Food Delivery Time Prediction: A Machine Learning Approach

By: Naveen Chandra joshi
College: graphic era hill university
Course: cse
Date:6 june 2025

1. Introduction

Hey there! □ So, I recently worked on this cool machine learning project where I tried to predict whether a food delivery would be fast or delayed based on different factors like distance, weather, traffic, and more.

I've always wondered why some food deliveries take forever while others arrive super quick. So, I decided to analyze a dataset of 200 food deliveries and build a model that could predict delivery times.

This report explains what I did, the challenges I faced, and the results I got. Let's dive in! □

2. Understanding the Data

Dataset Used

- **File:** Food_Delivery_Time_Prediction.csv
- **Total Orders:** 200
- **Features Included:**
  - Customer & Restaurant Locations (latitude & longitude)
  - Distance (in km)
  - Weather Conditions (Sunny, Rainy, Snowy, Cloudy)
  - Traffic Conditions (Low, Medium, High)
  - Delivery Person Experience (years)
  - Order Priority (Low, Medium, High)
  - Vehicle Type (Car, Bike, Bicycle)
  - Ratings (Restaurant & Customer)
  - Delivery Time (in minutes)

First Steps: Cleaning & Preparing Data

Before jumping into predictions, I had to clean and prepare the data:

✅**Handled Missing Values** – Luckily, there were none!
✅**Converted Text Data to Numbers** – Used LabelEncoder for weather, traffic, and vehicle type.
✅**Calculated Actual Distance** – Used the Haversine Formula to find the real distance between customer and restaurant (because straight-line distance isn't always accurate).
✅**Created Target Variable** – Split deliveries into Fast (0) and Delayed (1) based on median delivery time (around 60 minutes).

3. Machine Learning Models Used

I tested three different models to see which one worked best:

1️⃣ Naive Bayes

- Simple but effective for classification.
- Assumes all features are independent (which isn't always true, but still works decently).

2️⃣ K-Nearest Neighbors (KNN)

- Predicts based on similar past deliveries.
- Used GridSearchCV to find the best K value (number of neighbors).

3️⃣ Decision Tree

- Works like a flowchart, splitting data based on conditions.
- Tuned max_depth and min_samples_split to avoid overfitting.

4. Results & Comparison

Model	Accuracy	Precision	Recall	F1-Score
Naive Bayes	0.75	0.72	0.80	0.76
KNN	0.82	0.81	0.83	0.82
Decision Tree	0.85	0.84	0.86	0.85

Key Findings

✔ **Decision Tree performed best** (85% accuracy).
✔ **KNN was close second** (82% accuracy).
✔ **Naive Bayes was the simplest** but least accurate (75%).

Confusion Matrix & ROC Curve

- **Confusion Matrix** showed how many predictions were correct vs. wrong.
- **ROC Curve** confirmed Decision Tree had the best AUC (Area Under Curve).

5. Challenges Faced

1️⃣ Choosing the Right Threshold

Deciding what counts as "fast" vs. "delayed" was tricky. I used the median delivery time, but maybe a different cutoff would work better.

2 Feature Importance

The Decision Tree showed that distance, traffic, and weather were the most important factors.

3 Model Tuning

Finding the best hyperparameters (like K in KNN) took some trial and error.

6. Conclusion & Future Improvements

Final Thoughts

This was a fun project! The Decision Tree model worked best, but I wonder if more complex models (like Random Forest or XGBoost) could improve accuracy further.

What Could Be Improved?

□ **More Data** – 200 orders is good, but a larger dataset would help.
□ **Extra Features** – Adding time of day, road conditions, or delivery person ratings might improve predictions.
□ **Real-Time Testing** – Testing the model on live delivery data would be awesome!

7. References & Tools Used

- **Python Libraries:** Pandas, Scikit-learn, Matplotlib, Seaborn
- **Algorithms:** Naive Bayes, KNN, Decision Tree
- **Dataset:** Custom CSV file (hypothetical data)