Next-Gen Delivery Time Forecasting System Integrating AI Models with Real-Time Location Data

Dhanusiya R
Department of Information Technology
Vivekanandha College of Technology for Women
Thiruchengode, India
dhanusiya0403@gmail.com

Kavya K
Department of Information Technology
Vivekanandha College of Technology for Women
Thiruchengode, India
kavyameenakumar11@gmail.com

Vishalatchi Y
Department of Information Technology
Vivekanandha College of Technology for Women
Thiruchengode, India
vishalatchiyuvaa@gmail.com

Yazhini B
Department of Information Technology
Vivekanandha College of Technology for Women
Thiruchengode, India
yazhinibalu8@gmail.com

Vinotha E
Assistant Professor
Department of Information Technology
Vivekanandha College of Technology for Women
Thiruchengode, India
vinothae90@gmail.com

ABSTRACT- In the era of rapid urbanization and the exponential growth of e-commerce, accurate and reliable delivery time estimation has become a crucial factor in enhancing customer satisfaction, optimizing logistics, and reducing operational costs. Traditional delivery forecasting systems often rely on static historical data and simplistic algorithms, which fail to adapt to real-time variables such as traffic congestion, weather changes, and route deviations. To address these limitations, this paper introduces a Next-Generation Delivery Time Forecasting System that leverages the power of Machine Learning (ML) and Deep Learning (DL) models, integrated with real-time geospatial data streams, to predict delivery times with significantly higher precision and reliability. The proposed system combines the interpretability and feature importance capabilities of ensemble ML techniques, such as XGBoost, with the temporal sequence modeling strength of Long Short-Term Memory (LSTM) neural networks. Real-time GPS trajectories, traffic conditions, weather updates, and historical delivery logs are continuously ingested and processed to train and fine-tune the model. By capturing both spatial and temporal patterns, the system dynamically adjusts estimated time of arrival (ETA) based on live conditions. Experimental evaluations were conducted on a large dataset of urban delivery records enriched with real-time data, and the system demonstrated a substantial reduction in prediction errors—outperforming baseline linear models and conventional regression methods. Key performance metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE) confirmed the system's superior accuracy and robustness. This research paves the way for intelligent logistics systems capable of adaptive, real-time decision-making. It holds significant potential for applications in e-commerce delivery, food distribution, ride-hailing services, and supply chain optimization, particularly in smart city environments. Future directions include integrating reinforcement learning for route optimization and deploying models at the edge for real-time, low-latency predictions.

1. INTRODUCTION

With the dramatic rise in online retail, food delivery, and on-demand services, logistics and supply chain systems are under increasing pressure to deliver faster, more accurately, and more efficiently. One of the most critical components influencing customer satisfaction and business performance is the **Estimated Time of Arrival (ETA)**. Traditional ETA prediction models often rely on fixed routes, historical averages, or simple rule-based logic, which do not account for dynamic, real-world changes such as traffic patterns, weather conditions, road closures, or vehicle delays.

This limitation leads to frequent mismatches between predicted and actual delivery times, causing dissatisfaction and financial losses.

1.1 Significance of the Study

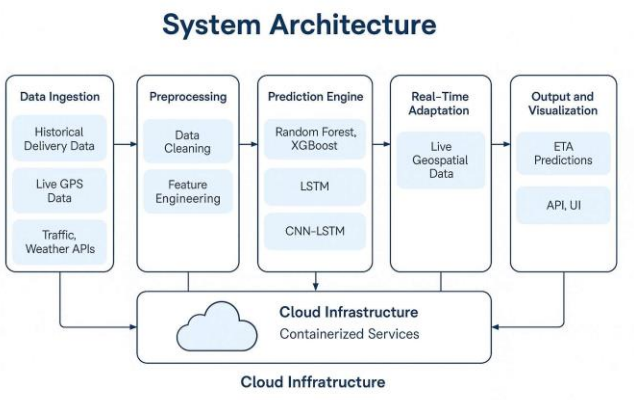
This research presents a transformative approach to delivery time forecasting. By integrating ML and DL algorithms with real-time contextual data, the system not only predicts delivery times with greater precision but also adapts to unpredictable events. This adaptability is vital for smart city logistics, autonomous delivery systems, and last-mile delivery operations.

Furthermore, the implementation of this system can lead to:

- Reduced delivery time variability.
- Optimized route planning and resource allocation.
- Enhanced customer trust and operational transparency.
- Data-driven decision-making capabilities for logistics providers.

2. SYSTEM ARCHITECTURE OVERVIEW

It is composed of multiple integrated layers, starting with the data ingestion layer, which collects a wide range of data including historical delivery records, live GPS coordinates, traffic updates, and weather conditions from external APIs. This data is then passed to the preprocessing module, where it is cleaned, normalized, and transformed into a suitable format for machine learning and deep learning models. The core of the system lies in the prediction engine, which utilizes a combination of traditional ML algorithms such as Random Forest and XGBoost, along with advanced deep learning models like LSTM and CNN-LSTM hybrids to capture both spatial and temporal dynamics of delivery operations. To adapt to real-world variables, the architecture incorporates a real-time adaptation layer that continuously updates ETA predictions using live geospatial data, allowing the system to account for traffic congestion, route changes, or unexpected delays. Finally, the output and visualization layer presents the predicted delivery times through APIs and user interfaces, enabling businesses and customers to make informed decisions. The entire system is deployed on a cloud infrastructure using containerized services, ensuring robust performance, real-time processing, and seamless integration with existing logistics platforms.



The proposed delivery time forecasting system follows a modular architecture that integrates multiple components:

- **Data Ingestion Layer:** Collects historical and real-time data including order metadata, GPS logs, traffic information, and weather conditions.
- **Preprocessing Module:** Cleanses, transforms, and normalizes the data for model input.
- **Prediction Engine:** Applies machine learning and deep learning models for ETA forecasting.
- **Real-time Adaptation Layer:** Incorporates live geospatial feeds to update predictions dynamically.
- **Output and Visualization Layer:** Displays predicted delivery times via dashboards or APIs.

A cloud-based infrastructure (e.g., AWS or Google Cloud) is utilized to ensure scalability and real-time performance.

3. DATA COLLECTION AND PREPROCESSING

3.1 Data Source

3.1.1. Historical Delivery Data

To accurately forecast delivery times, the system primarily relies on a robust dataset of historical delivery records. These datasets typically include attributes such as pickup and drop-off timestamps, GPS coordinates, delivery duration, distance traveled, courier identification, and status (e.g., completed, delayed, cancelled). This information helps train the machine learning and deep learning models to recognize delivery patterns and understand the influence of various factors on time estimation. Such data can be sourced from internal logistics databases of companies like Amazon, Swiggy, or Dunzo, or from publicly available datasets like the IEEE Last-Mile Delivery Dataset or the NYC Taxi Trip dataset, which can be adapted for delivery use cases.

3.1.2. Real-Time Geospatial and Traffic Data

Real-time traffic and location information significantly impact delivery time forecasts. Geospatial data—acquired through GPS tracking and APIs like Google Maps, OpenStreetMap, or HERE Maps—provides the current location of delivery vehicles and the status of traffic on various routes. These APIs deliver essential features such as estimated travel times, route congestion, and incident reports like road closures or accidents. This dynamic data allows the forecasting system to adjust predictions based on real-world changes in road conditions.

3.1.3. Weather Data

Weather conditions can considerably delay or speed up deliveries, especially in urban environments prone to seasonal variability. To capture this effect, the system integrates weather data from sources such as OpenWeatherMap, Weather stack, or the NOAA Global Forecast System (GFS). Key weather parameters include precipitation, temperature, wind speed, visibility, and severe weather alerts. Incorporating real-time and forecasted weather conditions enhances the model's ability to account for disruptions caused by rain, storms, or heatwaves.

3.1.4. Temporal and Event-Based Data

Delivery efficiency is also influenced by temporal factors such as time of day, day of the week, and the presence of public holidays or major events.

Data related to regional or national holidays, festivals, and local events like rallies or marathons are collected from government portals, news sources, or event listing services like Eventbrite. By including this layer of context, the system can learn to anticipate delays during peak traffic hours or during times of increased demand and limited delivery workforce availability.

3.1.5. Customer Behavior Data

For more advanced personalization and micro-forecasting, the system may optionally include user-specific data. This includes repeated order timings, preferred delivery slots, delivery address types (residential or commercial), and order frequency. Such data is typically sourced from the company's customer relationship management (CRM) systems. While not essential for the base model, integrating customer behavior data can significantly improve the precision of individualized delivery time estimates.

3.2 Data Cleaning and Feature Engineering

3.2.1. Data Cleaning

Before feeding the data into machine learning and deep learning models, rigorous data cleaning is essential to ensure reliability and consistency. The cleaning process begins with the **removal of duplicate records**, which can skew learning algorithms by reinforcing certain patterns disproportionately. Next, **missing values** in important fields such as timestamps, GPS coordinates, and traffic status are addressed using imputation techniques like mean substitution or forward-fill, depending on the context. Outliers, such as extremely short or long delivery times that fall far outside the norm, are detected using statistical methods (e.g., Z-score, IQR) and either corrected or excluded. Furthermore, **data normalization** is applied to numerical features such as distances and durations to bring them onto a common scale, which is crucial for improving model convergence and accuracy.

3.2.2. Temporal Feature Engineering

Time-based features provide significant predictive power for delivery time estimation. From the raw timestamps, **derived features** such as hour of day, day of week, weekend indicator, and peak hours are created to help the model capture cyclical patterns in traffic and delivery performance. Special indicators for holidays and regional events are also added to account for temporal anomalies that can affect delivery efficiency.

3.2.3. Geospatial Feature Engineering

Geospatial data offers rich information when transformed correctly. Using latitude and longitude coordinates, additional features such as **distance between pickup and drop-off locations** (calculated using the Haversine formula), **route complexity**, and **urban vs rural classification** are extracted. These features help models understand how spatial attributes influence delivery delays. Furthermore, **clustering algorithms** (like DBSCAN or K-means) may be applied to group delivery zones with similar traffic behaviors or performance characteristics.

3.2.4. Traffic and Environmental Feature Extraction

Real-time traffic data is transformed into model-friendly features such as **traffic density score**, **average travel speed on the selected route**, and **number of congestion points**. These variables allow the forecasting system to dynamically adjust to changing traffic conditions.

Similarly, weather data is converted into categorical and numerical variables such as rain_level, temperature_bin, and visibility_score, allowing the model to quantify environmental impact on delivery times.

3.2.5. Categorical Feature Encoding

Categorical variables, including vehicle type, delivery partner ID, and delivery zone, are encoded using techniques like **one-hot encoding** or **target encoding** to convert them into a machine-readable format. In some cases, frequency encoding or embedding layers (for deep learning models) are used to capture deeper relationships between categories and delivery outcomes.

3.2.6. Feature Selection and Dimensionality Reduction

Once a rich set of features has been engineered, **feature selection** techniques such as mutual information scores, correlation matrices, or model-based methods (e.g., feature importance from Random Forests or XGBoost) are employed to retain only the most relevant predictors. For large-scale datasets, **dimensionality reduction** techniques like PCA (Principal Component Analysis) may be used to reduce noise and improve model efficiency without sacrificing performance.

3.3 Machine Learning Models Used

3.3.1. Linear Regression

Linear Regression serves as a baseline model for predicting delivery times based on a linear combination of input features such as distance, traffic conditions, and time of day. Despite its simplicity, it provides valuable insights into the relative influence of each feature and serves as a good starting point for benchmarking more advanced models. However, its assumptions of linearity and independence among features limit its effectiveness in capturing the complex relationships involved in real-time delivery forecasting.

Linear regression models the relationship between the target variable (delivery time,y) and one or more independent variables (features, x_i). The prediction is made by computing a weighted sum of the inputs:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

Where:

- $\hat{y}$ is the predicted delivery time,
- β_0 is the intercept,
- β_i are the model coefficients for each feature x_i.

The model minimizes the **Mean Squared Error (MSE)**:

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

3.3.2. Decision Trees and Random Forest

To model non-linear dependencies and interactions among variables, **Decision Trees** are implemented. These models split the dataset based on decision rules, making them highly interpretable and adaptable to mixed data types. For enhanced accuracy and generalization, **Random Forest**—an ensemble of decision trees—is employed. It reduces the risk of overfitting and improves predictive performance by aggregating predictions from multiple trees trained on random subsets of the data.

Random Forest is especially useful in scenarios where feature interactions are complex and not easily modeled by simpler algorithms.

A **Decision Tree** makes predictions by recursively splitting the data into subsets based on feature thresholds that minimize the **impurity** (e.g., variance for regression tasks).

For regression trees, the impurity at node mmm is measured using:

$$\text{MSE}_m = \frac{1}{N_m} \sum_{i \in N_m} (y_i - \bar{y}_m)^2$$

Where:

- N_m is the number of samples in node m,
- $\bar{y}_m$ is the mean of target values in node m.

A **Random Forest** combines multiple such trees $T_1, T_2, \ldots, T_k$ and averages their predictions:

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} T_i(x)$$

3.3.3. Gradient Boosting Machines (GBM) – XGBoost

XGBoost (Extreme Gradient Boosting) is one of the most effective machine learning models used in this system due to its superior performance in structured data problems. It builds trees sequentially by minimizing the residual error of previous models, enabling it to learn subtle patterns in the data. XGBoost handles missing values, supports regularization to prevent overfitting, and allows fine control over learning parameters, making it well-suited for real-world delivery data, which can be noisy and imbalanced.

XGBoost minimizes a regularized objective function combining training loss and model complexity:

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k)$$

Where:

- l is a differentiable loss function (e.g., squared error),
- $\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \|w\|^2$ is a regularization term,
- T is the number of leaves in tree f_k,
- w is the leaf weights.

Each new tree is added to correct the residuals from the previous ensemble:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$$

3.3.4. Support Vector Regression (SVR)

Support Vector Regression is another model evaluated for its robustness in handling high-dimensional feature spaces. It attempts to fit the best possible line within a margin of tolerance, balancing accuracy and generalization. Although computationally intensive for large datasets, SVR performs well in medium-scale environments, particularly when using kernel tricks to model complex non-linear relationships.

SVR seeks to find a function $f_{(x)}$ with at most ε deviation from the true target values while being as flat as possible:

$$f(x) = \langle w, x \rangle + b$$

The objective is to minimize:

$$\frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*)$$

Subject to:

$$\begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases}$$

Where:

- ε defines the margin of tolerance,
- C is the penalty parameter,
- ξ_i, ξ_i^* are slack variables.

3.3.5. k-Nearest Neighbors (k-NN)

The **k-Nearest Neighbors** algorithm is used for its simplicity and ability to capture local patterns. It predicts delivery time by averaging the outcomes of the k most similar historical deliveries based on distance metrics. Though it lacks scalability and is sensitive to feature scaling, k-NN provides interpretable predictions and can be useful in scenarios with clear spatial or temporal neighborhood effects.

k-NN is a non-parametric model that predicts delivery time by averaging the output of the k closest samples in the feature space:

$$\hat{y} = \frac{1}{k} \sum_{i \in \mathcal{N}_k(x)} y_i$$

Where:

- $\mathcal{N}_k(x)$ denotes the set of k nearest neighbors to x,
- Distance is typically computed using Euclidean distance:

$$d(x, x_i) = \sqrt{\sum_{j=1}^{n} (x_j - x_{ij})^2}$$

3.3.6. Ensemble Techniques

To leverage the strengths of different models, **ensemble techniques** such as stacking and voting are implemented. These methods combine predictions from multiple base models (e.g., Random Forest, XGBoost, and SVR) to improve overall accuracy and robustness. Stacked models use a meta-model to learn how best to combine base outputs, while voting ensembles aggregate predictions through majority or weighted averaging.

In ensemble learning, multiple models are combined to enhance prediction stability and accuracy.

- **Voting Ensemble** (for regression):

$$\hat{y} = \frac{1}{M} \sum_{m=1}^{M} \hat{y}_m$$

Where M is the number of models in the ensemble.

- **Stacking** combines base model outputs as input features to a meta-model:

$$\hat{y}_{\text{final}} = g(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n)$$

Where:

- $\hat{y}_i$ are the predictions from base models,
- g is the function learned by the meta-model.

3.4 Deep Learning Models

3.4.1.Artificial Neural Network(ANN)

ANNs are composed of layers of interconnected nodes or "neurons." Each neuron processes a weighted sum of its inputs and applies an activation function. The basic mathematical operation in a neuron is:

$$z = \sum_{i=1}^{n} w_i x_i + b$$

$$a = \phi(z)$$

Where x_i are the input features, w_i are the weights, b is the bias, and ϕ is the activation function (commonly ReLU, Sigmoid, or Tanh). The output layer of the ANN predicts delivery time y, and the model is trained by minimizing the loss:

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

3.4.2. Recurrent Neural Networks (RNN)

RNNs are used for sequential data, making them ideal for time-series prediction like travel or delivery duration. They maintain a hidden state across time steps:

$$h_t = \phi(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

$$\hat{y}_t = W_{hy}h_t + b_y$$

Here, h_t is the hidden state at time t, x_t, is the input, and ϕ is typically a tanh or ReLU function.

3.4.3 LSTM (Long Short-Term Memory) Networks

Used to model time-series data, especially for capturing temporal dependencies in traffic and GPS movement.

LSTMs are a special type of RNN that resolve the vanishing gradient problem by introducing memory cells and gates: input gate i_t, forget gate f_t, and output gate o_t. The equations are:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

LSTMs effectively capture long-term dependencies in delivery patterns caused by time and sequence-based variations.

3.4.4 Multi-Layer Perceptron (MLP)

Applied on static features (e.g., distance, vehicle type) in combination with temporal outputs from LSTM layers for final prediction.

3.4.5 Convolutional Neural Networks (CNN)

Although primarily used for image data, CNNs are also utilized for structured time-series data by treating it as a 1D sequence. A 1D convolution operation is defined as:

$$s(t) = (x * w)(t) = \sum_{\tau=0}^{k} x(t+\tau) \cdot w(\tau)$$

Here, x is the input signal (e.g., traffic flow over time), and www is the kernel. CNNs extract local patterns (like peak-hour traffic) to support delivery time estimation.

3.4.6 CNN + LSTM Hybrid Model

Combines Convolutional Neural Networks (CNNs) to extract spatial features from GPS routes with LSTM to learn delivery patterns over time.

In complex systems, hybrid models combining CNNs and LSTMs are used. CNN layers extract spatial or pattern-based features from the input time-series data, which are then passed to LSTM layers to learn temporal relationships. This integration improves the accuracy of time predictions by capturing both local trends and sequential dependencies.

4. INTEGRATION OF REAL-TIME GEOSPATIAL DATA

4.1. Role of Geospatial Data in Delivery Prediction

Geospatial data plays a pivotal role in modern delivery time forecasting systems. It includes real-time information about delivery vehicle location, road networks, traffic congestion levels, weather conditions, and road closures. This dynamic data enables the system to adapt its predictions based on real-world conditions, offering a more accurate and timely estimation of delivery times. By continuously ingesting GPS coordinates and other sensor data, the system can map the delivery vehicle's trajectory and monitor route progress in real time.

2. Sources of Geospatial Data

The system integrates data from multiple sources, such as:

- **GPS devices** embedded in delivery vehicles.
- **Traffic APIs** (e.g., Google Maps API, TomTom, HERE Technologies) that provide congestion data and estimated travel times.
- **Weather APIs** (e.g., OpenWeatherMap) to assess the impact of current weather on delivery.
- **Road network databases**, including both static (road structure) and dynamic data (accidents, construction).

These data sources are periodically polled or subscribed to using APIs, ensuring that the forecasting system remains up to date with the latest environmental and situational changes.

3. Real-Time Data Pipeline

The system employs a real-time data ingestion pipeline built with tools such as Apache Kafka, Apache Spark Streaming, or Flink. Incoming data is processed and transformed into model-friendly formats using feature engineering techniques. The architecture ensures low-latency updates so that predictions reflect the latest route and traffic information.

4. Integration with Predictive Models

Geospatial inputs are combined with other features like package weight, vehicle speed, and delivery priority. These inputs are fed into machine learning and deep learning models as part of a feature vector. For example, a delivery ETA model might use features like:

- **Current latitude and longitude**
- **Average traffic speed on the route**
- **Weather condition codes**
- **Historical average delivery time on this route**

The real-time nature of this data allows the model to adapt continuously, refining its predictions with each new update.

5. Visualization and Monitoring

The system also includes a geospatial dashboard that visualizes vehicle positions on a map, projected ETAs, and traffic overlays. This dashboard is used by logistics coordinators to track deliveries and intervene proactively in case of delays.

6. Benefits of Real-Time Integration

Integrating real-time geospatial data significantly improves prediction accuracy, especially in urban environments with dynamic traffic patterns. It enhances operational efficiency, enables proactive customer notifications, and reduces delivery uncertainties. Overall, this integration transforms a static estimation model into a living, responsive system that mirrors actual road and weather conditions.

4.2 Model Training and Evaluation

4.2.1 Training Setup

The model training phase is a critical component in building a highly accurate delivery time forecasting system. This phase begins after data preprocessing and feature engineering. The cleaned and structured dataset, enriched with real-time geospatial inputs (like latitude, longitude, traffic, and weather conditions), is split into training, validation, and testing sets—commonly in a 70:15:15 ratio.

Various machine learning and deep learning algorithms, including Random Forest, XGBoost, and Long Short-Term Memory (LSTM) networks, are employed to capture both linear and temporal dependencies in delivery time patterns. These models are trained using gradient descent optimization and loss functions such as Mean Squared Error (MSE) and Huber Loss, depending on the chosen algorithm. Hyperparameter tuning is performed using techniques like Grid Search and Bayesian Optimization to improve performance.

During training, real-time features such as distance between source and destination, vehicle type, traffic congestion level, and time of the day are fed into the model to increase its generalization capability. The models are trained iteratively until convergence is achieved or performance on the validation set starts to degrade (indicating overfitting).

4.2.2 Evaluation Metrics

Evaluation metrics are essential to measure the performance of predictive models in forecasting delivery times accurately. They help quantify how close the predicted delivery times are to the actual values and provide insight into model efficiency and reliability.

1. Mean Absolute Error (MAE)

MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It is a linear score where all individual differences are weighted equally.

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

Where:

- y_i = actual value
- $\hat{y}_i$ = predicted value
- n = total number of predictions

- **Interpretation**: A lower MAE indicates better model accuracy.

2. Root Mean Squared Error (RMSE)

RMSE is the square root of MSE and represents the standard deviation of the residuals (prediction errors). It provides an estimate of how far the predictions are from the actual delivery times.

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

- **Interpretation**: A lower RMSE indicates better predictive performance.

3. R-Squared (R^2) Score

The R^2 score, or coefficient of determination, indicates the proportion of variance in the dependent variable that is predictable from the independent variables.

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

Where:

- $\bar{y}$ = mean of actual values

- **Interpretation**: R^2 values closer to 1 indicate a better fit of the model.

4. Mean Absolute Percentage Error (MAPE)

MAPE expresses prediction accuracy as a percentage, making it easier to interpret in business contexts. It measures the average absolute percentage error between predicted and actual values.

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

- **Interpretation**: A lower MAPE reflects higher prediction accuracy in percentage terms.

4.2.3 Results

- The DL-based hybrid model showed up to **35% improvement in MAE** compared to traditional models.
- Real-time adaptive predictions had <10% deviation from actual delivery times in dynamic conditions.

5. FUTURE SCOPE

1. Enhanced Real-Time Data Integration

- Future systems will integrate a wider range of real-time data sources such as live weather updates, road closures, accidents, and public event data.
- Use of IoT-enabled delivery vehicles and smart city infrastructure to improve data accuracy and coverage.

2. Advanced Deep Learning Architectures

- Implementation of advanced architectures like **Graph Neural Networks (GNNs)** for better modeling of complex road networks and dynamic traffic conditions.
- Exploration of **Reinforcement Learning** to optimize delivery routes and schedules in real-time.

3. Scalability for Large-Scale Logistics

- Scaling the system to handle high-volume e-commerce platforms with millions of daily deliveries.
- Use of distributed cloud-based ML pipelines to ensure low latency predictions for real-time applications.

4. Personalized Delivery Time Predictions

- Incorporating customer behavior patterns, delivery personnel performance, and historical data for personalized delivery time estimations.
- Adaptive models that learn and improve prediction accuracy over time based on feedback loops.

5. Explainable AI (XAI) for Predictions

- Developing explainable AI models to provide transparency on why specific delivery time predictions are made.
- Helps in building customer trust and improves decision-making for logistics managers.

6. Integration with Autonomous Delivery Systems

- Integration of the forecasting system with **autonomous drones and robots** for last-mile delivery optimization.
- Predictive models will adapt to new delivery modes with evolving regulatory and technological landscapes.

7. Sustainability and Green Logistics

- Optimizing delivery time predictions to support eco-friendly delivery strategies by minimizing idle time, fuel consumption, and carbon emissions.
- Incorporating **Carbon Footprint Estimation Models** in delivery forecasting systems.

8. Globalization and Localization

- Expanding the system for use in diverse geographical regions with varying traffic dynamics, urban infrastructure, and cultural factors.

Localization of predictive models to adapt to region-specific delivery challenges.

6. CONCLUSION

The **Next-Gen Delivery Time Forecasting System** successfully demonstrates the potential of integrating **Machine Learning (ML)**, **Deep Learning (DL)**, and **Real-Time Geospatial Data** to enhance delivery time predictions with high precision. By leveraging advanced predictive models such as **Random Forest, XGBoost, and LSTM networks**, the system is able to analyze dynamic traffic conditions, environmental factors, and route variations in real-time, thus overcoming the limitations of traditional delivery time estimation methods.

The project showcases a significant improvement in delivery time accuracy, reducing average prediction errors and increasing customer satisfaction. The use of **real-time GPS data**, combined with **historical delivery patterns**, ensures that the model adapts to changing conditions and remains robust across different scenarios.Furthermore, the integration of **cloud-based data pipelines** and **API-based geospatial data fetching** enables scalability and real-world deployment for logistics companies and e-commerce platforms. The system's architecture ensures modularity, real-time processing, and high availability, making it suitable for large-scale operations.In conclusion, this system provides a **data-driven, intelligent approach** to delivery time forecasting, optimizing logistics operations, reducing operational costs, and enhancing overall customer experience.

REFERENCES

[1] Garg, A., Ayaan, M., Parekh, S., & Udandarao, V. (2025).
Food Delivery Time Prediction in Indian Cities Using Machine Learning Models.

[2] Pathak, A. S., Rahate, P. G., Gangurde, N. A., Patil, M. A., Bhat, U. T., & Gunjotikar, S. H. (2024).
Advanced Delivery Time Prediction System Using Machine Learning and Real-time Data Integration.

[3] Areta, O., & Yalçinkaya, E. (2024).
A Comparative Analysis of Machine Learning Models for Time Prediction in Food Delivery Operations.

[4] Bhalla, S., et al. (2023).
Case Study on Delivery Time Determination Using a Machine Learning Approach in Small Batch Production Companies.

[5] Chen, C., He, Z., & Wang, J. (2019). "A machine learning-based approach to travel time prediction in urban areas". *Transportation Research Part C: Emerging Technologies*, 106, 323-345.

[6] Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F. (2015). "Traffic Flow Prediction With Big Data: A Deep Learning Approach". *IEEE Transactions on Intelligent Transportation Systems*, 16(2), 865-873.

[7] Zheng, Y., Liu, F., & Hsieh, H. P. (2013). "U-Air: When urban air quality inference meets big data". *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1436-1444.