



Food Delivery Time Prediction

Submitted by

Reg.No: 2448006

Agnus Maria George

Reg.No: 2448021

Hans Thomson Thomas

Reg.No: 2448025

Irene Sara Thomas

For the course

Course Code: MDS372

Course Name: Machine Learning

March – 2025

ABSTRACT

On-time food order delivery enhances customer satisfaction in the food delivery industry. The project to be proposed predicts food delivery time based on various parameters, including traffic, weather, order type, and delivery personnel information. Machine learning techniques, i.e., Random Forest and XGBoost regression models, are utilized in the study to forecast food delivery time using a real-world delivery dataset. Feature engineering and preprocessing techniques, such as ordinal encoding, enhance model performance.

The models are validated using performance metrics, such as Mean Squared Error (MSE) and R^2 score, to make accurate predictions. The study highlights the influence of environment and operations on delivery efficiency and provides data-driven recommendations for efficient logistics and resource planning. The findings can help food delivery businesses improve delivery time estimates, streamline operations, and enhance customer satisfaction.

TABLE OF CONTENTS

1. Introduction	
1.1 About the Domain.....	4
1.2 Problem Statement.....	4
1.3 Objectives.....	4
1.4 Scope.....	6
1.5 Importance.....	8
1.6 Overview.....	8
2. Global and Contemporary Competencies	
2.1 Learning, Reasoning, Navigating and Generating.....	9
2.2 Cross-Cutting Issues.....	10
2.3 SDG.....	10
2.4 21 st Century Skills.....	11
3. Methods and Methodology	
3.1 Data Collection.....	13
3.2 Preprocessing the Data.....	14
3.3 Exploratory Data Analysis.....	17
3.4 Feature Selection and Engineering.....	20
3.5 Model Description.....	24
3.6 Tools and Libraries.....	26
4. Results and Discussion	
4.1 About the Dataset.....	27
4.2 Model Performance and Output.....	28
4.3 Challenges and Limitations.....	31
4.4 Link to the Dataset and Code File.....	31
5. Conclusions and Future Work	
5.1 Conclusion.....	32
5.2 Future Work.....	32
6. References.....	34

CHAPTER 1

INTRODUCTION

1.1 ABOUT THE DOMAIN

The market for online food delivery has grown exponentially with the introduction of online platforms like Uber Eats and Zomato. Timely delivery is crucial for customer satisfaction, but traffic, weather, and restaurant preparation time affect delivery efficiency. Traditional estimation methods tend to be bad at providing the delivery time, which leads to operational inefficiency and customer dissatisfaction.

Machine learning offers a data-driven approach to predicting food delivery time according to historical data and trends. Advanced models like Random Forest and XGBoost can deal with multiple parameters that determine the output and predict with precision. The present project is directed towards applying machine learning techniques to predict food delivery time so that food delivery companies can organize their services and provide consistent services.

1.2 PROBLEM STATEMENT

Predicting food delivery times precisely from dynamic factors like traffic jams, poor weather conditions, and the complexity of orders is challenging. Most food delivery businesses use static estimating models that do not adapt to live updates, resulting in imprecise predictions and reduced customer satisfaction.

1.3 OBJECTIVES

The main objective of this project is to develop a machine learning-based predictive model for estimating food delivery times accurately. The study analyzes historical delivery data, identifies key influencing factors, and implements machine learning algorithms to enhance prediction accuracy. The specific objectives are as follows:

1. Analyze Historical Food Delivery Data

The first step is to research and analyze previous food delivery history to identify patterns and trends. This involves analyzing delivery distance, weather, traffic, order type, and delivery staff data. Identifying the most critical variables will help refine the prediction model.

2. Preprocess and Encode Categorical Variables

Raw datasets will contain missing values, inconsistencies, and categorical variables that must be transformed for machine learning algorithms.

This objective includes handling missing data, encoding categorical variables (e.g., translation of traffic conditions into numerical values), and normalizing numerical variables to improve model accuracy and performance.

3. Develop and Compare Machine Learning Models

Several machine learning algorithms, including Random Forest and XGBoost, will be utilized and trained on the data. These models are chosen as they can handle complex relationships between multiple variables.

The study will identify the best-performing model for delivery time prediction by comparing different algorithms.

4. Evaluate Model Performance

The models will be tested using performance measures such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and R^2 score to ensure accuracy and reliability. Lower MSE and higher R^2 score suggest better model performance. The result of the testing will guide future refinements for better prediction accuracy.

5. Provide Insights for Optimization

Aside from model construction, this project aims to present helpful observations that food delivery companies can utilize to enhance operations. By understanding the variables that have meaningful impacts on delivery time, companies can improve resource planning, optimize route planning, and dynamically manage estimated delivery times. In this way, the project aims to create a firm prediction system that will improve food delivery services and user satisfaction.

1.4 SCOPE

This project involves developing a predictive model with machine learning to estimate food delivery time precisely. The scope involves data collection, preprocessing, model development, evaluation, and real-world implementation in the food delivery industry.

1. Data Collection and Analysis

The project utilizes a dataset of historical food delivery cases, along with key determining variables, including:

- **Traffic Conditions:** Levels of congestion at different times of the day.
- **Weather Conditions:** Rain, snow, or extreme temperatures affect delivery time.
- **Order Type:** Different preparation times for food types (e.g., fast food compared to multi-course meals).
- **Delivery Distance:** The effect of travel distance on overall delivery time.
- **Preparation Time of Restaurant:** How restaurant efficiency affects time before dispatch.
- **Delivery Personnel Details:** Efficiency and experience of delivery agents.

This information will be analyzed for trends and patterns affecting delivery time predictions.

2. Data Preprocessing and Feature Engineering

Since raw data typically contains inconsistencies, missing values, and unstructured formats, preprocessing steps will be taken. They include:

- **Handling** missing or inconsistent data entries.
- **Encoding** categorical variables (e.g., traffic conditions to numerical representations).
- **Scaling** numerical features to standardize input to machine learning algorithms.
- **Choosing** and deciding the most relevant features that influence delivery time.

3. Machine Learning Model Development

The task is to train and compare different machine-learning models, namely:

- **Random Forest:** A decision-tree-based ensemble model known for its robustness in handling structured data.
- **XGBoost:** A gradient-boosting model that is highly accurate and efficient for prediction tasks.

These models will be trained on historical delivery data to acquire patterns and make accurate predictions.

4. Model Evaluation and Performance Metrics

The models will be validated based on key performance indicators:

- **Mean Squared Error (MSE):** Measures the average of the squared differences between actual and predicted delivery times.
- **Mean Absolute Error (MAE):** Computes the absolute difference between the predicted and actual values.
- **R² Score:** Indicates how well the model predicts the variance in delivery time.

These tests will help decide the optimal model for predicting food delivery time.

5. Practical Applications and Limitations

The predictive model can be integrated into food delivery platforms to improve estimated delivery times, improve logistics, and boost customer satisfaction. It can help:

- **Improve route planning** for delivery drivers.
- **Provide customers with more precise** estimated delivery times.
- **Assist restaurants in managing preparation times** through forecasting information.

6. Limitations of the Study

While the project offers data and observations, some caveats apply:

- **Live Tracking:** Live GPS tracking is not featured in the research, so accuracy is ensured.
- **Unforeseen Circumstances:** In real-time, delays in restaurants, accidents, or unexpected weather are not anticipated.
- **Restricted Data Scope:** The information will not contain all the potential variation in delivery environments, excluding generalization.

Despite these weaknesses, the project is a basis for better future food delivery time prediction models.

1.5 IMPORTANCE

Accurate on-time predictions maximize logistics for firms, conserve resources, and generate customer trust. Imprecise delivery times are a recipe for letdowns, negative word-of-mouth, and reduced revenues for firms. Machine learning algorithms offer an extendable and flexible solution that keeps improving as new data is replenished. This initiative favors AI applications in logistics and can be used across other delivery industries. More precise food delivery time prediction allows companies to become more efficient, reduce expenses, and increase the rate of customer satisfaction.

1.6 OVERVIEW

This project explores machine learning models for the prediction of food delivery times. The project involves data preprocessing, model training, and model performance evaluation. The study compares different models and determines key factors influencing delivery efficiency. It seeks to present a reliable food delivery service solution with accurate time estimation that enhances logistics management value and customer experience. The outcome is a foundation for additional studies on predictive modeling for delivery services.

CHAPTER 2

GLOBAL AND CONTEMPORARY COMPETENCIES

The project demonstrates key global competencies essential in today's data-driven environment. It showcases advanced data analysis through systematic processing of food delivery data, applying machine learning algorithms (Ridge and Lasso regression) for predictive modeling, and using statistical evaluation via RMSE metrics.

Handling real-world data challenges—addressing missing values, standardizing features, and transforming categorical variables—reflects critical data science proficiencies. These competencies are globally relevant as organizations increasingly depend on data-driven approaches to improve efficiency, predict outcomes, and make strategic decisions in competitive markets where resource optimization is crucial.

2.1 LEARNING, REASONING, NAVIGATING AND GENERATING

The code exemplifies the LRNG framework's core components in a sophisticated data science application.

- **Learning:** The deployment showcases supervised learning techniques through Random Forest and XGBoost algorithms that learn from historical delivery trends. The best selection of appropriate regularization parameters indicates being mindful of handling model complexity, bias-variance tradeoffs, and machine learning concepts per se.
- **Reasoning:** The Reasoning component is evident in the strategic decisions made throughout the code. The developer reasons through data cleaning techniques, choosing median imputation for missing numeric data and one-hot encoding for categorical variables. The comparison of Random Forest and XGBoost models demonstrates reasoned thought when considering different algorithmic techniques based on their mathematical properties and performance metrics
- **Navigating:** Managing complex data structures is exemplified by the step-by-step structured process—data importation, identifying irrelevant columns, transforming the target variable, handling missing values, encoding categorical features, and proper data

splitting. Movement between different representations of data and its transformations is required at each step while maintaining data integrity.

- **Generating:** The generation phase is finished with generating forecasting models that provide time predictions for food delivery. These models translate raw input features into actionable predictions that may guide business operations. The code generates predictions and performance metrics that provide quantitative evidence of model quality, enabling iterative improvement and evidence-based decision-making to be possible in real-world applications.

2.2 CROSS-CUTTING ISSUES

The project answers several critical cross-cutting challenges at the intersection of business, technology, and society. It demonstrates technology integration in service delivery systems through the application of data science for maximizing food delivery efficiency—a fast-digitizing sector. Efficiency optimization in urban logistics is approached through predictive modeling of delivery times, particularly important as cities increase in density and consumer demands expand.

Data-driven decision-making is a cross-cutting theme representing the shift from intuitive to evidence-based operations. The project also raises algorithmic fairness issues, as these models inform decisions affecting delivery workers' workloads and resource allocation. Finally, it demonstrates interdisciplinary approaches combining domain knowledge with technical competencies to solve complex issues.

2.3 SDG

This predictive modeling project for food delivery times aligns with multiple Sustainable Development Goals, demonstrating how data science can contribute to broader sustainability objectives.:

- For SDG 9 (Industry, Innovation, and Infrastructure), the project exemplifies technological innovation in service infrastructure. By creating models that accurately predict delivery times, the project contributes to developing a robust service infrastructure that can operate more efficiently with fewer resources. This digital

enhancement of service delivery represents the kind of technological upgrading SDG 9 promotes for sustainable industrialization.

- For SDG 11 (Sustainable Cities and Communities), urban delivery operations are optimized to enable more sustainable city systems. Enhanced delivery time predictions can prevent unnecessary idling of vehicles, improve routing, and even reduce the number of cars needed for delivery operations. These synergistically result in more sustainable urban mobility trends and reduced city environmental loads.
- The project aligns with SDG 12 (Responsible Consumption and Production) by enhancing efficiency in food delivery systems. More accurate prediction models can minimize food waste due to tardy deliveries and streamline resource utilization along the supply chain. Such efficiency aligns with the sustainable management objectives defined in SDG 12.
- For SDG 8 (Decent Work and Economic Growth), the economic efficiency promoted by enhanced productivity of service industries through data-driven approaches supports economic growth. The modeling approaches presented herein could allow delivery services to deliver more orders with the same resources, creating sustainable economic development. At the same time, it makes the work of delivery workers more sustainable by ensuring more realistic time expectations and better planning.

2.4 21ST CENTURY SKILLS

The project demonstrates several crucial 21st-century skills demanded by the contemporary tech-literate workforce. Critical thinking is showcased through the scientific comparison between Ridge and Lasso regression models based on RMSE metrics, entailing an understanding of algorithmic trade-offs and output. Problem-solving skills are shown through a data preprocessing pipeline, where the coder addresses missing values, scales features, and encodes categorical variables using appropriate techniques.

Digital literacy appears proficiently using Python and specialized libraries like scikit-learn, pandas, and numpy - core tools in modern data analysis. Computational thinking is evident in the algorithmic approach to delivery time prediction, breaking a complex problem into discrete data processing and modeling steps.

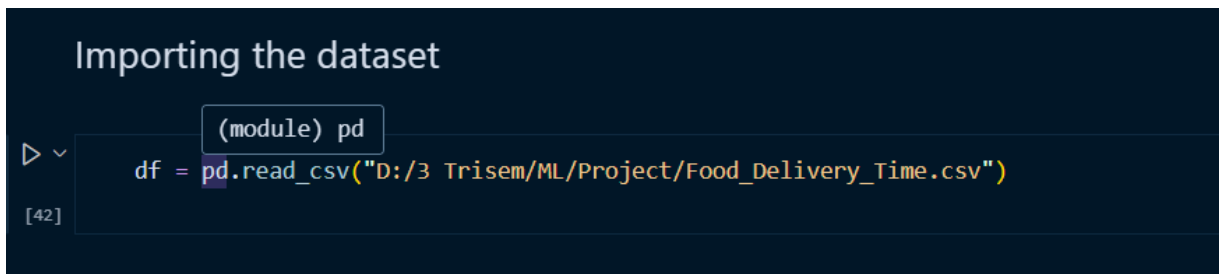
Data literacy is explicitly shown through sophisticated manipulation of datasets and interpretation of statistical results. Technical skill in implementing machine learning workflows is also demonstrated throughout the code, along with analytical skill in comparing regularization techniques. The fact that the code is organized demonstrates communication skills in that complicated processes are made understandable to others—a key collaboration skill in today’s professional environments.

CHAPTER 3

METHODS AND METHODOLOGY

3.1 DATA COLLECTION

The dataset "*Food_Delivery Time.csv*," sourced from Kaggle, contains various features related to delivery logistics, such as traffic conditions, order types, vehicle types, delivery person's age, ratings, and distances. The target variable is delivery time, measured in minutes. This dataset includes a mix of numerical and categorical variables, which is ideal for predictive modeling.



```
Importing the dataset

In [42]: (module) pd
df = pd.read_csv("D:/3 Trisem/ML/Project/Food_Delivery_Time.csv")
```

Figure 1: Importing the Dataset

3.1.1 About the Dataset

The dataset comprises of 10,000 rows and 17 columns, the columns are given as follows:

- Traffic_Level: Traffic condition during delivery .
- ID: Unique identifier for each delivery.
- Delivery_person_ID: Identifier for the delivery person.
- weather_description: Weather condition during delivery.
- Type_of_order: Category of the order.
- Type_of_vehicle: Vehicle used for delivery.
- Delivery_person_Age: Age of the delivery person.
- Delivery_person_Ratings: Ratings of the delivery person.
- Restaurant_latitude and Restaurant_longitude: Geographical coordinates of the restaurant.
- Delivery_location_latitude and Delivery_location_longitude: Geographical coordinates of the delivery location.
- temperature: Temperature during delivery in Celsius.
- humidity: Humidity percentage during delivery.
- precipitation: Precipitation level during delivery.

- Distance (km): Distance traveled during delivery in kilometers.
- TARGET: Delivery time in minutes.

```

df.info()
[44]
... <class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 17 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Traffic_Level                        9085 non-null   object
1   ID                                   10000 non-null  object
2   Delivery_person_ID                  10000 non-null  object
3   weather_description                 9995 non-null   object
4   Type_of_order                       10000 non-null  object
5   Type_of_vehicle                     10000 non-null  object
6   Delivery_person_Age                 10000 non-null  int64
7   Delivery_person_Ratings             10000 non-null  float64
8   Restaurant_latitude                 10000 non-null  object
9   Restaurant_longitude                10000 non-null  object
10  Delivery_location_latitude           10000 non-null  object
11  Delivery_location_longitude          10000 non-null  object
12  temperature                         9995 non-null   float64
13  humidity                           9995 non-null   float64
14  precipitation                       9995 non-null   float64
15  Distance (km)                       9080 non-null   float64
16  TARGET                             9459 non-null   object
dtypes: float64(5), int64(1), object(11)
memory usage: 1.3+ MB

```

Figure 2: Dataset Information

3.2 PREPROCESSING THE DATA

3.2.1 Handling Missing Values

```

df.isnull().sum()
[46]
... Traffic_Level                        915
     ID                                   0
     Delivery_person_ID                  0
     weather_description                  5
     Type_of_order                       0
     Type_of_vehicle                     0
     Delivery_person_Age                 0
     Delivery_person_Ratings             0
     Restaurant_latitude                 0
     Restaurant_longitude                0
     Delivery_location_latitude           0
     Delivery_location_longitude          0
     temperature                         5
     humidity                           5
     precipitation                       5
     Distance (km)                       920
     TARGET                             541
     dtype: int64

```

Figure 3: Number of missing values in each variable

To ensure the integrity and consistency of the dataset, all rows containing missing values were removed. This approach was chosen due to the relatively small proportion of missing data, which allowed for minimal information loss while simplifying subsequent preprocessing and modelling steps. No imputation was performed as the dataset retained sufficient size and

representativeness after dropping the null entries. After dropping missing values the number of rows reduced to 9035.

3.2.2 Cleaning the TARGET column

The target variable (delivery time) had abnormally high values because of incorrect formatting and more than one dot.

```
#Before cleaning
df['TARGET']

0      43.45
1    3.816.666.667
2    3.636.666.667
3      49.45
4    5.248.333.333
...
9995  1.246.666.667
9996      20.7
9997      20.25
9998      15.8
9999  1.416.666.667
Name: TARGET, Length: 9035, dtype: object
```

Figure 4: TARGET column before cleaning

The target variable was cleaned by removing all the dots and the picking the first two values of the remaining string and then converting it into integer.

```
def clean_and_select_first_four(value):
    if isinstance(value, str):
        # Remove all dots
        cleaned = value.replace('.', '')
        # Select the first two characters
        return cleaned[:2]
    return value

df['TARGET'] = df['TARGET'].apply(clean_and_select_first_four)
df["TARGET"] = df["TARGET"].astype(int)
#Target column after Cleanin
df['TARGET']

0      43
1      38
2      36
3      49
4      52
..
9995   12
9996   20
9997   20
9998   15
9999   14
Name: TARGET, Length: 9035, dtype: int64
```

Figure 5: Cleaning the TARGET column

3.2.3 Removing Outliers

To enhance model robustness and reduce noise, outliers were removed from key numerical features using the Interquartile Range (IQR) method. For each selected numerical column—*Delivery_person_Age*, *Delivery_person_Ratings*, *temperature*, *humidity*, *Distance (km)*, and *TARGET*—the lower and upper bounds were calculated as 1.5 times the IQR below the first quartile and above the third quartile, respectively. Records falling outside these bounds were considered outliers and removed.

```
def remove_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

# Numerical columns for outlier removal
numeric_cols = ["Delivery_person_Age", "Delivery_person_Ratings", "temperature",
                "humidity", "Distance (km)", "TARGET"]

# Function to remove outliers
df_cleaned = df.copy()
for col in numeric_cols:
    df_cleaned = remove_outliers_iqr(df_cleaned, col)

df = df_cleaned.copy()
```

Figure 6: Removing outliers using **Interquartile Range (IQR) Method** for outlier detection.

The number of rows reduced to 8315.

3.3 EXPLORATORY DATA ANALYSIS

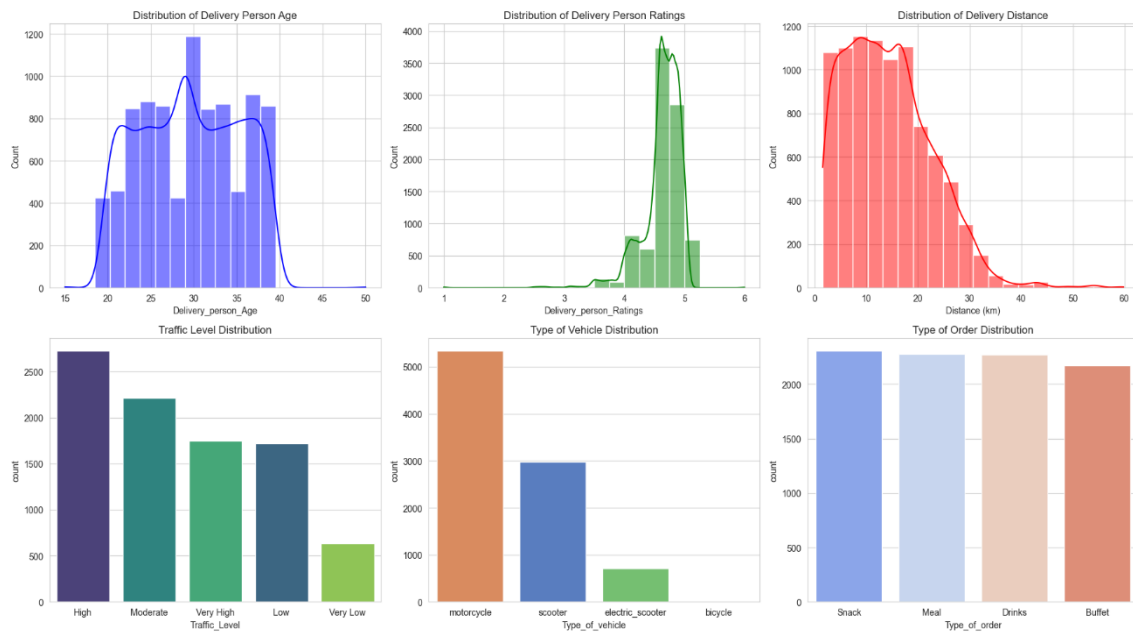


Figure 8: Distributions of various variables

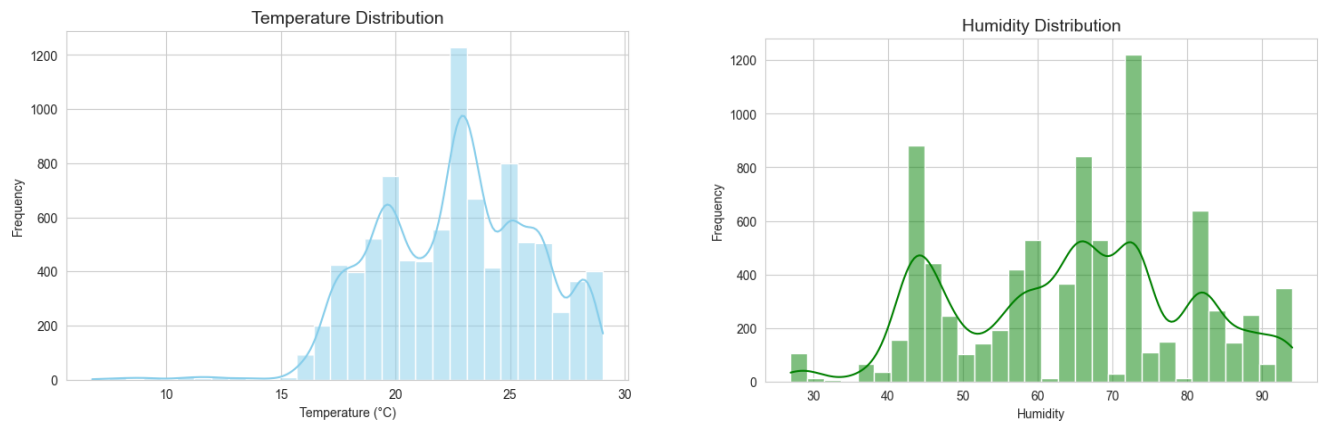


Figure 7: Temperature and Humidity Distribution

Explanations

- The age distribution of delivery person showed a peak at 30, with most ranging from 20 to 40 years. This indicates that the delivery force is made up mainly of young adults, perhaps because of the physical demands of the job and the flexibility it provides.
- Regarding delivery persons' ratings, most people were rated very highly, usually in the range 4.5 to 5. This would reflect consistent customer satisfaction or rating inflation, which tends to occur with user-based feedback systems.
- The pattern of delivery distances indicated that deliveries were concentrated within a 20-

kilometer radius. This is consistent with what would be expected in urban delivery environments, where service areas are geographically tight and dense.

- High traffic volumes were found to be most prevalent upon analyzing traffic volumes, followed by moderate and very high traffic volumes. This is consistent with the urban character of the data, where congestion is a typical problem that can have direct effects on delivery time and routing.
- The type of vehicle used for deliveries was largely motorcycles, with scooters a close second. Only a few respondents used electric scooters and bicycles. According to reason this is probably because motorcycles are lightweight and fast in maneuvering through heavy traffic environments.
- The distribution of order types was relatively balanced across snacks, meals, drinks, and buffet orders. This indicates a diversified demand pattern from customers and a well-represented sample in the dataset across various food categories.
- The temperature distribution shows that the data has been recorded at a place where temperature levels are low below 30.

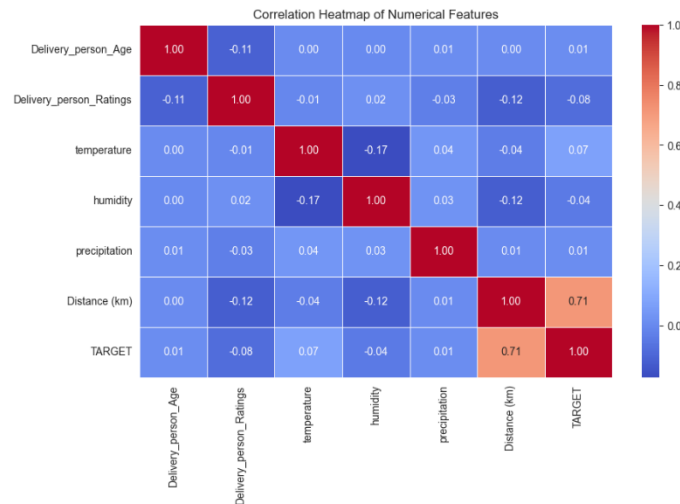


Figure 9: Correlation Heatmap

Explanation:

- The most notable correlation is observed between Distance (km) and Delivery Time (TARGET), with a coefficient of 0.71, indicating a strong positive correlation. This suggests that as distance increases, delivery time also increases substantially.
- Other numerical features such as temperature, precipitation, humidity, and

Delivery_person_Ratings show very weak correlations (near zero) with the target variable, suggesting minimal linear influence.

- Delivery_person_Age also shows an insignificant correlation with delivery time, implying that the age of the delivery personnel does not systematically impact delivery performance.

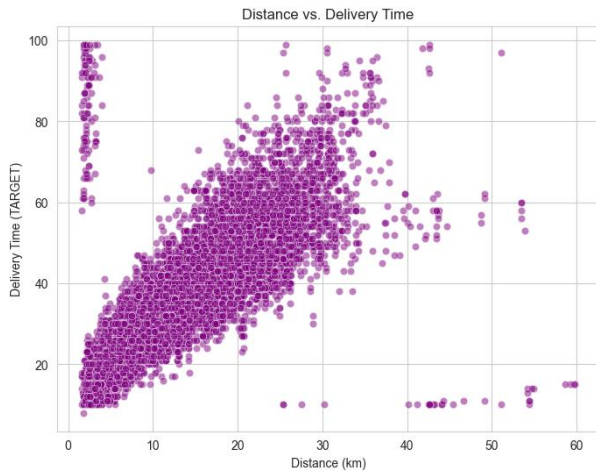


Figure 12: Scatterplot for Distance vs Delivery Time

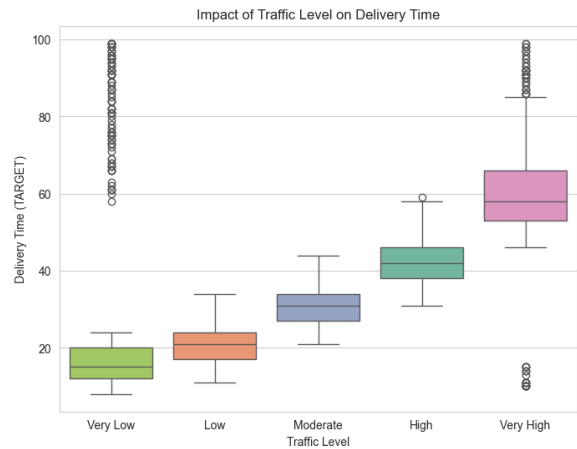


Figure 10: Boxplot to know impact of Traffic Level on Delivery Time

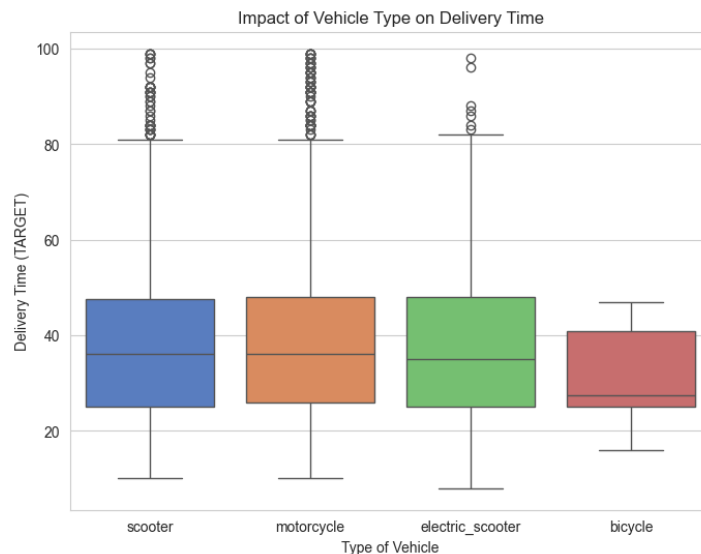


Figure 11: Impact of Vehicle on Delivery

Explanation for Figure 12: This scatter plot offers a visual validation of the heatmap findings by illustrating the relationship between distance traveled and delivery time.

- A clear positive linear trend is visible: as distance increases delivery time tends to rise.
- However, there is increased variability in delivery time at larger distances, possibly due to additional influencing factors such as traffic conditions, vehicle type, or route

complexity.

- Outliers are present at both ends, with some long-distance deliveries being completed in surprisingly short times, and vice versa.

Explanation for Figure 10: This box plot examines how traffic congestion levels affect delivery duration:

- A clear increasing trend in median delivery time is observed as traffic level progresses from Very Low to Very High.- The interquartile range (IQR) also widens with higher traffic levels, indicating greater variability in delivery times under congested conditions.
- The presence of outliers in all categories, especially at extreme traffic levels, reflects occasional delays or unusually quick deliveries.
- This suggests that traffic conditions have a significant and nonlinear effect on delivery performance, especially in high-traffic scenarios.

Explanation for Figure 11: Motorcycles have the quickest and most reliable deliveries, with the peak at 30-35 minutes, but there are occasional longer ones. Scooters come second, slightly less quick and less reliable, peaking 35-40 minutes, but are still effective. Electric scooters have more variability, taking 30-60 minutes, meaning slower and less reliable service. Bicycles are seldom employed, possibly due to restrictions on distance and demand, and are underrepresented in the data. Each mode's distribution reflects its suitability for urban delivery, with motorcycles excelling in speed and consistency.

3.4 FEATURE SELECTION AND ENGINEERING

3.4.1 Dropping Irrelevant Columns

Columns such as "*ID* " and "*Delivery_person_ID*" *do not give the model* any useful or predictive information. These columns are used to identify rows and their values uniquely. Therefore, these two columns were dropped.

The "*precipitation*" column was also dropped because the column contains mostly zeros which will not contribute much to prediction by the model.

3.4.2 Encoding Categorical Features

Categorical variables include *Traffic_Level*, *weather_description*, *type_of_order* and *type_of_vehicle*. Amongst this, *Traffic_Level* has a meaningful order (Very High, High, Moderate, and Low, Very low), thus implying that it is ordinal and the rest are nominal variables.

```
Value counts for 'Traffic_Level':
Traffic_Level
High      2529
Moderate  2126
Low       1666
Very High 1438
Very Low   556
Name: count, dtype: int64

Value counts for 'weather_description':
weather_description
clear sky      2879
haze           2223
mist           1648
broken clouds   499
smoke           458
scattered clouds 368
overcast clouds 161
fog             44
few clouds      35
Name: count, dtype: int64
```

Figure 13: Showing the values each category has

Encoding Traffic_Level: Before encoding, we see that there are five categories- Very High, High, Moderate, and Low, Very low. To reduce redundancy and create balance (from Figure 7 we see that there is an imbalance in the number of values each category has) in the value counts we minimise the number of categories by grouping them as given below:

- Very Low and Low → Low
- Moderate → Moderate
- High and Very High → High

Now, we use ordinal encoder to encode them into numerical values representing meaningful

```
#Making 5 categories in a column to 3 categories
traffic_mapping = {}
    'Very Low': 'Low',
    'Low': 'Low',
    'Moderate': 'Moderate',
    'High': 'High',
    'Very High': 'High'
}

df['Traffic_Level_Simplified'] = df['Traffic_Level'].map(traffic_mapping)

#Ordinal Encoding Traffic Level since it is ordinal in nature
traffic_order = [['Low', 'Moderate', 'High']]
encoder = OrdinalEncoder(categories=traffic_order)
df['Traffic_Level_Encoded'] = encoder.fit_transform(df[['Traffic_Level_Simplified']])
```

Figure 14: Using Ordinal Encoder to encode Traffic_Level

order.

Encoding weather_description: The weather_description column had several similar weather conditions, many of which had few occurrences (Figure 7). These were grouped into three consolidated categories:

- clear_sky, few_clouds → Clear
- haze, mist, fog, smoke → Hazy/Foggy
- scattered_clouds, broken_clouds, overcast_clouds → Cloudy

```
df = df[df['weather_description'] != 'moderate rain'].copy()
#Minimising the number of categories
weather_mapping = {
    'clear sky': 'Clear',
    'few clouds': 'Clear',
    'haze': 'Hazy/Foggy',
    'mist': 'Hazy/Foggy',
    'fog': 'Hazy/Foggy',
    'smoke': 'Hazy/foggy',
    'scattered clouds': 'Cloudy',
    'broken clouds': 'Cloudy',
    'overcast clouds': 'Cloudy'
}
df['weather_description'] = df['weather_description'].map(weather_mapping)
```

Figure 15: Grouping weather categories

The rare moderate rain category was excluded from the dataset to improve consistency.

Encoding nominal variables: The remaining nominal variables were encoded by creating a new column for each category and assigning values 0 or 1 based on whether they are present or not.

```
#One Hot encoding nominal variables
nominal_cols = ['weather_description', 'Type_of_order', 'Type_of_vehicle']
df_encoded = pd.get_dummies(df, columns=nominal_cols, drop_first=True)

df_encoded = df_encoded.astype({col: 'int' for col in df_encoded.select_dtypes(include='bool').columns})
```

Figure 16: Encoding nominal variables

3.4.3 Feature selection through feature importance

To see the contribution of each feature to predicting delivery time, we use Random Forest Regressor. We split the data into a training set and a test set and trained the model on the entire feature set.

The feature importance scores were retrieved using the model's *feature_importances_* attribute, which assesses the relative contribution of each feature towards minimizing prediction error. A bar plot was produced to show the importance scores of all features. This analysis offered

insights into what variables most contributed to delivery time, aiding interpretability as well as possible future feature selection efforts.

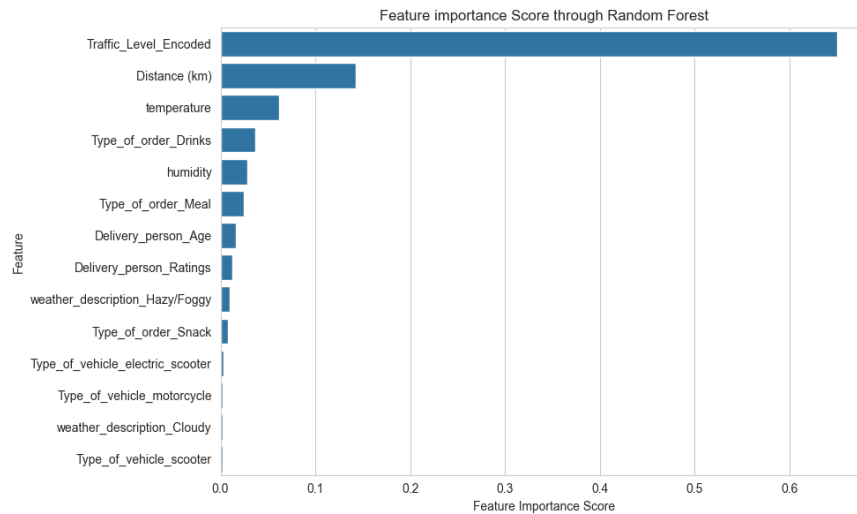


Figure 17: Feature Importance Score by Random Forest

From Figure 11, we understand that *Traffic_Level* is the feature that contributes most to predicting the *TARGET*, followed by *Distance(km)*, *temperature* and *type_of_order*.

- We set the threshold at 0.02 and select the features above it, except for *Delivery_person_ratings*. If we check the standard deviation of ratings being very low, this proves that ratings do not vary much in their values, which may result in lower prediction rates.
- We select *weather_description* even though the scores are low because temperature and humidity itself does not make sense on how target varies.

Selected Features: *'Traffic_Level_Encoded'*, *'Distance (km)'*, *'Delivery_person_Age'*, *'weather_description_Hazy/Foggy'*, *'weather_description_Cloudy'*, *'temperature'*, *'humidity'*, *'Type_of_order_Drinks'*, *'Type_of_order_Meal'*, *'Type_of_order_Snack'*.

3.5 MODEL DESCRIPTION

To predict delivery time effectively, we employed two powerful machine learning models: Random Forest Regressor and XGBoost Regressor. These models were chosen due to their proven performance in handling complex, non-linear relationships in tabular data, robustness to outliers, and minimal preprocessing requirements.

3.5.1 Model Selection

- Random Forest Regressor is a decision tree ensemble method. It builds many trees on random samples of the data and takes a mean of the predictions. It does this in an

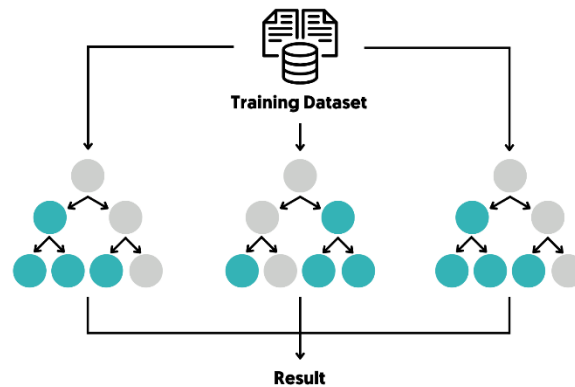


Figure 18: Working of Random Forest

attempt to reduce variance and avoid overfitting. It also has good inherent ability at estimating feature importance, which will aid in interpretation.

- XGBoost Regressor (Extreme Gradient Boosting) is a boosting algorithm that constructs trees sequentially, with each subsequent tree targeting the residuals of

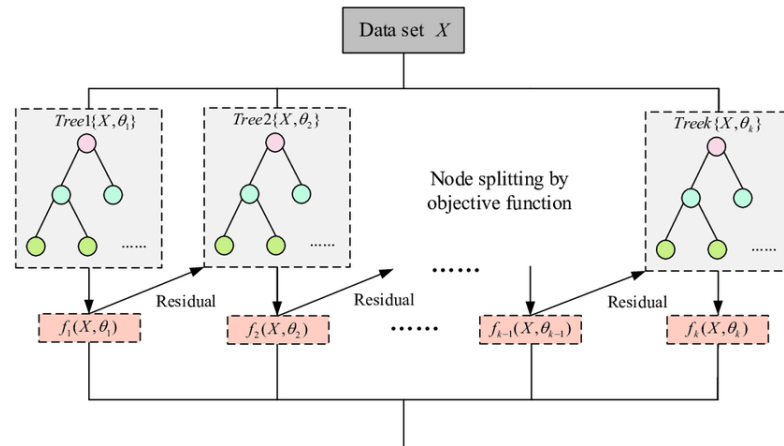


Figure 19: Working of XGBoost

the preceding ones. It is efficient, scalable, and can handle regularization, which helps to manage overfitting and enhance generalization.

3.5.2 Evaluation Metrics

- **R² Score (Coefficient of Determination):** Indicates the proportion of variance in the target variable explained by the model. An R² closer to 1 signifies better predictive performance.
- **RMSE (Root Mean Squared Error):** Captures the average magnitude of prediction errors. It penalizes large deviations and provides a useful interpretation in the same units as the target variable (delivery time in minutes).

3.5.3 Hyperparameter Tuning

In machine learning, hyperparameters are configuration variables set before training begins. These differ from model parameters (like weights in linear regression or nodes in decision trees), which are learned from the data during training. Examples of hyperparameters include the number of trees in a forest (`n_estimators`), the depth of a tree (`max_depth`), and the learning rate in boosting algorithms. Choosing the right combination of hyperparameters is critical to building an effective and efficient model. This process is known as hyperparameter tuning.

RandomizedSearchCV, a technique provided by scikit-learn, for efficient hyperparameter tuning. Rather than evaluating every single combination of hyperparameters (as in GridSearchCV), RandomizedSearchCV:

- Randomly samples a fixed number of combinations from the search space.
- Performs cross-validation on each sample to estimate model performance.
- Reduces computational cost while often finding comparable or even better results than grid search.

This approach is useful when:

- The search space is large or high-dimensional.
- You have limited computational resources or time.

- Some hyperparameters are more important than others, and full grid search would waste time on unimportant combinations.

3.6 TOOLS AND LIBRARIES

- **pandas** and **numpy**: For data manipulation, cleaning, and handling structured datasets.
- **matplotlib** and **seaborn**: For data visualization, including distribution plots, correlation heatmaps, and feature importance charts.
- **scikit-learn (sklearn)**: For key machine learning tasks:
 - **Modeling**: Random Forest Regressor
 - **Data Preprocessing**: Train-test split, Ordinal Encoding
 - **Hyperparameter Tuning**: RandomizedSearchCV
- **Evaluation Metrics**: R^2 score, Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE)
- **XGBoost (xgboost)**: A high-performance gradient boosting algorithm used for regression modeling due to its speed and accuracy.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 ABOUT THE DATASET

The data set employed in the study consists of detailed delivery records covering numerous operational and environmental factors that affect delivery performance. The target variable is delivery time in minutes. It combines numerical attributes like delivery distance, age and rating of the delivery staff, temperature, and humidity, as well as categorical variables like traffic level, weather, type of order, and type of vehicle. In order to maintain data quality and consistency, missing value records were eliminated. The target column was also preprocessed by removing unrealistic or placeholder values. Additionally, outliers in important numerical features were detected and removed through the Interquartile Range (IQR) approach to reduce skewness and improve model stability.

In order to improve the predictive power of the dataset, irrelevant features like identifiers and columns with predominantly zero values were eliminated. Categorical features were encoded properly: ordinal encoding was used for traffic levels and one-hot encoding for nominal features such as weather and order type. Moreover, categories were reduced—e.g.; weather conditions were classified into three more general categories for improved generalization. A correlation heatmap was produced, which showed that delivery distance had the highest linear correlation with delivery time, and other features such as age, ratings, and weather had lower correlations. Lastly, feature importance was determined through a Random Forest model, where distance, traffic level, and type of order were among the most significant predictors.

4.2 MODEL PERFORMANCE AND OUTPUT

Model Evaluation before Hyperparameter Tuning

```
print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R2 Score:", r2)
```

✓ 1.8s

MAE: 3.2734486727944336
MSE: 28.276323972408687
RMSE: 5.317548680774694
R2 Score: 0.8692773261078028

Figure 21: Random Forest Model Performance

```
print(f"R² Score: {r2:.4f}")
print(f"RMSE: {rmse:.2f}")
```

✓ 0.1s

R² Score: 0.8690
RMSE: 5.32

Figure 20: XGBoost Model Performance

Model	R-Squared	RMSE
Random Forest	0.869	5.31
XGBoost	0.869	5.32

Model Evaluation after Hyperparameter Tuning

```
print("Best Params for RF:", random_search_rf.best_params_)
print(f"R² Score: {r2:.4f}")
print(f"RMSE: {rmse:.2f}")
```

✓ 47.9s

Fitting 5 folds for each of 25 candidates, totalling 125 fits
Best Params for RF: {'n_estimators': 200, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'log2', 'max_depth': 40}
R² Score: 0.8690
RMSE: 5.32

Figure 22: Random Forest Hypertuning Results

```
rmse = np.sqrt(mse)
print("Best Params:", random_search.best_params_)
print(f"R² Score: {r2:.4f}")
print(f"RMSE: {rmse:.2f}")
```

✓ 7.3s

Fitting 5 folds for each of 25 candidates, totalling 125 fits
Best Params: {'subsample': 0.8, 'n_estimators': 200, 'max_depth': 6, 'learning_rate': 0.05, 'colsample_bytree': 0.8}
R² Score: 0.8731
RMSE: 5.24

Figure 23: XGBoost HyperTuning results

Model	R-Squared	RMSE
Random Forest	0.869	5.32
XGBoost	0.8731	5.24

To evaluate model performance, we employed two key metrics: R-Squared (R^2) and Root Mean Squared Error (RMSE). The R^2 score indicates how well the independent variables explain the variability of the target variable, while RMSE measures the standard deviation of the prediction errors, expressed in the same units as the target (minutes).

The following table presents the performance of both the Random Forest and XGBoost models, before and after hyperparameter tuning:

Model	R^2 (Before)	RMSE (Before)	R^2 (After)	RMSE (After)
Random Forest	0.869	5.31	0.869	5.32
XGBoost	0.869	5.32	0.8731	5.24

Both models initially demonstrated strong predictive performance, with R^2 scores around 0.869, indicating that nearly 87% of the variance in delivery time is explained by the features. After tuning, the XGBoost model showed a modest yet meaningful improvement, achieving an R^2 of 0.8731 and a reduced RMSE of 5.24 minutes, making it the best-performing model among the two.

```
# Random Forest Prediction
rf_pred_1 = predict_delivery_time(best_rf_model, traffic_level=2, distance=5.0, age=29,
                                  temperature=28.0, humidity=75.0,
                                  weather_desc='Cloudy', order_type='Meal')

# XGBoost Prediction
xgb_pred_1 = predict_delivery_time(best_xgb_model, traffic_level=2, distance=5.0, age=29,
                                   temperature=28.0, humidity=75.0,
                                   weather_desc='Cloudy', order_type='Meal')

print(f"RF Prediction (Test 1): {rf_pred_1} minutes")
print(f"XGB Prediction (Test 1): {xgb_pred_1} minutes")

✓ 0.0s
RF Prediction (Test 1): 37.08 minutes
XGB Prediction (Test 1): 35.08000183105469 minutes
```

Figure 24: Hardcoded Prediction set

To evaluate the practical utility of the models developed, a comparative prediction was made using both Random Forest and XGBoost models on a uniform test scenario. The input

parameters were a traffic level represented as High, a 5.0-kilometer delivery distance, age of delivery personnel as 29 years, ambient temperature of 28°C, humidity as 75%, weather as Cloudy, and type of order as Meal.

The Random Forest model estimated a delivery time of 37.08 minutes, while the XGBoost model estimated a slightly lower value of 35.08 minutes. This small difference indicates that although both models are capable of producing reasonably accurate estimates, XGBoost might be able to capture underlying data patterns with slightly higher accuracy, most likely because of its boosting-based ensemble learning paradigm. Nevertheless, the closeness of predictions confirms the strength and dependability of both models under normal operating conditions.

The trained Random Forest and XGBoost models were serialized using Python's pickle module and subsequently integrated into a Streamlit web application for real-time prediction. This enabled users to input relevant delivery parameters and receive estimated delivery times. The resulting output from the application is shown below in Figure 25.

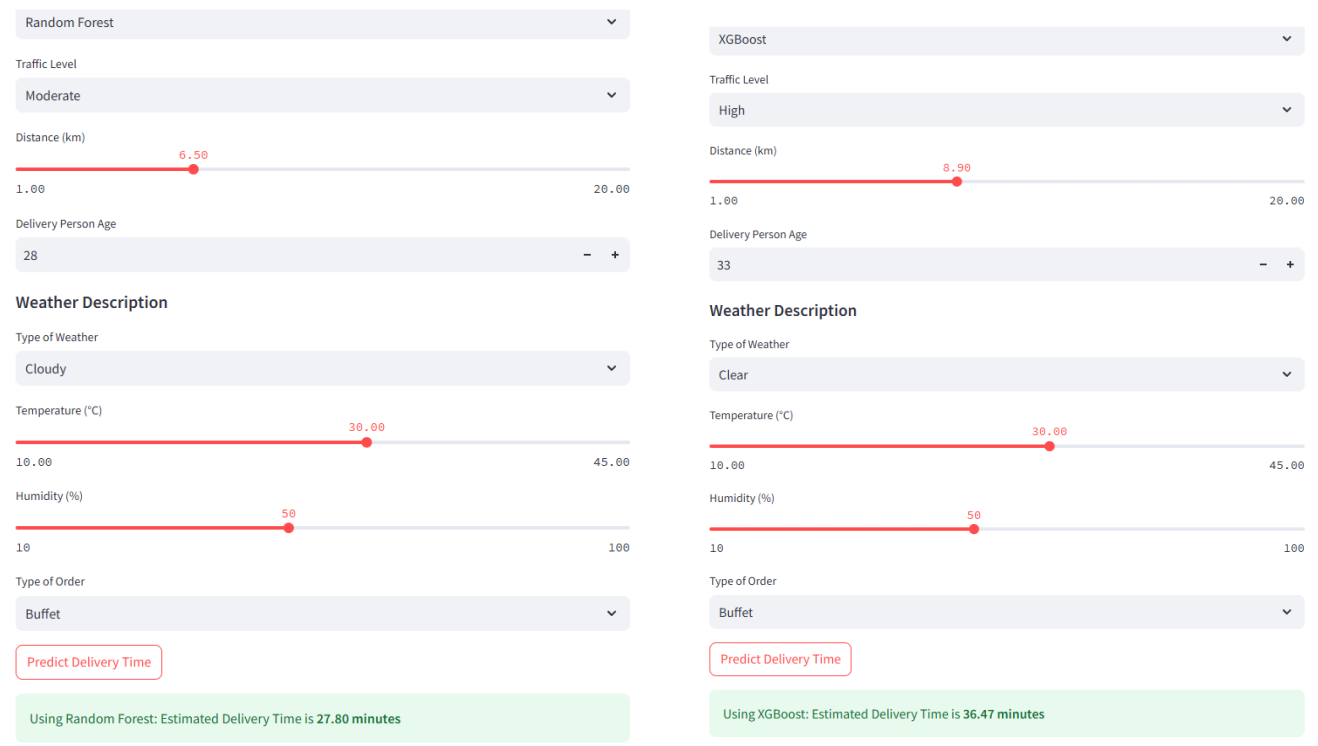


Figure 25: Streamlit Implementation of the Model

4.3 CHALLENGES AND LIMITATIONS

Although the Random Forest and XGBoost models achieved robust performance in predicting delivery time, numerous issues and limitations were faced throughout the process. To begin, the dataset contained noise and inconsistencies, including poorly formatted coordinates and a vast range of delivery times, which required significant data cleaning and removal of outliers. Further, while tree models are robust, they will also overfit to training data patterns if not tuned properly — that is why hyperparameter optimization was necessary.

Another major limitation is that these models take static relationships between features and outcomes, whereas actual delivery times can be affected by dynamic, real-time variables like abrupt traffic patterns, weather conditions, or order cancellations — things not included in our dataset. Additionally, categorical simplification (e.g., combining traffic levels or weather types) could result in loss of granularity, which could influence model subtlety.

Lastly, interpretability is also an issue with ensemble techniques like XGBoost, where interpreting feature contributions by a single feature may not be directly possible without using extra tools like SHAP values.

4.4 LINK TO THE DATASET AND CODE FILE(.ipynb file)

<https://drive.google.com/drive/folders/1MJFOc2BL7zrkWkd6U4PDyU83kVh60l-y?usp=sharing>

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

In this project, our goal was to forecast food delivery time with the help of machine learning models and a dataset densely populated with realistic delivery features. By thoroughly preprocessing the data—addressing missing values, outliers, and categorical variable encodings—we achieved a clean, well-organized dataset for training the model. Feature engineering and feature selection optimized the input space in order to increase the learning potential of the model.

Two strong ensemble techniques, Random Forest and XGBoost, were used for prediction, with hyperparameter tuning through Randomized Search CV improving their performance. XGBoost performed marginally better than Random Forest, with an R^2 value of 0.8731 and an RMSE of 5.24 minutes. The models showed strong accuracy in predicting the relationships between delivery-related variables and the target outcome.

In order to make the model accessible and intuitive, we deployed it with Streamlit, enabling users to feed in delivery conditions and obtain immediate predictions. The project not only demonstrates the predictive power of machine learning in logistics but also offers a useful tool for estimating delivery times in real-world applications.

5.2 FUTURE WORK

Many promising directions can extend and improve this project on predictive modeling:

Feature Engineering: Create additional features like time-based ones (hour of the day, day of the week), geographical clusters, or weather conditions that would spur changes in delivery times.

Advanced Models: Investigate Random Forest, Gradient Boosting, and Neural Networks as non-linear algorithms potentially more capable of capturing complex relationships missed by linear algorithms.

Ensemble Methods: Formulate model predictions from any number of models to obtain higher accuracy than any single model can achieve.

Time Series Analysis: Factor in temporal patterns and seasonality with variations in delivery times with different periods.

Explainable AI: Employing explanatory techniques to help stakeholders dissect model predictions to rationalize which factors most influence delivery times.

Model Deployment: An API or web interface for the operational teams to gain real-time access to predictions.

Feedback Loop: A means of continuously updating the model with new delivery data to keep its accuracy over time as conditions change.

Spatial Analysis: Extending geospatial modeling to better cater to location-centric factors influencing delivery time.

Customer Segmentation: Specific modeling for the various orders done by customers or customer segments are done to enhance prediction specificity.

REFERENCES

- [1] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.
<https://doi.org/10.1023/A:1010933404324>
- [2] Biau, G. (2012). Analysis of a random forests model. *Journal of Machine Learning Research*, 13(Apr), 1063–1095. <https://jmlr.org/papers/volume13/biau12a/biau12a.pdf>
- [3] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). <https://arxiv.org/abs/1603.02754>
- [4] <https://www.kaggle.com/code/gauravmalik26/food-delivery-fe-stacking-and-ensemble>