APPLIED MACHINE LEARNING

PREDICTING ESTIMATED DELIVERY TIME FOR FOOD ORDERS

GROUP 11 (BUAN 6341.003)

Ron Chaudhuri - rxc240006
Haritha Rajendra Rao Savithri - dal136929
Salwa Niaz Preet – sxp240086
Naveena Paleti – nxp230055
Yuting Zhou - yxz230037

Introduction

In an era where food delivery services are becoming increasingly popular, accurate prediction of delivery time is critical for customer satisfaction and operational efficiency. Delayed deliveries often result in customer dissatisfaction, impacting brand reputation and repeat business. Conversely, accurate Estimated Time of Arrival (ETA) enhances customer trust, optimizes resource allocation, and improves the overall delivery network management.

This project focuses on predicting the estimated delivery time for food orders using advanced machine learning techniques. By leveraging historical data and external factors, we aim to create a robust model capable of providing accurate delivery time predictions, helping businesses streamline operations and enhance customer experiences.

Problem Statement

The primary objective of this project is to build predictive models that accurately estimate the delivery time for food orders. Several internal and external variables affect delivery time, including the delivery person's age, rating, weather conditions, traffic density, vehicle condition, order type, and delivery distance.

Accurate delivery time predictions can:

- Improve customer satisfaction by setting realistic expectations
- Enhance operational efficiency by optimizing delivery routes and resource allocation
- Reduce delivery-related complaints and enhance service quality
- Aid strategic planning during peak hours, festivals, and adverse weather conditions

Dataset Summary

The dataset has been taken from Kaggle. The data is for food delivery in India. It includes the target variable "Time Taken" in minutes. It is the time taken to deliver food from the restaurant to the destination after being picked. All other variables can be used to predict the delivery time taken. The variables can be divided into the following categories:

Delivery Person Info
Delivery_person_Age, Delivery_person_Ratings, multiple_deliveries (yes, no), Vehicle_condition (0, 1, 2, 3 with 3 being the best condition), Type_of_vehicle (bicycle, scooter, motorcycle, electric scooter)

Order & Timing Details
Order_Date, Time_Ordered, Time_Order_picked, Order_to_Pickup_Duration,Type_of_order (buffet, drink, meal, snack)

Geolocation
Restaurant_latitude/longitude, Delivery_location_latitude/longitude
Using this information, we were able to create a new variable, "delivery distance in km".

External Factors
Weatherconditions (sunny, windy, storm, fog, sandstorm, cloudy), Road_traffic_density, Festival (yes, no), City (urban, semi urban, metropolitan)

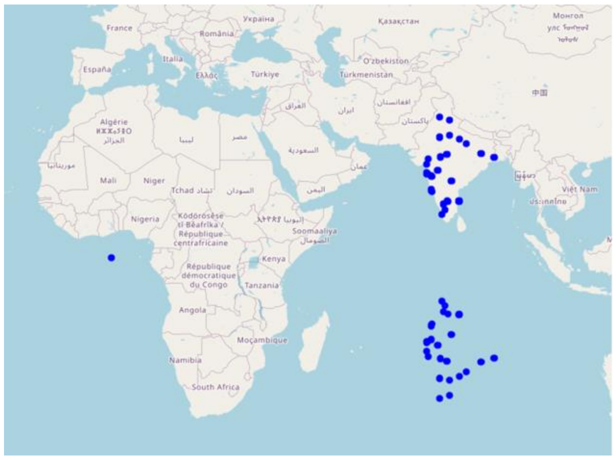
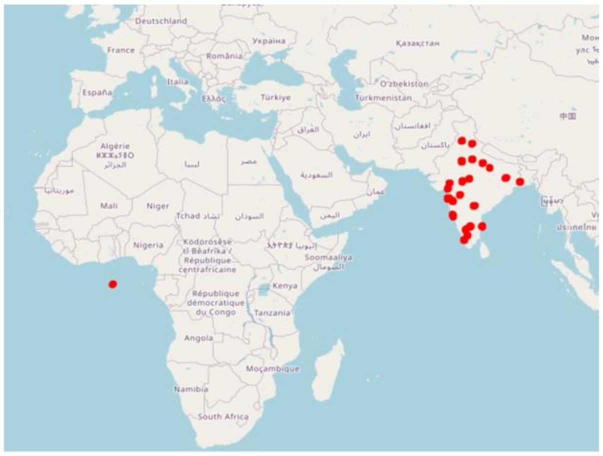
Data Preprocessing

A comprehensive data preprocessing pipeline was applied to ensure the dataset was clean, consistent, and suitable for machine learning. Text artifacts such as "(min)" and "conditions" were removed, and all numerical and datetime fields were converted to appropriate types. Missing values were handled through a combination of row removal and mode imputation for key fields like Delivery_person_Ratings, Festival, and City. Derived features were created, including Delivery_distance_km (using geodesic distance between coordinates) and Order_to_Pickup_Duration (in minutes).

Categorical variables were transformed using a mix of ordinal and one-hot encoding, while boolean features were converted to binary values. To ensure equal feature contribution—especially for models sensitive to scale like KNN and MLP—all numerical features were standardized using StandardScaler. Finally, the dataset was split into training and testing sets using an 80/20 stratified split (random_state = 42).

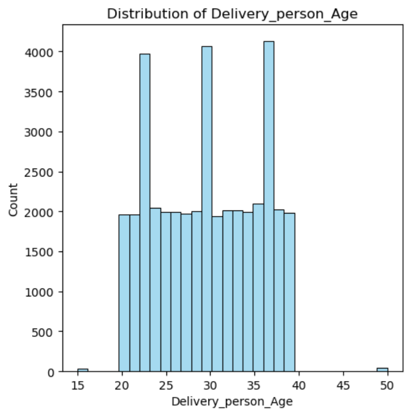
All machine learning models in the subsequent sections were developed using this uniformly preprocessed dataset.

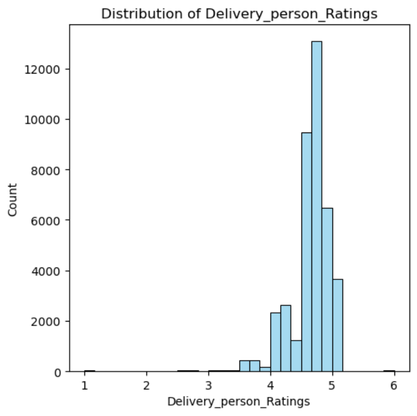
Exploratory Data Analysis

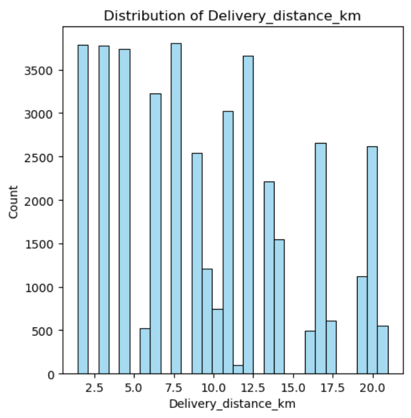



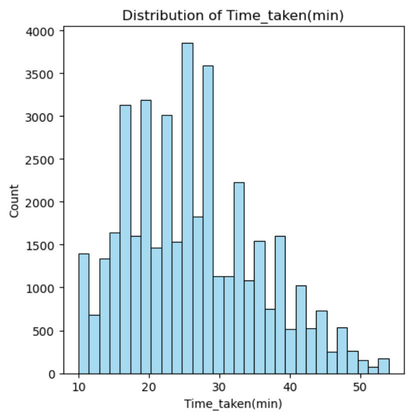
The figure on the left with blue points shows locations of the restaurants and the one on the right shows delivery locations. As mentioned in the dataset summary section of the report, the data was exclusively for India, however, the maps above show some locations not on the Indian map. It was observed that some of these longitude and latitude values were either zero or negative. Taking absolute values resolved the issue for negative geolocation observations. Rows with the zero values were removed entirely as we did not want these observations to bias our results. As a matter of fact, the total number of such values was about 600 (roughly 2% of the dataset).

Distributions of Numerical Variables



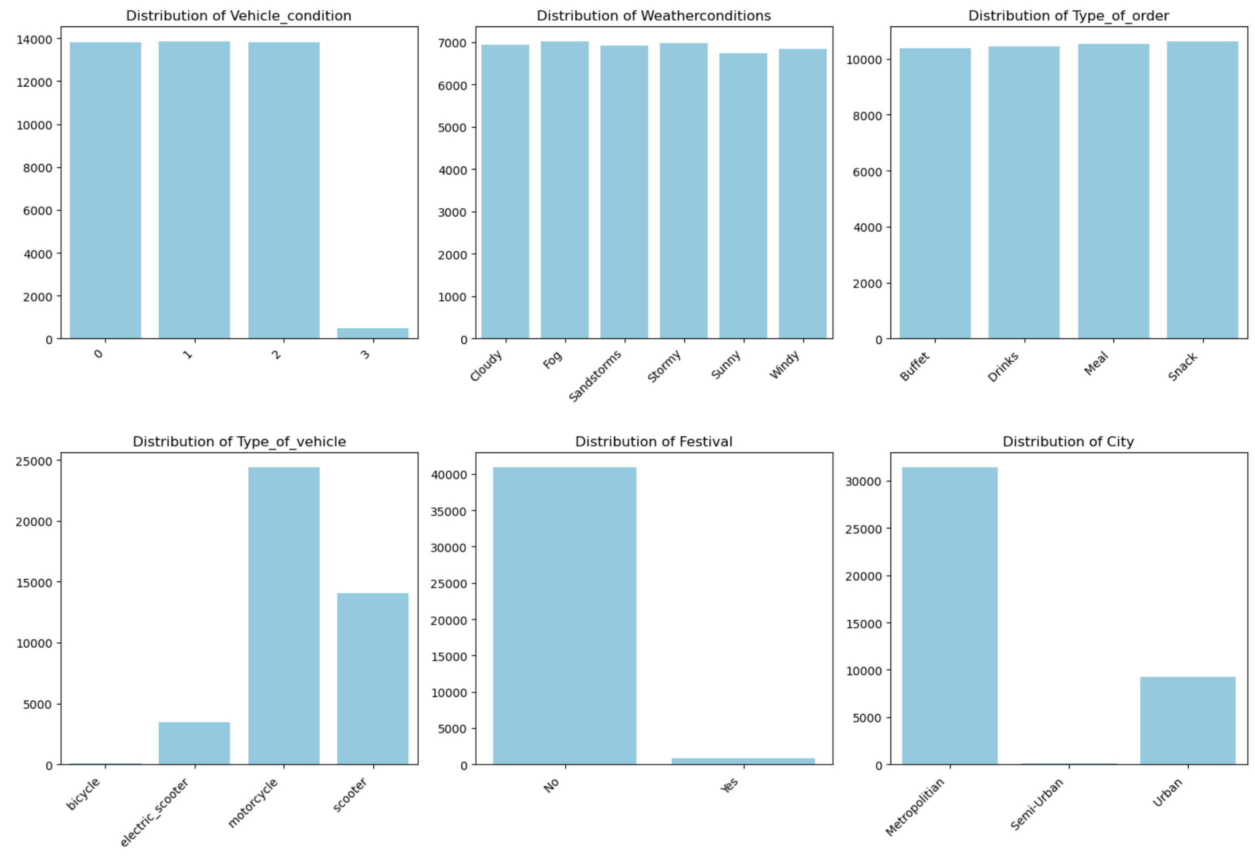


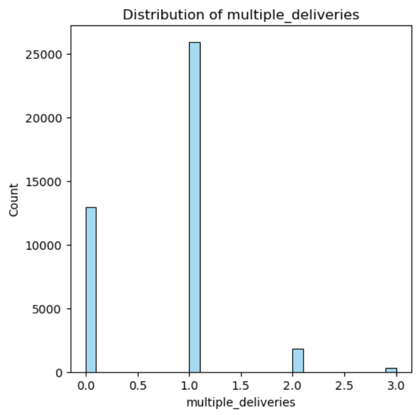


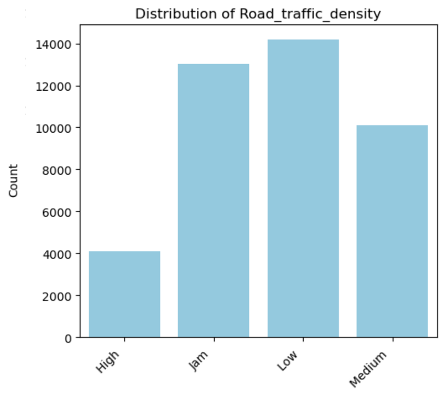


The graphs presented here illustrate the distribution of key numerical variables, including delivery person age, delivery person ratings, and delivery distance. The age distribution indicates that the majority of drivers fall within the 35 to 40-year range, with the overall age span primarily concentrated between 20 and 40 years. This suggests that the delivery workforce is largely composed of individuals in early to mid-adulthood, which may have implications for physical stamina and delivery efficiency. The ratings of delivery personnel show a negatively skewed distribution, with most drivers receiving high ratings in the 4 to 5 range, and only a small number of individuals rated between 1 and 4. This skew suggests generally strong customer satisfaction or performance among the majority of delivery staff. In contrast, the distribution of delivery distance is more widespread, ranging from approximately 2.5 kilometers to 20 kilometers. This indicates a high degree of variability in delivery assignments, which may contribute to fluctuations in delivery time depending on route complexity and geographic coverage. Overall, these distributions offer valuable insights into workforce demographics, service quality, and logistical patterns within the delivery system.

Distributions of Categorical Variables

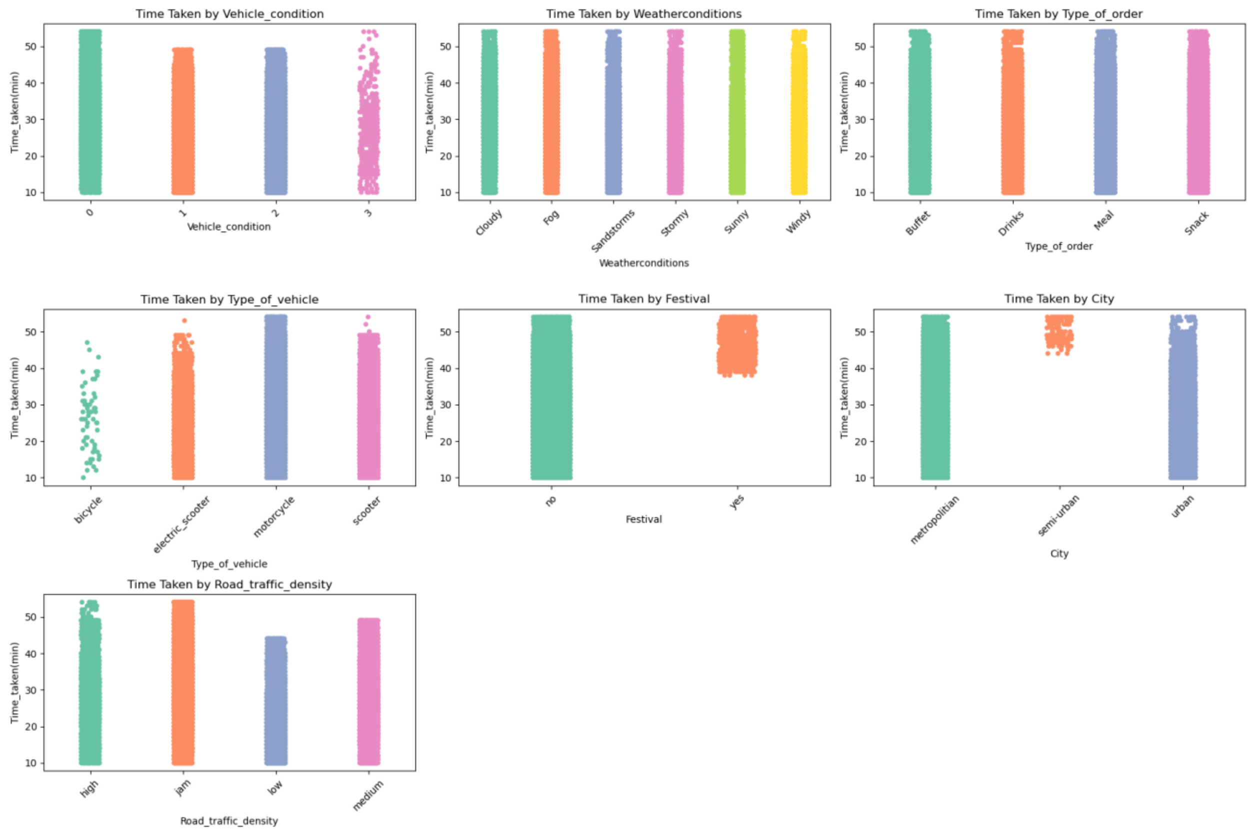






The distributions of categorical variables are useful for comparing the sizes of different groups, identifying the most or least common categories, and spotting potential data entry issues such as typos or missing values. While some categorical variables like weather conditions and type of order have a roughly equal number of observations for different categories, others like type of vehicle, festival, city, road traffic density and multiple deliveries have a clear modal class. We see that low traffic density, motorcyle as mode of transport, no festival day and metropolitan city are the most frequent data entry points. The graphs even helped impute missing values with the mode in the festival and city columns of the dataset.

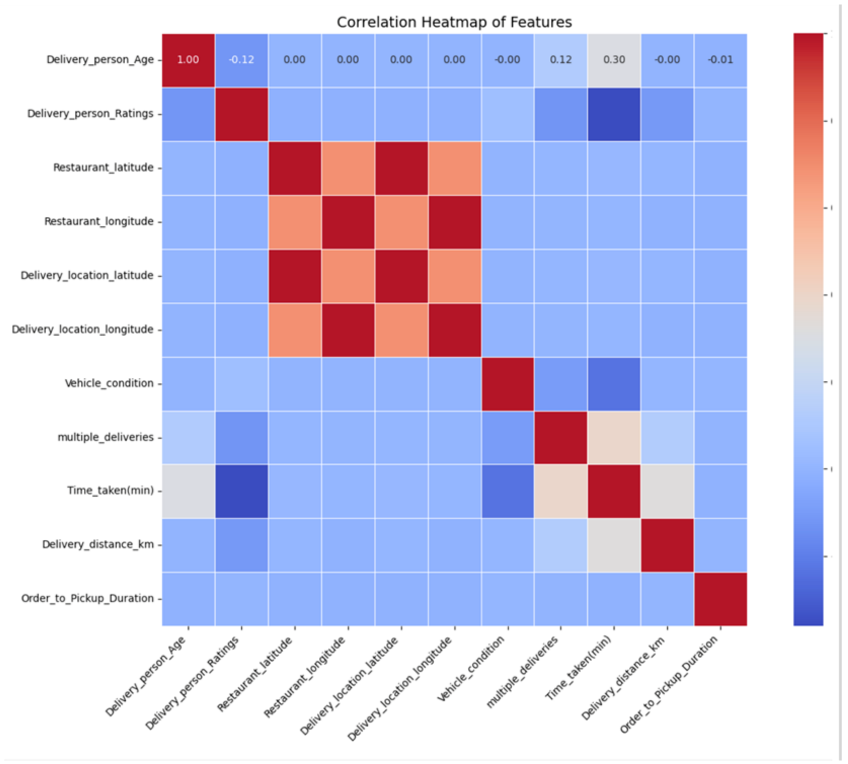
Scatter Plots



The scatter plots show the relationship between the categorical variables and the target variable, time taken to deliver. While most individual categories do not have a distinct relationship with the target variable, we see that bicycles have lesser delivery times, perhaps because bicycles are mainly used for closer delivery locations. Festival days take more time for orders to be delivered, mainly because of an influx of orders on such days. Delivery times in semi urban cities are higher possibly because the road infrastructure isn't well maintained or restaurants are farther away from the delivery locations.

Correlation Heatmap

Features such as delivery distance, driver age, and the number of deliveries assigned to a driver exhibit a positive correlation with delivery time, meaning that as these variables increase, the time it takes to complete a delivery tends to increase as well. Longer distances naturally require more travel time, while older drivers or those handling multiple deliveries simultaneously may experience delays due to physical limitations, routing inefficiencies, or time spent at each stop. On the other hand, delivery person ratings and the condition of the delivery vehicle show a negative correlation with delivery time. Higher ratings often reflect greater experience, efficiency, and customer service skills, all of which can contribute to quicker deliveries. Similarly, well-maintained vehicles are less likely to experience mechanical issues or slowdowns, leading to faster and more reliable service. These relationships highlight the multifaceted nature of delivery performance and suggest key areas for operational improvement, such as optimizing route assignments, monitoring vehicle maintenance, and investing in driver training and performance management.



Machine Learning Models

K-Nearest Neighbors (KNN)

K-Nearest Neighbors Regression (KNeighborsRegressor) is a simple, instance-based machine learning algorithm that makes predictions for a new data point by averaging the target values of the ‘k’ nearest training samples based on feature similarity (usually Euclidean distance). It is non-parametric and highly interpretable, but can be sensitive to data scaling and irrelevant features.

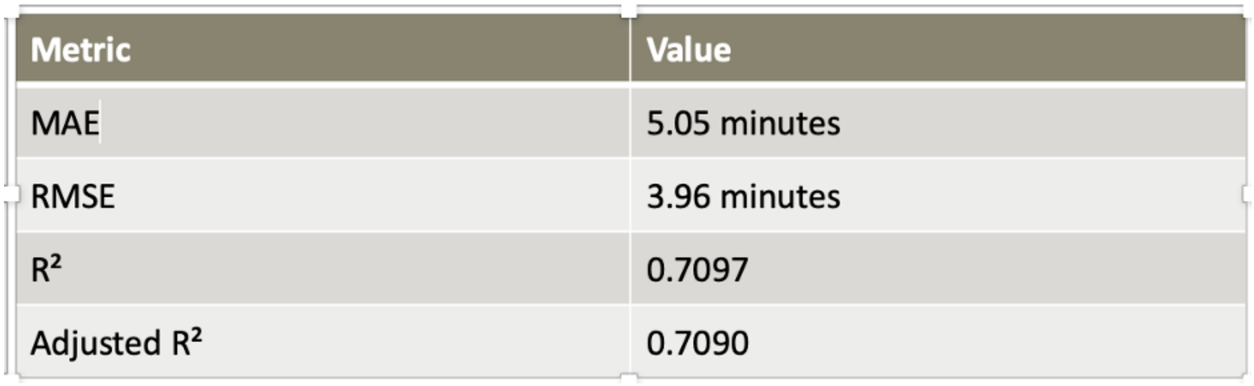
Key Features:

- Lazy learner: no explicit training phase
- Sensitive to feature scaling (standardization or normalization is essential)
- Local approximation — predictions are influenced by nearby observations

Hyperparameters:

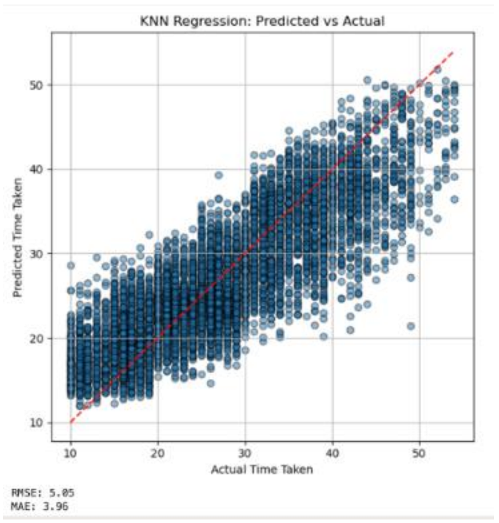
- n_neighbors = 15
- Specifies the number of nearest neighbors used for prediction. A larger k helps smooth out noise but may oversimplify the model.
- weights = ‘distance’
- Closer neighbors have more influence on the prediction.
- metric = ‘minkowski’
- Distance function used (default is Euclidean distance when p=2).

Performance Metrics:



Metric	Value
MAE	5.05 minutes
RMSE	3.96 minutes
R^2	0.7097
Adjusted R^2	0.7090

KNN: Predicted vs Actual Delivery Time



The scatter plot shows a moderately dispersed pattern around the ideal line (Predicted = Actual), indicating average accuracy. The model captures general trends but suffers from reduced performance in extreme or outlier cases due to its sensitivity to local density and variance.

Decision Tree Regressor

Decision Tree Regression (DecisionTreeRegressor) is a tree-based model that splits the dataset into subsets based on feature thresholds that minimize error in predicting the target variable. It is intuitive and interpretable but may suffer from overfitting unless constrained.

Key Features:

- Can model complex, non-linear relationships
- Does not require feature scaling
- Prone to overfitting on noisy data if not pruned or regularized

Hyperparameters:

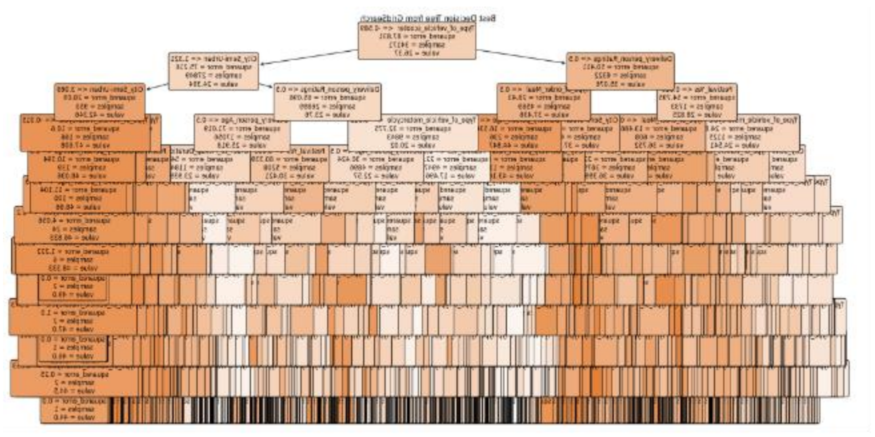
- random_state = 0
- Ensures reproducibility of results.
- max_depth = None
- No depth limit, allowing the tree to grow fully unless constrained by other parameters.
- min_samples_split = 2

- Minimum number of samples required to split an internal node.

Performance Metrics:

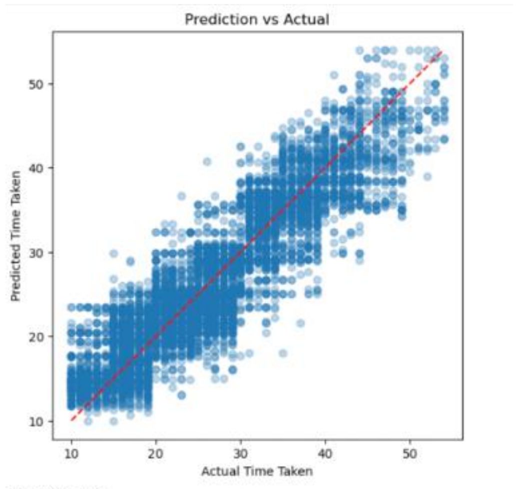
Metric	Value
MAE	3.22 minutes
RMSE	4.06 minutes
R^2	0.8125
Adjusted R^2	0.8120

Figure X. Decision Tree Structure



The tree illustrates how features like delivery distance and pickup time influence predicted delivery time through a series of binary splits.

Decision Tree: Predicted vs Actual Delivery Time



```
Test MSE: 4.06
```

The plot of predicted versus actual values shows a moderate clustering along the red diagonal line, indicating good predictive power. However, the presence of sharp decision boundaries can lead to slight overfitting, especially in sparse regions of the feature space.

Random Forest

Random Forest is a powerful ensemble learning method that builds multiple decision trees and combines their results to improve predictive accuracy and control overfitting. To accurately predict the delivery time of food orders by leveraging the strengths of Random Forest, which balances model complexity and generalization by averaging predictions across many decision trees.

Key Features:

- Handles both categorical and numerical features with minimal preprocessing
- Reduces overfitting by averaging across multiple trees
- Provides feature importance, allowing interpretation of key delivery-time drivers
- Robust to outliers and non-linear patterns in data

Hyperparameters:

Parameter	Value
objective	RandomForest Regressor
Random_state	42
n_estimators	100

Description of Hyperparameters:

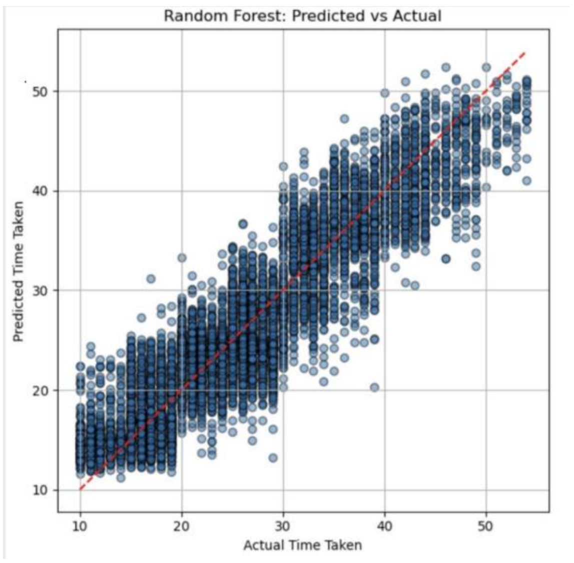
objective = RandomForest Regressor
Specifies the model type to be used. In this case, it is a regression model from the Random Forest family, which combines predictions from multiple decision trees to improve accuracy and control overfitting.
Random_state = 42

Ensures **reproducibility** of results by setting a fixed seed for the random number generator. This
n_estimators = 100
The number of decision trees to be built in the ensemble. More trees typically improve performance up to a point but also increase computation time.

Performance Metrics:

Metric	Value
MAE	3.08 minutes
RMSE	3.85 minutes
R^2	0.830
Adjusted R^2	0.8304

Random Forest: Predicted vs Actual Delivery Time



This scatter plot compares the **predicted delivery times** generated by the **Random Forest model** to the **actual delivery times** from the test dataset.

- The red dashed line represents the ideal scenario where the predicted time exactly equals the actual time taken (i.e., Predicted = Actual).
- The majority of data points cluster closely around the red line, indicating that the Random Forest model provides high prediction accuracy for most food delivery scenarios.
- There are some deviations—especially at the extreme lower and upper ends of the time range—suggesting that the model may slightly over- or under-predict in edge cases.

These could be due to uncommon traffic, multiple deliveries, or exceptional weather conditions.

- Overall, this visual evidence supports the model's strong R^2 score (0.830) and low RMSE (3.85 minutes), affirming that the Random Forest model generalizes well across the entire delivery time range.

SVR + MLP Regressor Stacked Pipeline

Overview

We build a two-stage stacking pipeline that leverages the strengths of a kernel-based SVR and a neural-network MLP, then blends their predictions with a Ridge meta-learner. We also tested two of the models individually to check whether the combined model was a improvement over each of the individual ones.

Pipeline & Components

- **Preprocessing:**
  - Mean-imputation of any remaining missing values
  - (All features already scaled/encoded in preprocessing stage)
- **Base Learners:**
  - **SVR (RBF kernel)**
    - Captures smooth, nonlinear trends via the Gaussian kernel
    - Regularization (C) and tube width (ε) tuned for generalization
  - **MLP Regressor**
    - Two hidden layers, ReLU activation
    - Tuned over layer sizes, L2 penalty (α), learning rate, tolerance, batch size
    - Trained efficiently with successive-halving on the epoch budget
- **Meta-Learner:**
  - **RidgeCV** with built-in cross-validation
  - Takes base-learner predictions (passthrough=True) plus original features
  - Learns optimal linear blend to minimize overall error

Hyperparameter Tuning

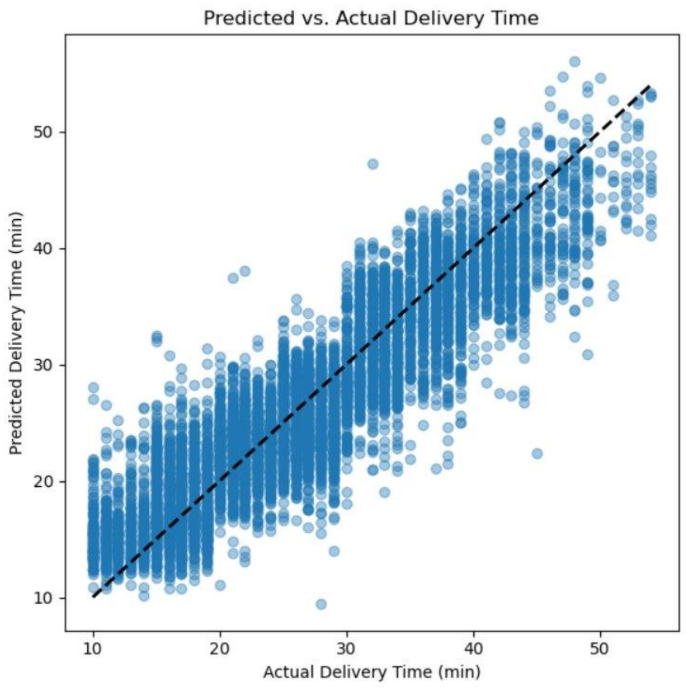
- **SVR:** RandomizedSearchCV (3-fold CV, 10 trials) over
  - $C \in [1e\text{-}2 \ldots 1e3]$,
  - $\gamma \in [1e\text{-}4 \ldots 1]$,
  - $\varepsilon \in \{0.1, 0.2, 0.3\}$

- **MLP:** HalvingRandomSearchCV (2-fold CV, successive-halving on max_iter up to 50 epochs) over
  - hidden_layer_sizes ∈ {(64,32), (128,64)},
  - $\alpha \in [1e\text{-}4 \ldots 1e\text{-}2]$,
  - learning_rate_init ∈ [1e-4 … 1e-2],
  - tol ∈ {1e-3}, batch_size=32

- **Final Retrain:** Best MLP retrained for 200 epochs on full training set

Performance on Hold-Out Test Set

- **SVR alone:** $R^2 \approx 0.758$
- **MLP alone:** $R^2 \approx 0.777$
- **Stacked Ensemble:**
  - Adjusted $R^2 \approx 0.803$
  - RMSE ≈ 4.12 min
  - MAE ≈ 3.26 min



Predicted vs. Actual Delivery Time for the SVR + MLP ensemble shows most points tightly clustered along the 1:1 line, confirming over 80% variance explained.

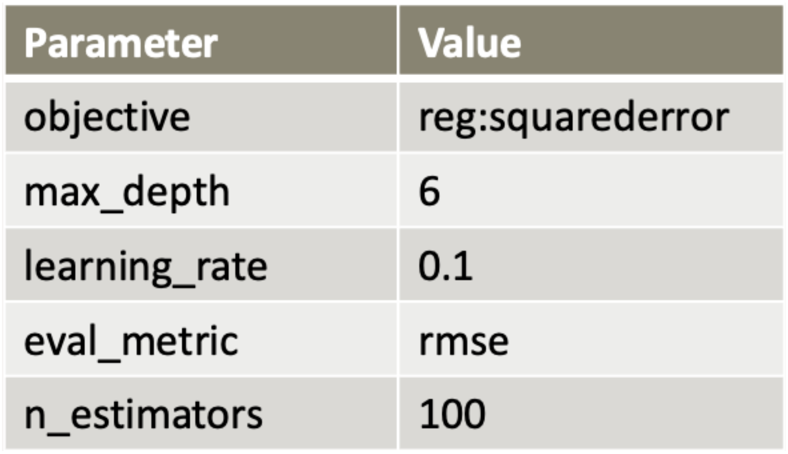
XGBoost

Extreme Gradient Boosting (XGBoost) is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It is highly efficient, flexible, and portable.It improves model performance by sequentially building trees that correct the errors of previous ones and introduces regularization (L1 and L2), making the model less prone to overfitting.

Key Features:

- Handles missing values automatically
- Supports regularization, improving generalization
- Highly efficient and can leverage parallel computation
- Provides feature importance scores, allowing better interpretability

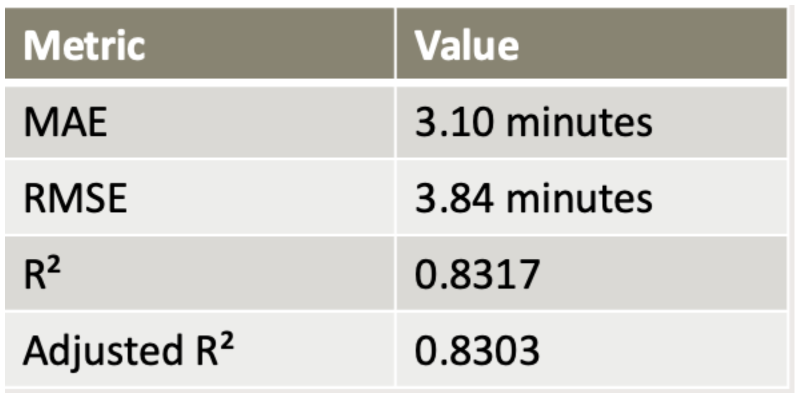
Hyperparameters:



Parameter	Value
objective	reg:squarederror
max_depth	6
learning_rate	0.1
eval_metric	rmse
n_estimators	100

Description of Hyperparameters:
objective = reg:squarederror
Specifies that the model's goal is regression, minimizing the squared error between predicted and actual values.
max_depth = 6
Limits the maximum depth of each decision tree to control model complexity and avoid overfitting.
learning_rate = 0.1
Also called eta, it controls how much each tree contributes to the final prediction. A lower learning rate typically requires more trees.
eval_metric = rmse
The evaluation metric used is Root Mean Squared Error (RMSE), which measures the average magnitude of the prediction errors.
n_estimators = 100
The number of boosting rounds or trees to be built sequentially.

Performance Metrics:



Metric	Value
MAE	3.10 minutes
RMSE	3.84 minutes
R^2	0.8317
Adjusted R^2	0.8303

Description of Metrics:

Mean Absolute Error (MAE):

The average absolute difference between the predicted and actual delivery times. In our case, predictions were off by about **3.10 minutes** on average.

Root Mean Squared Error (RMSE):

The square root of the average squared differences between predicted and actual values. The RMSE was **3.84 minutes**, slightly higher than MAE due to heavier penalization of larger errors.

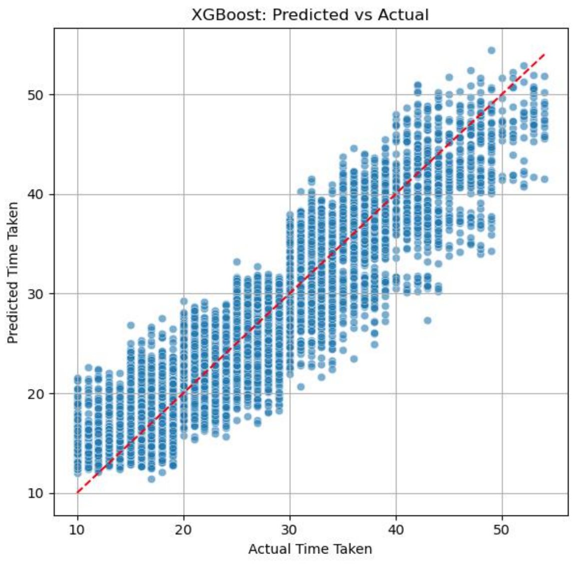
R^2 Score (Coefficient of Determination):

Measures the proportion of variance in the delivery time that is predictable from the features. Our XGBoost model achieved a high **R^2 value of 0.8317**, meaning it explains **83.17%** of the variability in delivery time.

Adjusted R^2 Score:

Adjusts the R^2 value based on the number of features used, penalizing for unnecessary complexity. Our adjusted R^2 was **0.8303**, very close to R^2, indicating the model is neither overfitted nor unnecessarily complex.

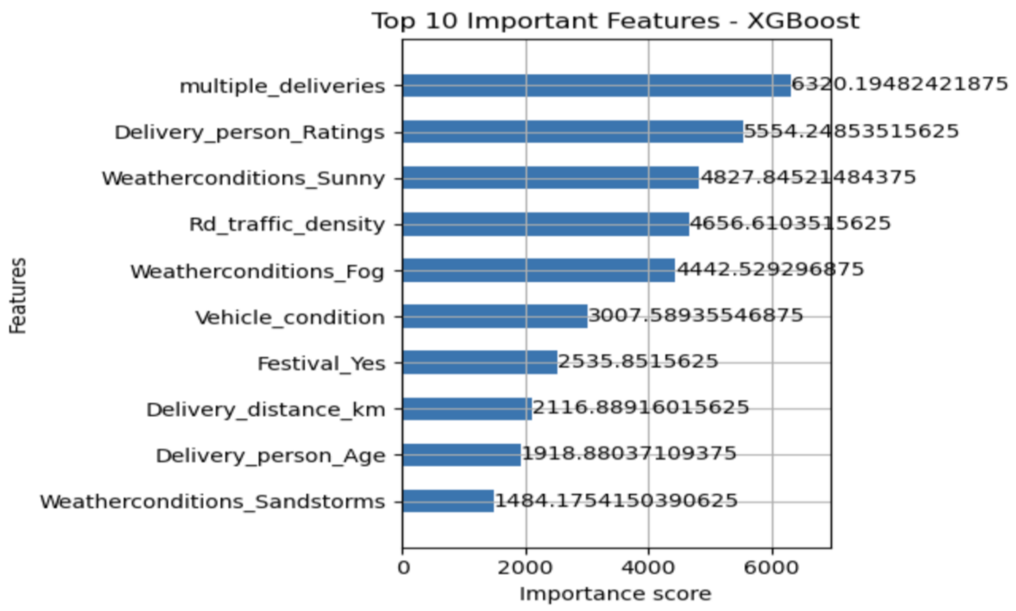
XGBoost: Predicted vs Actual Delivery Time



This scatter plot compares the **predicted delivery times** generated by the XGBoost model to the **actual delivery times** from the test dataset.

- The red dashed line represents the ideal case where the predicted time exactly matches the actual time (i.e., Predicted = Actual)
- The majority of the points are closely clustered around the red line, indicating that the XGBoost model achieves high prediction accuracy
- Some slight deviations can be observed, particularly at extreme values, but overall, the model generalizes well across the full delivery time range
- This visual confirmation supports the strong R² score (0.8317) and low error metrics obtained during model evaluation

Top 10 Influential Features Identified by XGBoost



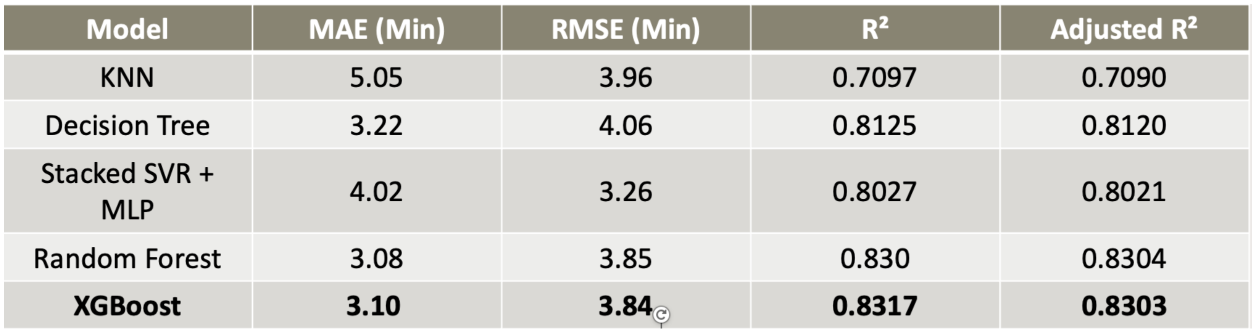
The bar chart displays the top 10 most influential features contributing to the prediction of delivery time as determined by the XGBoost model.

- Multiple Deliveries, Delivery Person Ratings, and Sunny Weather Conditions were the most important factors impacting delivery time
- Features such as Traffic Density, Foggy Weather, and Vehicle Condition also played a significant role in influencing delivery duration
- Other notable factors included whether the order was placed during a festival, the delivery distance, the delivery person's age, and the occurrence of sandstorms
- The importance scores were calculated based on how frequently and effectively each feature was used for splitting decision nodes in the trees

Key Insights:

- Assigning multiple deliveries to a single driver increases delivery times significantly
- Higher-rated delivery personnel are generally faster and more efficient
- Sunny weather conditions correlate with faster deliveries, while foggy and sandstorm conditions slow down deliveries
- Traffic congestion and festivals also contribute to delays

Model Performance Comparison



Model	MAE (Min)	RMSE (Min)	R²	Adjusted R²
KNN	5.05	3.96	0.7097	0.7090
Decision Tree	3.22	4.06	0.8125	0.8120
Stacked SVR + MLP	4.02	3.26	0.8027	0.8021
Random Forest	3.08	3.85	0.830	0.8304
XGBoost	**3.10**	**3.84**	**0.8317**	**0.8303**

The table above compares the performance of different machine learning models based on four evaluation metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), R^2 Score, and Adjusted R^2 Score.

- XGBoost emerged as the best overall model among all models tested
- It achieved the highest R^2 score (0.8317), meaning it explained the most variance in delivery time predictions
- Although its MAE (3.10 minutes) and RMSE (3.84 minutes) are very close to Random Forest, XGBoost generalized slightly better based on R^2 values
- The Adjusted R^2 score (0.8303) is also comparable to Random Forest (0.8304), indicating strong consistency even after accounting for the number of features used

Key Observations

- KNN had the lowest performance with the lowest R^2 score (0.7097) and the highest MAE

- Decision Tree and Stacked SVR + MLP performed better than KNN but still lagged behind Random Forest and XGBoost
- Random Forest showed excellent performance but was slightly outperformed by XGBoost in terms of R^2

Generated Insights

The insights below were derived by analyzing the feature importance and model behavior during the delivery time prediction:

- **Multiple Deliveries**:

  Handling multiple orders at once significantly increases total delivery time.

- **Delivery Person Ratings**:

  Higher-rated delivery personnel tend to complete deliveries faster, likely due to greater experience and efficiency.

- **Weather Conditions**:

  Sunny weather is associated with quicker deliveries, while adverse weather conditions such as fog and sandstorms cause delays.

- **Traffic Density**:

  Higher road traffic density leads to slower deliveries. Traffic congestion emerged as a major external factor impacting ETA.

- **Festival Periods**:

  Deliveries during festivals tend to take longer due to roadblocks, detours, and spikes in demand.

- **Vehicle Condition**:

  Better vehicle conditions correlate with faster deliveries, suggesting the importance of vehicle maintenance for operational efficiency.

- **Distance and Preparation Time**:

  Longer restaurant-to-customer distances and longer food preparation times naturally increase total delivery duration.

Recommendations for Improving Delivery Efficiency

Based on the generated insights and the model's analysis, the following actionable recommendations are proposed to optimize delivery times and improve customer satisfaction:

- **Avoid assigning too many orders to a single delivery partner**:

  Limiting the number of simultaneous deliveries helps reduce delays and improves overall efficiency.

- **Prefer experienced or highly rated delivery partners during peak hours**:

  Drivers with better ratings tend to complete deliveries faster and handle challenging conditions more effectively.

- **Use traffic and weather forecasts to adjust delivery time estimates**:

  Integrating real-time traffic and weather data into delivery planning can lead to more accurate ETA predictions and fewer customer complaints.

- **During festivals, increase delivery buffer time or allocate additional resources**:

  Anticipating delays during high-demand periods like festivals allows companies to manage expectations and maintain service quality.

- **Consider vehicle type and condition when assigning long-distance deliveries**:

  Assigning well-maintained, suitable vehicles for longer routes ensures timely deliveries and reduces operational risks.

By implementing these recommendations, food delivery platforms can achieve faster, more reliable deliveries, leading to improved customer experience, better resource utilization, and higher operational efficiency.