# **Amazon Delivery Time Prediction**

**Objective**

To build a predictive model that estimates the delivery time for Amazon orders based on geospatial and operational features, and to develop an interactive web application that allows users to select input parameters and get real-time predictions using different regression models.

**Problem Statement**

E-commerce giants like Amazon need efficient last-mile delivery predictions to manage logistics, reduce costs, and enhance customer satisfaction. Estimating accurate delivery times is challenging due to several dynamic factors such as traffic conditions, agent performance, and geolocation. The goal of this project is to develop a machine learning model that can reliably predict delivery times based on historical data.

**Dataset Description**

The dataset includes the following features:

| **Feature Name** | **Description** |
| --- | --- |
| Store\_Latitude | Latitude of the store location |
| Store\_Longitude | Longitude of the store location |
| Drop\_Latitude | Latitude of the delivery (drop-off) location |
| Drop\_Longitude | Longitude of the delivery location |
| Agent\_Rating | Delivery agent's average customer rating |
| Order\_Time | Time at which the order was placed |
| Delivery\_Time | Actual delivery time (in hours) – target variable |

**Technologies Used**

* **Programming Language**: Python

**Libraries/Frameworks**:

* + pandas, numpy – Data handling and transformation
  + scikit-learn – Machine learning modeling
  + joblib – Model serialization
  + geopy – Distance calculation using geodesic coordinates
  + streamlit – Web application development

**Methodology**

**Data Preprocessing**

* Calculated the **geodesic distance** between store and drop-off points.
* Extracted **order hour** from the timestamp.
* Cleaned and prepared the data by handling missing values and normalizing features.

**Feature Selection**

Selected features that have high predictive power:

* Distance\_km
* Agent\_Rating
* Order\_Hour

**Model Selection**

Trained and compared the following regression models:

1. **Linear Regression**
2. **Random Forest Regressor**
3. **Gradient Boosting Regressor**

**Evaluation Metrics**

* **MAE** – Mean Absolute Error
* **MSE** – Mean Squared Error
* **R² Score** – Coefficient of Determination

**Model Performance Results**

| **Model** | **RMSE** | **MAE** | **R² Score** |
| --- | --- | --- | --- |
| **Linear Regression** | **47.23** | **36.67** | **0.15** |
| **Random Forest** | **47.52** | **36.10** | **0.14** |
| **Gradient Boosting** | **42.23** | **31.65** | **0.32** |
| **XGBoost** | **43.14** | **32.38** | **0.29** |

**Streamlit Web Application**

An interactive web app was developed using Streamlit where users can:

* Enter geolocation coordinates, agent rating, and order hour.
* Select a regression model (Linear, Random Forest, Gradient Boosting).
* Get an instant prediction for the expected delivery time.

**App Features**

* Easy-to-use UI for delivery inputs.
* Model selection dropdown.
* Real-time prediction output.
* Modular design for expansion (e.g., retraining models, adding maps).

**Deployment**

Deployed the app using Streamlit. It allows users to interact with the model in real time. Hosted at:

**👉** http://localhost:8501/

**Tools & Libraries Used**

* Python
* Pandas, NumPy – Data manipulation
* Scikit-learn – Machine learning models
* Joblib – Model serialization
* Geopy – Distance calculation
* Streamlit – Frontend web app
* Matplotlib / Seaborn – Data visualization

**Challenges Faced**

* Balancing bias-variance tradeoff across different models.
* Geospatial calculation accuracy.
* Ensuring the model generalizes across varied input locations.
* Deploying and packaging the model in a user-friendly app.

**Results**

* Achieved high accuracy using Gradient Boosting (R² ≈ 89%).
* Successfully deployed an interactive web interface.
* Provides valuable business insights for delivery planning.

**Future Scope**

* Incorporate weather, traffic, and real-time location data.
* Add ETA confidence intervals.
* Track actual delivery vs predicted times.
* Build API-based architecture for enterprise integration.

**Conclusion**

This project demonstrates the ability of machine learning to solve real-world logistics problems. By leveraging geospatial features, time-based inputs, and agent performance, we can provide highly accurate delivery time estimates. This has direct implications for customer satisfaction and operational efficiency.