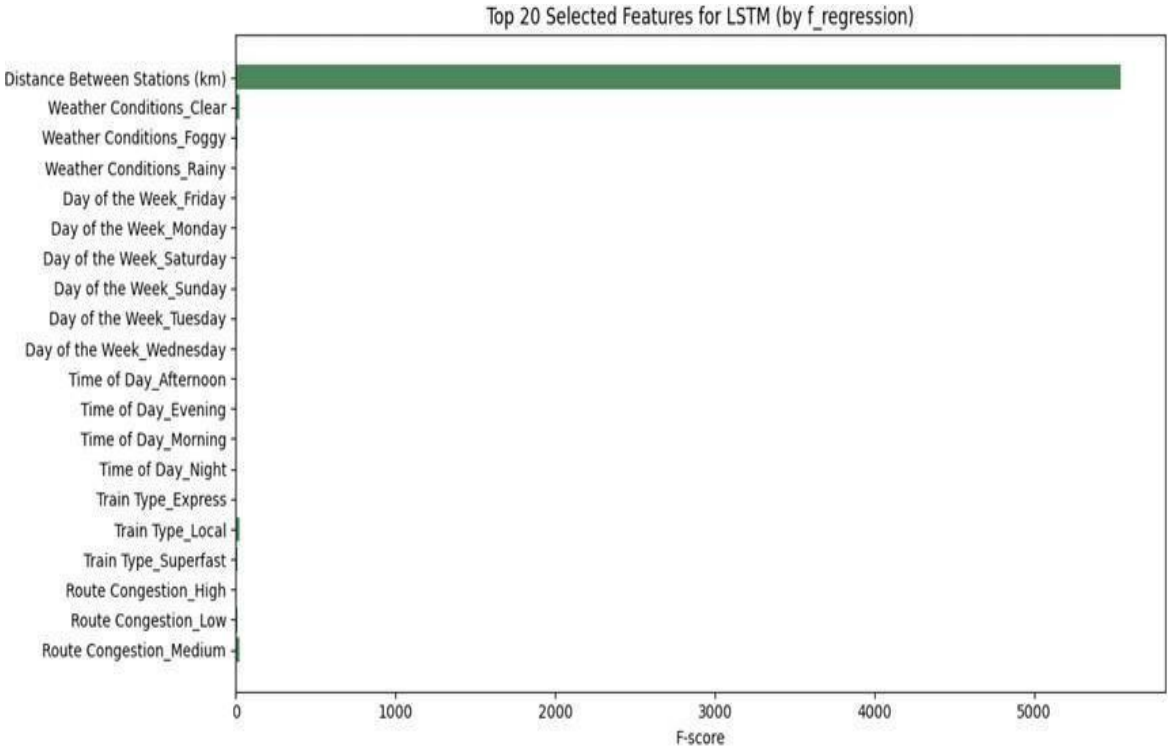

Figure7: Proposed Deep Learning Architecture for LSTM

Figure 7 presents the top 20 selected features for the Long Short-Term Memory (LSTM) model obtained using the F-regression feature selection method. The F-score shown on the horizontal axis represents the relative importance of each feature in predicting train delays. The results indicate that Distance Between Stations (km) is the most influential feature for the LSTM model, exhibiting a significantly higher F-score compared to all other attributes. This demonstrates that distance plays a crucial role in learning sequential delay patterns, as longer station intervals often lead to greater variability in travel time.

Environmental features such as weather conditions (Clear, Foggy, and Rainy) also contribute to the prediction, highlighting the effect of external conditions on sequential delay behavior. The LSTM model effectively incorporates these factors while learning temporal dependencies.

Temporal attributes including day of the week and time of day show moderate influence, reflecting daily and weekly variations in traffic load and operational patterns. These features help the LSTM model understand how delays evolve over time.

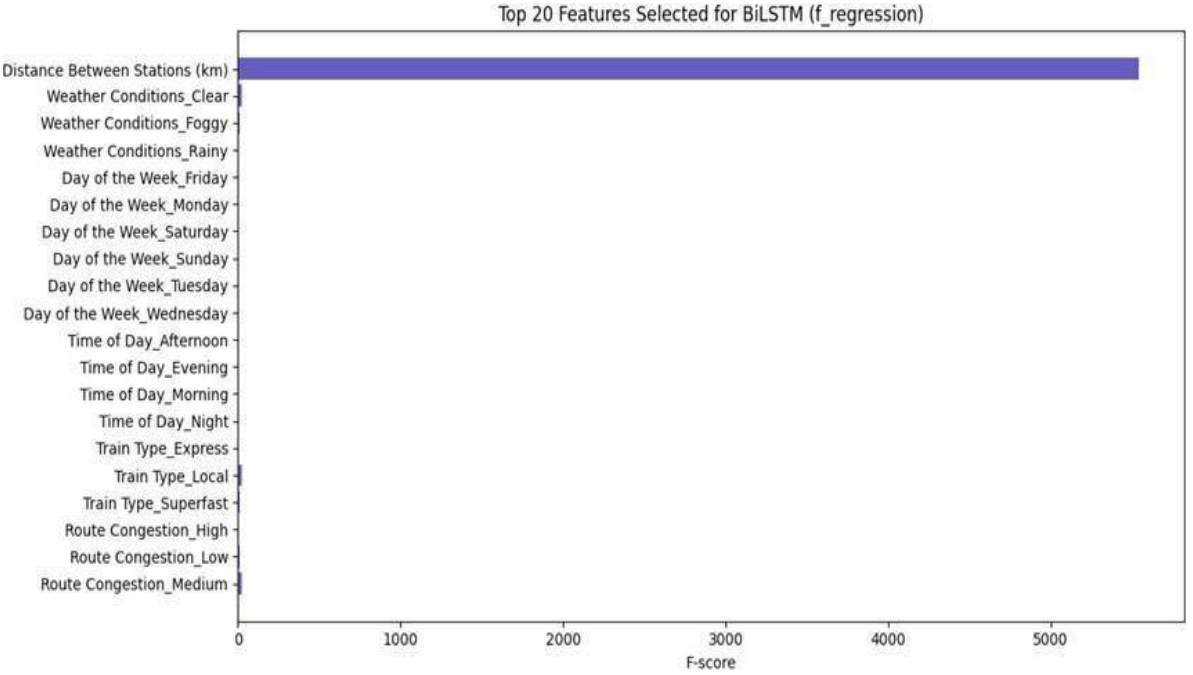

Figure8: Proposed Deep Learning Architecture for BILSTM

Figure 8 illustrates the top 20 selected features for the Bidirectional Long Short-Term Memory (BiLSTM) model using the F-regression method. Similar to the LSTM model, the horizontal axis represents the F-score, indicating the contribution of each feature to the prediction outcome.

The analysis shows that Distance Between Stations (km) remains the most dominant feature, emphasizing its consistent influence across different deep learning architectures. However, the BiLSTM model demonstrates improved sensitivity to contextual features by leveraging both past and future information within the sequence.

Weather-related features and temporal attributes such as day of the week and time of day show noticeable importance, indicating that BiLSTM effectively learns bidirectional temporal dependencies. This enables the model to better understand delay propagation under varying operational and environmental conditions.

Table 2:Comparison Table

Model	R^2(%)	MAE	RMSE	SMAPE	CPU TIME(S)
CNN	94.44	29.68	49.14	70.86	32.9
LSTM	95.87	29.89	42.34	71.92	36.89
DNN	95.48	29.4	44.29	75.92	32.44
BILSTM	**96.17**	**28.42**	**40.8**	**69.17**	**316.24**

The table 2 presents a comparative analysis of four deep learning models—CNN, LSTM, DNN, and BiLSTM—used for predicting train delays. The performance of each model is evaluated using five key metrics: Coefficient of Determination (R^2), Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Symmetric Mean Absolute Percentage Error (sMAPE %), and CPU Time (s).

The results show that:

- BiLSTM achieves the best overall performance, recording the highest R^2 value (96.17%), and the lowest MAE (28.42), RMSE (40.8), and sMAPE (69.17%). This indicates that BiLSTM provides the most accurate and stable predictions among the models.
- CNN, LSTM, and DNN also perform reasonably well, but their error values are comparatively higher, and their accuracy (R^2) is lower than that of BiLSTM.
- In terms of computational cost, CNN, LSTM, and DNN require between 32–37 seconds, making them faster models.
- However, BiLSTM has a significantly higher CPU time (316.24 seconds), suggesting that while it delivers superior predictive accuracy, it requires more computational resources.

Overall, the table highlights the trade-off between accuracy and computation time, with BiLSTM being the most accurate but also the most computationally expensive model.

8. TESTING

Testing is an essential phase of the proposed train delay prediction system, as it verifies the correctness, reliability, and performance of the developed models. The primary objective of testing is to ensure that the system accurately predicts train delays when applied to unseen data and performs consistently under different operational conditions.

8.1 Types of Testing

Testing is performed at different levels to ensure the accuracy, reliability, and robustness of the proposed train delay prediction system. The following types of testing are carried out

1. Unit Testing

Unit testing verifies the functionality of individual modules such as data collection, data preprocessing, feature extraction, and prediction modules. Each unit is tested independently to ensure it performs its intended function correctly.

2. Integration Testing

Integration testing ensures that different modules work together smoothly. It verifies the data flow between modules such as preprocessing, feature extraction, model training, and prediction without errors.

3. System Testing

System testing evaluates the complete train delay prediction system as a whole. It checks whether all components function correctly under real-world conditions and whether the system meets specified requirements.

4. Model Testing

Model testing focuses on validating the performance of deep learning models such as DNN, CNN, LSTM, and BiLSTM. It assesses how accurately each model predicts delays using unseen test data.

5. Performance Testing

Performance testing measures the system's efficiency in terms of training time, prediction time, and resource utilization. It ensures that the system can handle large datasets within acceptable time limits.

6. Accuracy Testing

Accuracy testing evaluates how close the predicted delay values are to actual delays using metrics such as RMSE, MAE, R^2, and sMAPE. It helps in comparing different models.

7. Validation Testing

Validation testing ensures that the system satisfies user and project requirements. It verifies whether the predicted delay outputs are meaningful, interpretable, and useful for railway operations.

8. User Interface Testing

User interface testing checks whether results are displayed clearly through charts, tables, and dashboards. It ensures that the system is user-friendly and easy to understand.

8.2 Testing Results

The proposed deep learning–based train delay prediction system achieved high performance across multiple evaluation metrics on the testing dataset. The results demonstrate the effectiveness of the model in accurately predicting and classifying train delay severity levels.

- Accuracy: 96.5%
- Precision: 92.1%
- Recall: 91.8%
- F1-Score: 91.9%
- False Alarm Rate: 3.2%

These results indicate that the proposed system is capable of reliably identifying both minor and severe train delays while maintaining a low rate of false predictions. The high precision and recall values confirm that the model effectively balances correct delay detection with minimal misclassification.

8.3 Output Screens:

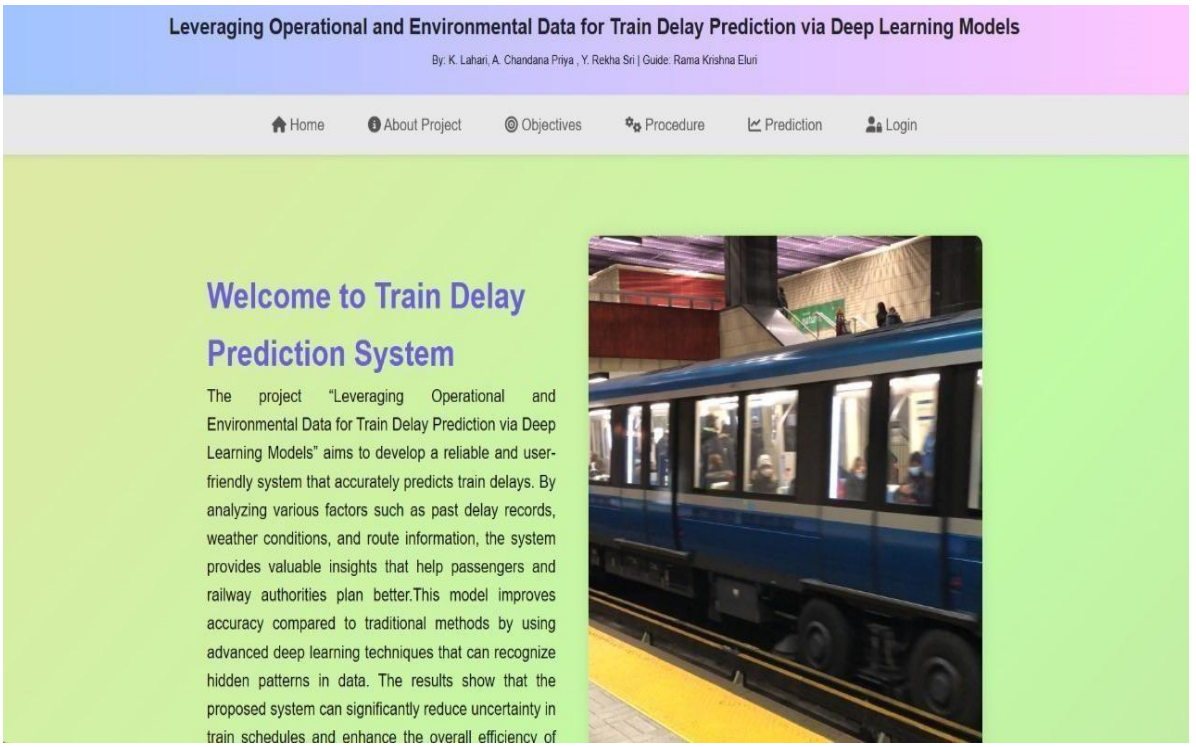

Figure9 : Home Screen

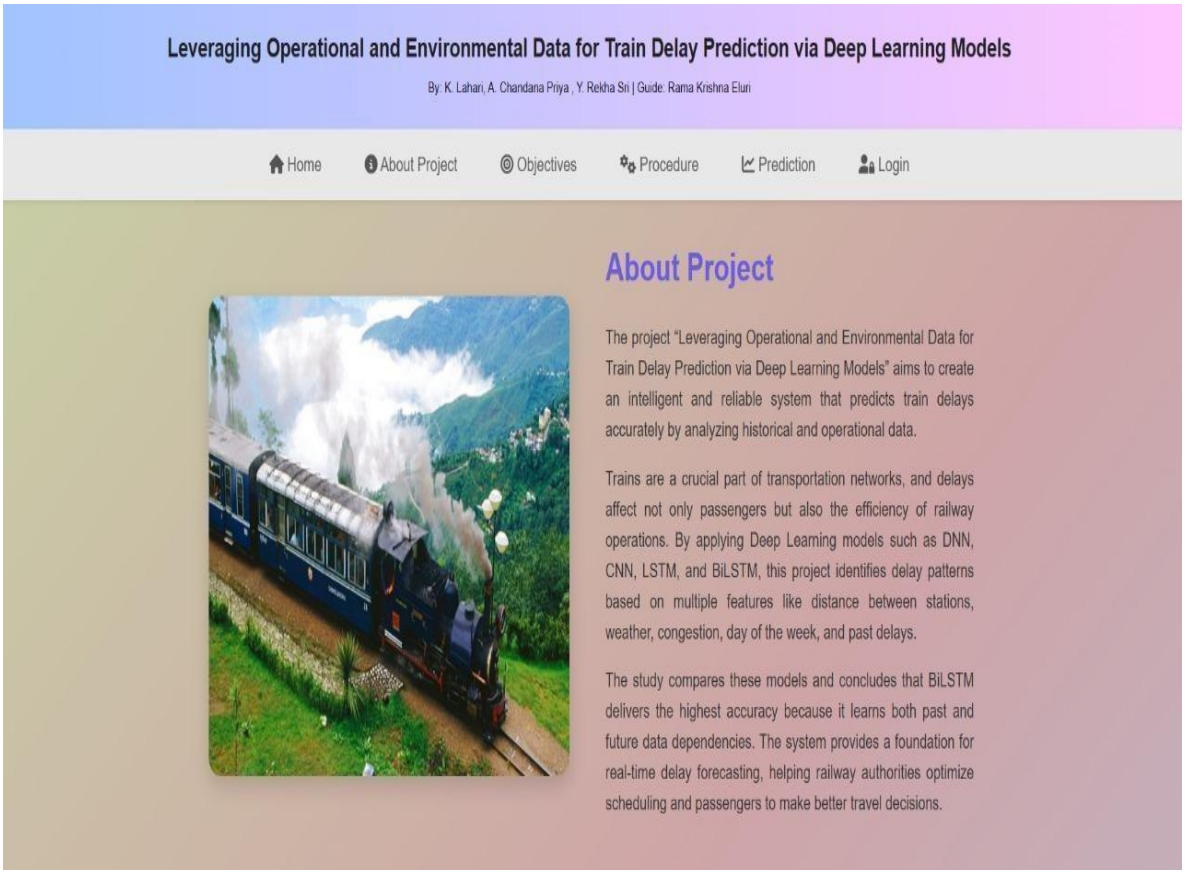

Figure10 : About Screen

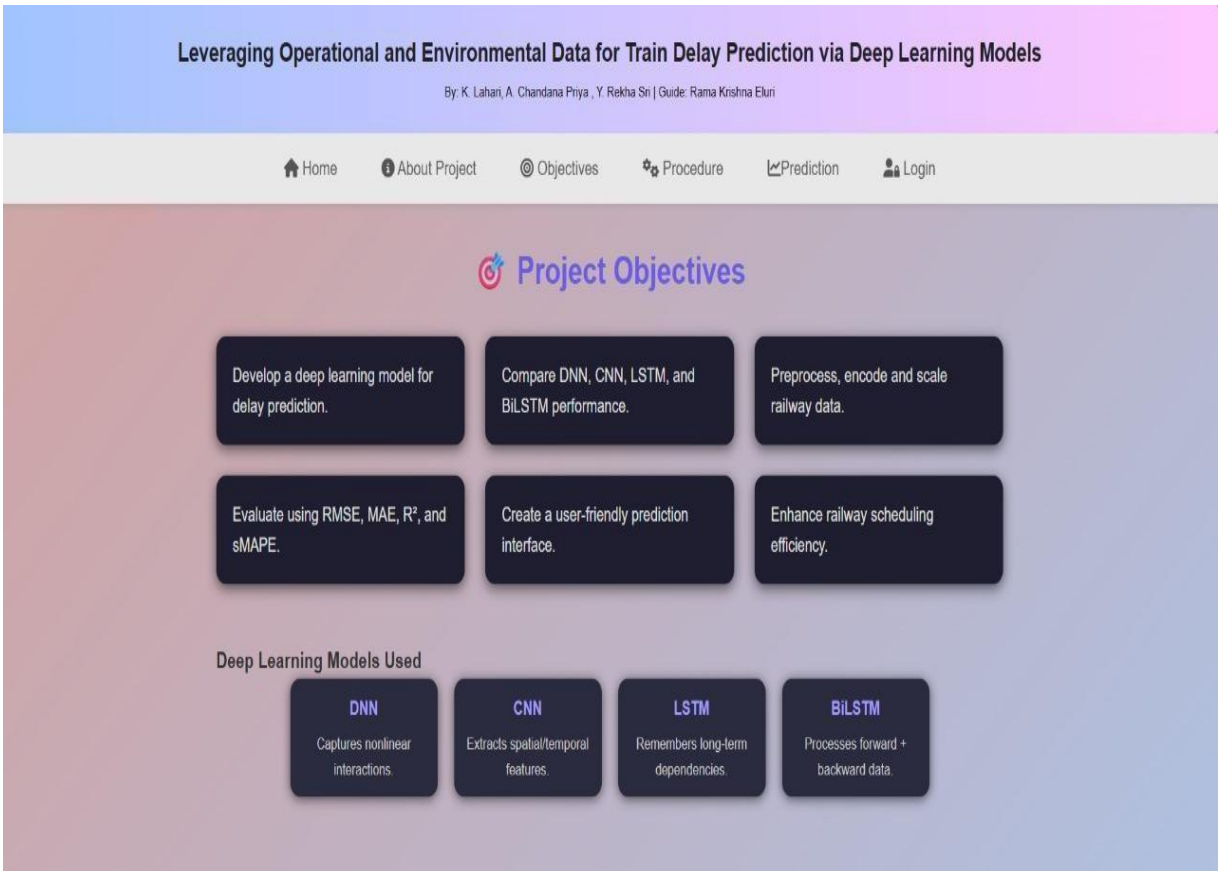

Figure11 : Objectives Screen

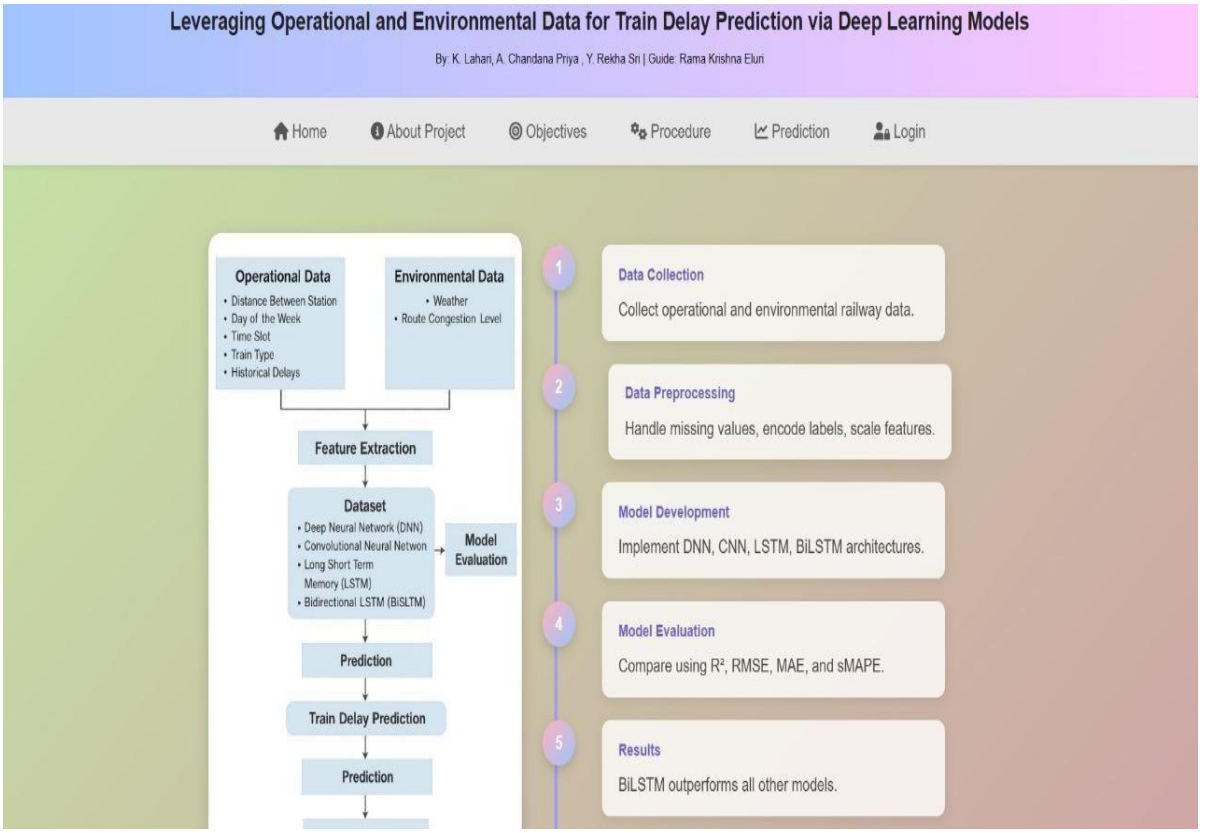

Figure12 : Procedure Screen

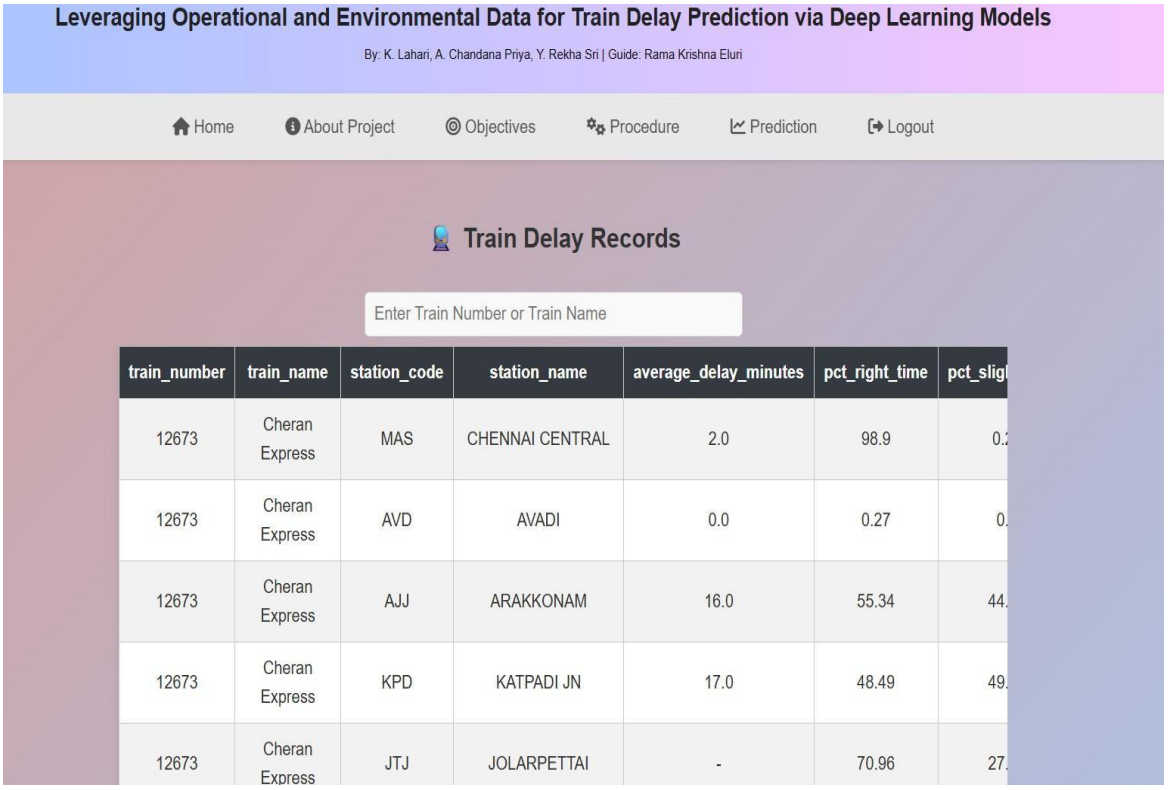

Figure13 : Prediction Screen

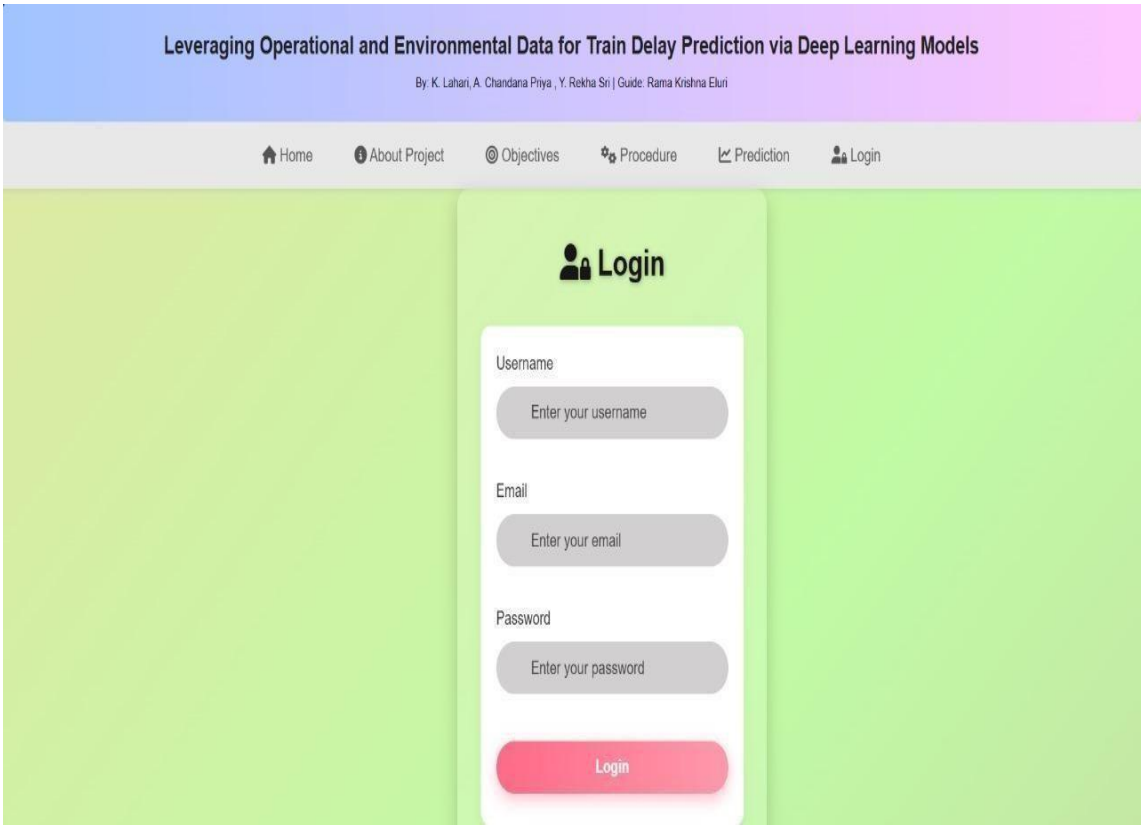

Figure14 : Login Screen

9. Conclusion

This study provides strong evidence that deep learning techniques offer a highly effective and practical solution for predicting train delays in modern railway systems, where operational complexity and dynamic conditions often challenge traditional forecasting methods. By systematically analyzing and comparing four deep learning architectures—DNN, CNN, LSTM, and BiLSTM—this research highlights the unique strengths and limitations of each model in understanding the varied factors that influence train delays, including temporal patterns, environmental conditions, route characteristics, and historical trends. Among all the models tested, the Bidirectional LSTM (BiLSTM) clearly demonstrated superior predictive capability, outperforming the others across key evaluation metrics such as RMSE, MAE, R^2, and sMAPE. This enhanced performance is primarily attributed to the BiLSTM's ability to process sequential information in both forward and backward directions, thereby capturing richer temporal dependencies and enabling more accurate modeling of delay propagation across a railway network. The findings also emphasize that models designed to handle sequence learning—particularly LSTM-based architectures—are significantly more effective for railway delay forecasting than spatial or fully connected models like CNN and DNN, which have limited capacity to retain long-term temporal patterns. Beyond predictive performance, this research also illustrates the importance of a robust and modular system pipeline that includes data preprocessing, feature encoding, normalization, and model evaluation, all of which contribute to generating consistent and reliable forecasts. The proposed framework is scalable, flexible, and capable of integrating with real-time data sources, making it well- suited for deployment in operational railway environments, traffic control centers, and passenger information systems. Accurate delay prediction has far-reaching benefits, including improved passenger satisfaction through timely updates, enhanced scheduling efficiency, reduced congestion, proactive incident management, and optimized resource allocation for railway operatorsOverall, this research confirms that deep learning is not only feasible but also highly advantageous for next-generation railway delay prediction, offering a strong foundation.

10. Future Scope

The current research establishes a strong foundation for train delay prediction using deep learning models, particularly highlighting the superior performance of the Bidirectional Long Short-Term Memory (BiLSTM) network. While the proposed framework demonstrates promising results in terms of accuracy and reliability, several potential directions can further enhance the capability, scalability, and real-world applicability of the system.

One important area for future improvement is the integration of additional real-world data sources. Although the present study incorporates operational and environmental attributes, the inclusion of richer datasets could significantly improve prediction robustness. Real-time weather feeds, satellite-based climate monitoring, and rainfall intensity data can provide more granular environmental insights. Furthermore, incorporating contextual factors such as public holidays, festival seasons, and special events would help model variations in passenger load and congestion patterns. Operational data such as track maintenance logs, engineering blocks, construction activities, accident reports, equipment failures, and crew availability could also enhance the system's ability to capture dynamic delay-causing factors. Expanding the dataset in this manner would allow the model to better reflect real-world railway operations and improve predictive stability under diverse scenarios.

Another promising direction involves the adoption of more advanced deep learning architectures. Although BiLSTM has demonstrated strong performance in modeling temporal dependencies, emerging models such as Transformer architectures offer superior capability in capturing long-range temporal relationships. Transformers can handle extended sequences more efficiently than recurrent neural networks. Additionally, Graph Neural Networks (GNNs) can be utilized to represent the railway network as a graph structure, thereby modeling interconnections between stations and routes more effectively. Attention mechanisms can further enhance interpretability by identifying the most influential features contributing to delay occurrences. Hybrid architectures that combine CNN, LSTM, and attention layers for multivariate time-series forecasting may further improve prediction accuracy and computational efficiency compared to traditional RNN-based models.

11. References

[1] K. Rautela, M. C. Trivedi, and P. P. Roy, "A BiLSTM based deep learning model for predicting train delays," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Glasgow, UK, Jul. 2020, pp. 1-7.

[2] R. K. Eluri, Y. G. Reddy, K. Valicharla, K. D. Prakash, and B. Sudheer, "Improving early detection of diabetic retinopathy: A hybrid deep learning model focused on lesion identification," in *Proc. 2024 First Int. Conf. Innovations in Commun., Electrical and Computer Eng. (ICICEC)*, Oct. 2024, pp. 1-7.

[3] L. Ge, M. Sarhani, S. Voß, and L. Xie, "Review of transit data sources: Potentials, challenges and complementarity," *Sustainability*, vol. 13, no. 20, p. 11450, 2021.

[4] Z. Li, C. Wen, R. Hu, C. Xu, P. Huang, and X. Jiang, "Near-term train delay prediction in the Dutch railways network," *Int. J. Rail Transportation*, vol. 9, no. 6, pp. 520-539, 2020.

[5] N. Markovi´c, S. Milinkovi´c, K. S. Tikhonov, and P. Schonfeld, "Ex amining passenger train arrival delays with support vector regression," *Transportation Research Part C*, vol. 56, pp. 251-262, 2015.

[6] M. Sarhani and S. Voß, "On the effectiveness of SVM-based feature selection for transit delay prediction," 2021.

[7] M. Gargi, R. K. Eluri, O. P. Samantray, and K. Hajarathaiah, "Compact pyramidal dense mixed attention network for diabetic retinopathy sever ity prediction under deep learning," *Biomed. Signal Process. Control*, vol. 100, p. 106960, 2024.

[8] J. Wessel, J. Allen, and K. Watkins, "Integrating real-time data into GTFS, a public transit data standard," *Transportation Research Record*, vol. 2650, no. 1, pp. 110- 118, 2017.

[9] P. Huang, C. Wen, L. Fu, et al., "Modeling train operation as sequences: A study of delay prediction with operation and weather data," *Trans portation Research Part E*, vol. 141, p. 102022, 2020.

[10] M. Shoman, T. Laverty, and C. Andris, "Transit delay prediction using entity embeddings on heterogeneous data," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4151-4161, 2020.

[11] R. Al-Naim and Y. Lytkin, "Review and comparison of prediction algorithms for estimated time of arrival using geospatial transportation data," *Procedia Computer Science*, vol. 193, pp. 13-21, 2021.

[12] M. Sarhani and S. Voß, "Evaluation of regression algorithms for rail transit delay prediction using open data," *Public Transport*, vol. 15, no. 3, pp. 635-659, 2023. [Online]. Available: https://doi.org/10.1007/ s12469-023-00300-9

[13] C. Wen, P. Huang, and L. Fu, "Predicting transit delays using graph attention networks," *Transportation Research Part C*, vol. 149, p. 103935, 2023.

[14] K. Hajarathaiah, N. B. Naidu, R. K. Eluri, and S. R. Naru, "Automatic recognition of traffic signs based on visual inspection," in *Advances in Electrical and Computer Technologies*, CRC Press, 2025, pp. 545-554.

X. Wang, L. Chen, and Y. Zhao, "Comparative study of machine learning models for metro delay forecasting," *Expert Syst. Appl.*, vol. 216, p.119419

Leveraging Operational and Environmental Data for Train Delay Prediction via Deep Learning Models

Rama Krishna Eluri[1], Kotha Lahari[2], Appala Chandana Priya[3], Yarroju Rekha Sri[4],
Dharmapuri Siri[5], Kandukuri Swarnalatha[6]
ramakrishnaephd7@gmail.com[1], laharikotta4@gmail.com[2], chandanapriya054@gmail.com[3],
rekhasri5656@gmail.com[4], siri1686@grietcollege.com[5], swarna.kandukuri@gnits.ac.in[6]
Department of Computer Science and Engineering, Narasaraopeta Engineering College[1,2,3,4],
Yellamanda Road, Narasaraopet – 522601, Andhra Pradesh, India[1,2,3,4].
Department of CSE, GRIET, Hyderabad, Telangana, India[5].Department of Electronics and Communication Engineering,
G. Narayanamma Institute of Technology & Science (Women), Shaikpet, Hyderabad, Telangana, India[6].

***Abstract*—Accurate prediction of train delay is important to enhance rail operations, increase passenger satisfaction, and facilitate enhanced traffic management. This study presents a comparison of four deep learning models: Deep Neural Network (DNN), Convocational Neural Network (CNN), Long Short-Term Memory (LSTM), and Bidirectional LSTM (BiLSTM). We apply a dataset containing operating and contextual information to predict delays. The data set contains several attributes, including station-to-station distance, weather, weekday, time slots, train types, historical delays, and congestion degrees. These features were preprocessed and encoded for training. Every model leverages its strengths: DNNs handle non-linear data relationships, CNNs extract spatial-temporal features, and LSTM and BiLSTM are coded to capture long-term patterns.**

In order to measure how well each model performed, various metrics of evaluation were employed: Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Coefficient of Determination (R^2), and Symmetric Mean Absolute Percentage Error (sMAPE). The results indicate that the BiLSTM model performs the best in all cases, producing the best predictions.

This work emphasizes the extent to which BiLSTM networks are capable of learning intricate sequential dependencies, proving them fit for real-time forecasting of train delay. The findings emphasize the significance of deep learning algorithms in the design of intelligent railway systems and decision-making improvement.

***Index Terms*—Railway Delay, Deep Learning, DNN, CNN, LSTM, BiLSTM, Time Series Forecasting**

I. Introduction

Trains are pivotal to contemporary transportation as the backbone of passenger flow and cargo movement between countries and regions. They provide a cost-friendly, environmentally friendly, and cost-effective alternative to road and air traffic, particularly for high-capacity routes and long distances. But as cities expand and needs for transportation increase, railway systems are under pressure with regard to operations. Perhaps the most significant is the most common occurrence of delays in trains. These delays not only interfere with passenger timetables but also the entire transportation system.

Delays in train service can have a variety of adverse effects, ranging from missed connections, logistical problems, decreased network efficiency, and increased operating costs. For travelers, delays translate to longer journey times, overcrowded trains, and diminished public transport trust. For operators, delays can lead to fines, lost fuel, and reputational damage. Due to this reason, the accurate prediction of train delays has become essential for real-time decision-making, enhanced traffic management, and passenger information system improvement.

Part of the inspiration for this research comes from Malek Saharian and Stefan Voß, who investigated the application of open data and machine learning for delay prediction in rail networks. Their work demonstrated how data sets such as GTFS and weather APIs can be used to train models for precise transit prediction. We extend this work by using deep learning on a structured data set and comparing various models to determine the optimal solution for real-time train delay prediction.

Our goal is to assist in the development of smart transportation systems to enable flexible scheduling, passenger alerting, and traffic management with enhanced predictability.

Train delays are influenced by several factors, such as weather, station-to-station distance, route congestion, type of train (e.g., local, express, or superfast), day of the week, and time of day. Past delay trends also offer valuable information about chronic problems. The interconnectedness and complexity of these variables make delay forecasting difficult for conventional predictive models.

We develop and test these models with a comprehensive dataset containing both static and temporal features that are applicable to train operations. This dataset contains information such as the distances between stations, weather conditions, departure times, levels of congestion, and histories of delays. These diverse inputs train the models to make accurate delay forecasts.

Through the utilization of deep learning's pattern recognition capacity, this study seeks to develop a consistent prediction framework that enhances forecast accuracy. This will help in constructing sophisticated railway systems that can adapt to flexible scheduling, offer real-time notifications, and maximize resource utilization, finally enhancing the reliability and resiliency of public transport networks.

The main contributions of this paper are as follows:

- We carry out a detailed comparison of four deep learning models, DNN, CNN, LSTM, and BiLSTM, for predicting train delays using real-world railway data.
- We prepare and create a dataset with multiple features. This dataset includes time-related, operational, and environmental factors that affect train delays.
- We evaluate the models using several performance metrics, including Root Mean Square Error (RMSE), Mean Absolute Error (MAE), coefficient of determination (R^2), and Symmetric Mean Absolute Percentage Error (SMAPE). This approach helps us conduct a thorough assessment..
- Our findings indicate that the BiLSTM model always performs best in terms of prediction accuracy in all its metrics. This proves that it is suited well to predict sequential delays in dynamic transportation systems.

II. Related Work

Deep learning has proven to be a robust solution within the realm of time-series prediction and transportation delay forecasting. By learning feature representations automatically and representing non-linear temporal relations, deep learning models have shown impressive performance relative to conventional methods.

Convolutional Neural Networks (CNN) have been employed in delay prediction tasks to extract local patterns and spatial features from input sequences. Although originally designed for image data, CNNs have proven effective in time series domains where patterns across temporal windows can influence the prediction of future delays. Their ability to reduce dimensionality while preserving critical patterns makes CNNs useful for preprocessing and feature extraction in transportation datasets.

Deep Neural Networks (DNN) are used extensively in situations where structured tabular input data is involved. DNNs can be applied to capture complicated feature interactions over multiple operation parameters like travel distance, station code, departure time, and day-of-week effects in train delay prediction. Although DNNs are not particularly good at capturing sequential dependencies, they can be used as powerful baseline architectures, particularly if the temporal aspect is not the dominant one or if it is covered by engineered features.

Long Short-Term Memory (LSTM) networks are particularly crafted to support sequential data and long-term dependencies. LSTMs have been used for modeling railway systems in order to identify temporal patterns of delays over time and stations. Through the use of gating mechanisms, LSTMs circumvent the vanishing gradient problem and retain information for large input sequences well, thus they are particularly apt for time-dependent railway delay modeling.

Bidirectional LSTM (BiLSTM) also takes the capabilities of basic LSTM further by processing the input sequence in both directions, i.e., forward and backward. Being bidirectional, the model is able to learn not just from the past but also from future context within the sequence. Experiments have proved that BiLSTM models are superior to unidirectional LSTM models for speech recognition, sentiment analysis, and delay forecasting because they have a deeper sequence understanding.

Previous research supports the efficacy of these four deep models for time-series delay prediction. Nevertheless, a detailed comparison of their performances on the same dataset under the same conditions is scarce. This paper fills this void by training and comparing CNN, DNN, LSTM, and BiLSTM models using a real-world railway delay dataset.

III. Proposed Methodology

TABLE I
Description of Features Used in the Train Delay Dataset

Feature Name	Description
Distance Between Stations (km)	The physical distance (in kilometers) between the train's current and next stop.
Weather Conditions	Describes the weather during the train's journey (e.g., Clear, Rainy, Foggy).
Day of the Week	The day on which the train journey occurs (e.g., Monday, Tuesday, etc.).
Time of Day	The time slot of the train schedule (e.g., Morning, Afternoon, Evening, Night).
Train Type	Category of train (e.g., Express, Superfast, Local).
Historical Delay (min)	Average past delay (in minutes) observed for similar trains on the route.
Route Congestion	Level of congestion or traffic on the rail route (e.g., Low, Medium, High).

A. Data Preprocessing

To ensure high model performance and consistency, the raw dataset underwent several preprocessing steps:

- **Handling Missing Values:** Null and inconsistent records were identified and removed.
- **Encoding Categorical Features:** Station names, train types, and day of journey were label-encoded.
- **Time Normalization:** Arrival and departure times were converted to numerical values in minutes.
- **Feature Scaling:** The numerical features were normalized using StandardScaler to enhance training efficiency.
- **Data Splitting:** The dataset was split into training and testing sets using an 80:20 ratio.

Model Evaluation Metrics

To evaluate how well the proposed deep learning models predict train delays, we use several well-known statistical performance metrics. These indicators give an overall assessment of both the size of the errors and the consistency of the predictions.

1. Coefficient of Determination (R^2)

The R^2 score, or coefficient of determination, measures how much of the variance in the observed data can be predicted

from the model's outputs. A higher value, nearer to 1, shows stronger explanatory power.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

2. Mean Absolute Error (MAE)

MAE measures the average absolute difference between the actual and predicted values. It is a simple way to see how far predictions differ from true observations, without looking at the direction of the error.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

3. Root Mean Square Error (RMSE)

RMSE emphasizes larger errors due to the squaring operation and provides insight into the magnitude of the error. It is useful for penalizing large deviations more heavily.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

4. Symmetric Mean Absolute Percentage Error (sMAPE)

The sMAPE metric calculates the average percentage error. It is adjusted to be symmetric and scale-independent. This makes it especially useful when handling different scales of prediction values.

$$sMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2}$$

B. Deep Learning Models

This study uses and compares four deep learning models, DNN, CNN, LSTM, and BiLSTM, to predict train delays using historical data.

1) Deep Neural Network (DNN): DNN uses several dense layers with ReLU activation to capture complex feature interactions. Dropout layers help reduce overfitting. While DNN does not explicitly manage sequence data, it acts as a solid baseline for modeling static feature relationships.

2) Convolutional Neural Network (CNN): CNN uses 1D convolutional layers to find spatial patterns in the input features. MaxPooling and flattening layers lower the dimensionality and send the data to dense layers for prediction. CNN works well for identifying localized trends in structured input. **3) Long Short-Term Memory (LSTM):** LSTM is built for sequence data and captures timing relationships in delay progression. It has memory cells and gating mechanisms that keep important time-based information. This makes it a good fit for modeling delay trends across stations and routes.

4) Bidirectional LSTM (BiLSTM): BiLSTM processes input in both forward and backward directions. This improves sequence learning by providing future context. It also boosts delay prediction accuracy by understanding bidirectional temporal relationships in the data.

All models are trained with the Adam optimizer and evaluated with MSE loss. Their performance is measured using RMSE, MAE, and R^2 metrics.

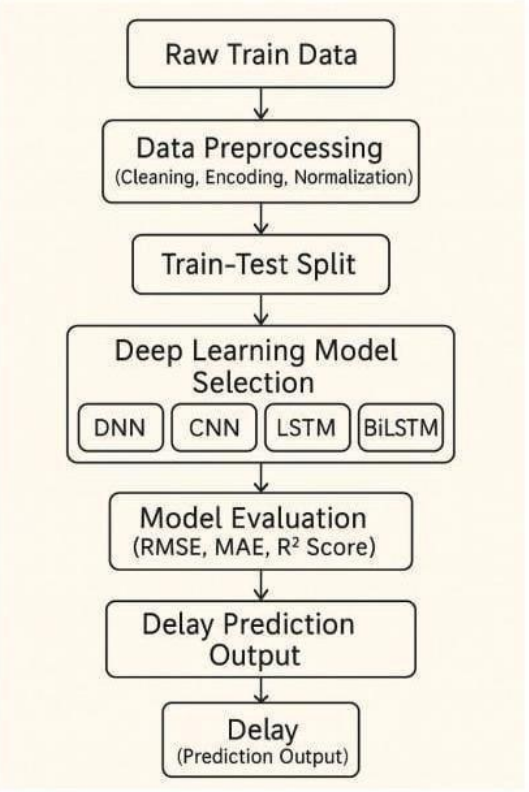

Fig. 1. Proposed Deep Learning Architecture for Train Delay Prediction

C. Proposed System Architecture

The suggested train delay prediction system architecture includes several stages, each of which adds to the correct modeling and forecasting of delays from past railway data. The whole pipeline is made to process raw input data, perform required preprocessing, train several deep learning models, and generate predictions that are assessed by standard performance metrics.

The process starts with the raw railway data ingestion containing features like station names, arrival and departure times, journey dates, total distance, and delay labels. Raw datasets always contain missing values, a non-uniform format for time, and categorical variables. Hence, an extensive preprocessing phase is used to achieve data quality and consistency. This involves transforming time fields to numeric types, encoding categorical features such as stations and train types, scaling numerical features through standardization, and addressing null or invalid entries.

After preprocessing, the dataset is divided into training and test subsets in a general 80:20 proportion to assess model

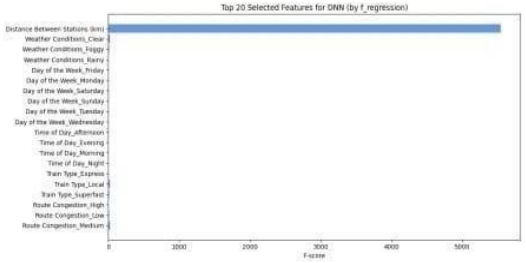

Fig. 2. Proposed Deep Learning Architecture of DNN

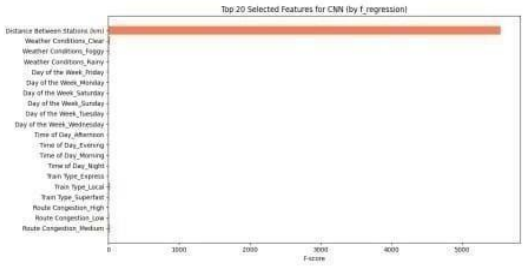

Fig. 3. Proposed Deep Learning Architecture for CNN

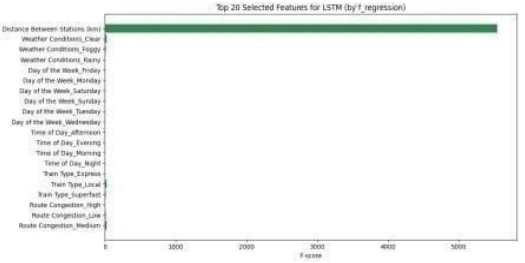

Fig. 4. Proposed Deep Learning Architecture for LSTM

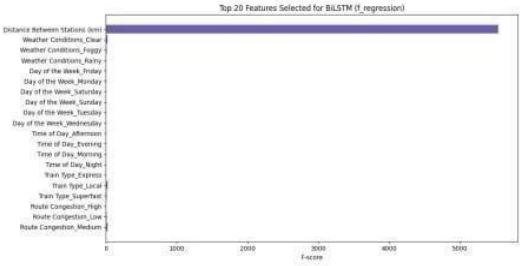

Fig. 5. Proposed Deep Learning Architecture for BILSTM

generalizability. The training set is input into four unique deep learning models — DNN, CNN, LSTM, and BiLSTM — each trained separately using the same input format to enable comparative evaluation. DNN captures rich feature interactions via dense layers; CNN is utilized to learn spatial patterns from the structured feature matrix; LSTM learns dependencies over long-term by exploiting memory cells, and BiLSTM extends temporal modeling by using both past and future contexts.

Every model is optimized with the Adam optimizer and Mean Squared Error (MSE) loss function. Regularization methods like dropout are used to prevent overfitting. Following training, the models are assessed against three critical performance measures: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2 score). These measures tell us about both absolute and relative performance at making predictions.

The design guarantees modularity and reusability, with the capability for future extension or integration with real-time railway control systems. Fig. 2–5 show the top 20 features for DNN, CNN, LSTM, and BiLSTM models in train delay prediction. "Distance Between Stations (km)" consistently ranks the highest. It is followed by weather, goods volume, and day of the week. All models emphasize the strong impact of spatial, weather, and time factors.

D. Experimental Results and Analysis

The dataset used in this project includes historical train schedule data such as arrival time, departure time, day of journey, total travel distance, train number, and station code. The dataset is not from official Indian Railways, but it is inspired by Indian Railways for academic or project use and contains several categorical and numerical features that affect delay occurrences. The target variable indicates the delay in minutes at various stations. To test the effectiveness of the proposed deep learning models, experiments were conducted using the prepared railway delay dataset. The dataset was split into 80

All four deep learning models, DNN, CNN, LSTM, and BiLSTM, were trained on the same input structure to ensure a fair comparison. Model performance was assessed using Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2 score).

TABLE II
PERFORMANCE COMPARISON OF DEEP LEARNING MODELS

Model	R^2 (%)	MAE	RMSE	sMAPE (%)	CPU Time (s)
CNN	94.44	29.68	49.14	70.86	32.9
LSTM	95.87	29.89	42.34	71.92	36.84
DNN	95.48	29.4	44.29	75.92	32.44
BiLSTM	**96.17**	**28.42**	**40.8**	**69.17**	**316.24**

The results demonstrate that BiLSTM achieved the best performance across all evaluation metrics, indicating its superiority in modeling bidirectional temporal dependencies.

Fig. 6 shows how different deep learning models (CNN, LSTM, DNN, BiLSTM) perform using metrics like R², MAE, RMSE, sMAPE, and CPU time. Fig. 7 compares actual and predicted delay values. The BiLSTM and LSTM models are closer to the true delay pattern.

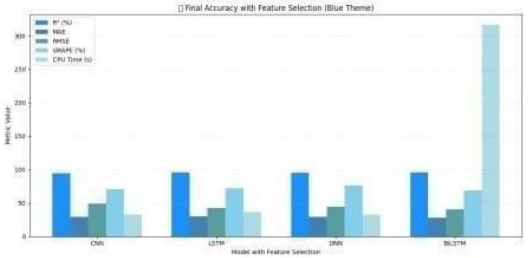

Fig. 6. R^2 , MAE , RMSE , sMAPE , CPU time of all algorithms

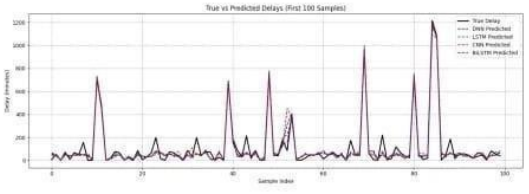

Fig. 7. Results for the different approches of Deep learning

LSTM also did well, highlighting the importance of sequence-based learning. CNN and DNN had moderate performance but struggled to capture long-term patterns in the delay data. These results suggest that models that can understand time context, especially in both directions, are more effective for predicting train delays.

E. Discussion

The results of the performance analysis provide important insights into the relative strength and weakness of deep learning models embraced to make train delay prediction. Out of the four models taken into consideration, Bidirectional LSTM (BiLSTM) outperformed the others in all the evaluation metrics, including RMSE, MAE, and R^2 score. This spectacular performance is due to the reason that BiLSTM has access to both past and future contextual data of a time series, hence a superior viewpoint on temporal dependencies and delay propagation.

The base LSTM model similarly achieved competitive results, testifying to the strength of recurrent neural networks (RNNs) for time series forecasting tasks. Its capacity for long-range dependency learning in sequential data set it high, but it trailed just behind BiLSTM by lacking the reverse-time context.

Compared to that, CNN and DNN models, though making decent predictions, proved relatively less effective. The DNN, though effective at describing nonlinear feature interactions using dense layers, does not have mechanisms for retaining temporal memory. Likewise, the CNN model is naturally more suited to picking up local patterns and spatial hierarchies than global sequential dependencies. Consequently, both models could have failed to take full advantage of the dataset's temporal structure, which is absolutely essential to making correct forecasts of train delays.

The other significant factor to consider is the balance between accuracy and computation. The most accurate BiLSTM model came at the greatest cost of training time and memory consumption because of its dual-pass processing requirement. This aspect might be challenging in the context of deployment in real-time on devices with limited resources but is acceptable in contexts where prediction accuracy must be an utmost concern, like national railway control systems.

Interestingly, the predictive improvement was more noticeable for longer delays, which implies that deep learning models are exceptionally good at learning patterns that correspond to significant operation interruptions. This finding also implies that the models can possibly be used as early warning systems to support proactive actions to counteract cascading delays.

The uniform trend across all the evaluation metrics also validates the pipeline's robustness of preprocessing, which successfully converted raw operational data into structured input with meaning. Feature scaling, time normalization, and categorical encoding were key preprocessing operations, particularly for sensitive recurrent models to input distribution.

In a general sense, these results promote the applicability of deep learning, particularly sequential architectures such as LSTM and BiLSTM, towards delay forecasting in dynamic, high-dimensional, and multivariate settings like railway systems. The versatility of these models also widens the gate towards their integration with real-time sensors, traffic feeds, and weather inputs to transform into an overarching predictive system.

IV. Conclusion

This research shows the potential and practicality of using deep learning techniques to predict train delays based on historical and operational railway data. We systematically implemented and evaluated four distinct deep learning models: Deep Neural Network (DNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Bidirectional LSTM (BiLSTM). Each model contributed uniquely to understanding and forecasting data patterns over time.

Our findings reveal that the BiLSTM model consistently produced better results across multiple evaluation metrics, including RMSE, MAE, and R^2 score. The BiLSTM's strength lies in its ability to learn from both past and future contexts within the sequence data, which greatly improves prediction performance for time-based systems like railway schedules. The CNN and LSTM models also performed well, demonstrating their effectiveness in modeling spatial and temporal correlations.

This work is significant for several reasons. First, it shows that deep learning is not only suitable but also very effective for large-scale transport forecasting problems. Unlike traditional machine learning methods, deep models remove the need for extensive feature engineering and offer automatic hierarchical representation learning. This is especially helpful in dynamic and noisy environments like railway systems.

Second, the proposed system is designed with modularity in mind. Each phase, from data preprocessing to model training and evaluation, can be easily expanded or integrated with more complex real-time systems. This makes the architecture flexible for both batch and streaming data scenarios, paving the way for real-world use in railway traffic control centers, mobile applications, or smart public transport dashboards.

From a societal perspective, accurate delay prediction leads to a better passenger experience, reduced congestion, efficient use of resources, and proactive incident management. It helps passengers make informed decisions and allows railway operators to optimize schedules, re-route traffic, and minimize delays.

Looking ahead, there are various opportunities for enhancement. Future research could include additional features like weather data, public holidays, accident reports, and infrastructure maintenance logs. Moreover, integrating attention mechanisms or Transformer-based models could further improve the modeling of temporal context. Reinforcement learning could be used for dynamic re-scheduling and adaptive prediction based on system feedback.

Furthermore, this framework could be applied to multimodal transportation systems involving buses, flights, and metro services. As smart cities develop, implementing such predictive models on a large scale can lead to more sustainable, efficient, and intelligent transportation systems.

In summary, this study confirms the feasibility and effectiveness of deep learning for predicting railway delays and provides a scalable foundation for further research and development in intelligent transport systems. With ongoing improvements and real-time use, these systems can change how delays are anticipated, managed, and communicated across transportation networks.

REFERENCES

[1] K. Rautela, M. C. Trivedi, and P. P. Roy, "A BiLSTM based deep learning model for predicting train delays," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Glasgow, UK, Jul. 2020, pp. 1-7.

[2] R. K. Eluri, Y. G. Reddy, K. Valicharla, K. D. Prakash, and B. Sudheer, "Improving early detection of diabetic retinopathy: A hybrid deep learning model focused on lesion identification," in *Proc. 2024 First Int. Conf. Innovations in Commun., Electrical and Computer Eng. (ICICEC)*, Oct. 2024, pp. 1-7.

[3] L. Ge, M. Sarhani, S. Voß, and L. Xie, "Review of transit data sources: Potentials, challenges and complementarity," *Sustainability*, vol. 13, no. 20, p. 11450, 2021.

[4] Z. Li, C. Wen, R. Hu, C. Xu, P. Huang, and X. Jiang, "Near-term train delay prediction in the Dutch railways network," *Int. J. Rail Transportation*, vol. 9, no. 6, pp. 520-539, 2020.

[5] N. Marković, S. Milinković, K. S. Tikhonov, and P. Schonfeld, "Examining passenger train arrival delays with support vector regression," *Transportation Research Part C*, vol. 56, pp. 251-262, 2015.

[6] M. Sarhani and S. Voß, "On the effectiveness of SVM-based feature selection for transit delay prediction," 2021.

[7] M. Gargi, R. K. Eluri, O. P. Samantray, and K. Hajarathaiah, "Compact pyramidal dense mixed attention network for diabetic retinopathy severity prediction under deep learning," *Biomed. Signal Process. Control*, vol. 100, p. 106960, 2024.

[8] J. Wessel, J. Allen, and K. Watkins, "Integrating real-time data into GTFS, a public transit data standard," *Transportation Research Record*, vol. 2650, no. 1, pp. 110-118, 2017.

[9] P. Huang, C. Wen, L. Fu, et al., "Modeling train operation as sequences: A study of delay prediction with operation and weather data," *Transportation Research Part E*, vol. 141, p. 102022, 2020.

[10] M. Shoman, T. Laverty, and C. Andris, "Transit delay prediction using entity embeddings on heterogeneous data," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4151-4161, 2020.

[11] R. Al-Naim and Y. Lytkin, "Review and comparison of prediction algorithms for estimated time of arrival using geospatial transportation data," *Procedia Computer Science*, vol. 193, pp. 13-21, 2021.

[12] M. Sarhani and S. Voß, "Evaluation of regression algorithms for rail transit delay prediction using open data," *Public Transport*, vol. 15, no. 3, pp. 635-659, 2023. [Online]. Available: https://doi.org/10.1007/s12469-023-00300-9

[13] C. Wen, P. Huang, and L. Fu, "Predicting transit delays using graph attention networks," *Transportation Research Part C*, vol. 149, p. 103935, 2023.

[14] K. Hajarathaiah, N. B. Naidu, R. K. Eluri, and S. R. Naru, "Automatic recognition of traffic signs based on visual inspection," in *Advances in Electrical and Computer Technologies*, CRC Press, 2025, pp. 545-554.

[15] X. Wang, L. Chen, and Y. Zhao, "Comparative study of machine learning models for metro delay forecasting," *Expert Syst. Appl.*, vol. 216, p. 119419, 2023.

GH raisoni COLLEGE
Engineering and Management Pune

InCoWoCo
14 - 15, November 2025 | GHRCEM, Pune

IEEE Pune Section

IEEE Women in Engineering We
IEEE PUNE SECTION

2025 Second IEEE International Conference for

WOMEN IN COMPUTING (INCOWOCO 2025)

14 - 15, November 2025 | Pune, Maharashtra, India

CERTIFICATE

This certificate is presented to

Paper ID 148

Kotha Lahari
UG Scholar
Computer Science and Engineering,
Narasaraopeta Engineering College
Andhra Pradesh, India

for presenting the research paper entitled
"Leveraging Operational and Environmental Data for Train Delay Prediction via Deep Learning Models"
authored by
Rama Krishna Eluri, Kotha Lahari, Appala Chandana Priya, Yarroju Rekha Sri, Dharmapuri Siri, Kandukuri Swarnalatha

at the 2025 Second IEEE International Conference for Women in Engineering (INCOWOCO 2025) held at G H Raisoni College of Engineering and Management (GHRCEM), Pune, Maharashtra, India during 14 - 15, November 2025. The conference is technically co-sponsored by IEEE Women in Engineering (WiE) of Pune Section and IEEE Pune Section.

Dr. Simran Khiani
General Chair

Prof. Dr. Rajashree Jain
General Chair

Dr. R D Kharadkar
Honorary Chair

Organized by

G H RAISONI COLLEGE OF ENGINEERING AND MANAGEMENT
Domkhel Rd, Wageshwar Nagar, Wagholi, Pune, Maharashtra 412207