1. Objective

The goal is to predict food delivery times and classify deliveries as **'Fast' or 'Delayed'** using Naive Bayes, K-Nearest Neighbors (KNN), and Decision Tree classifiers.

---

2. Dataset Overview

The dataset includes delivery-related features such as customer and restaurant location, weather and traffic conditions, delivery person experience, order priority, vehicle type, and more.

Target Variable:

- Binary classification (0 = Fast, 1 = Delayed) based on delivery time.

---

3. Data Preprocessing

- Missing values handled using **median imputation**
- Categorical features encoded with **LabelEncoder**
- **Haversine formula** used to calculate distance
- Delivery time converted into **binary target variable**
- Features **standardized using StandardScaler**

---

4. Naive Bayes Classifier

- **Model**: Gaussian Naive Bayes
- **Accuracy**: 1.00
- **Confusion Matrix**: [[10 0], [0 30]]
- ✔ Perfect predictions on both classes
- ⚠ May indicate **overfitting** or **data leakage**

---

5. K-Nearest Neighbors (KNN)

- **Best K**: 5
- **Accuracy**: 0.725
- **Confusion Matrix**: [[1 9], [2 28]]
- ❌ Poor prediction for 'Fast' deliveries (class 0)

- ✔ Performs better for 'Delayed' class
- ⚠ Affected by **class imbalance**

---

6. Decision Tree Classifier

- **Accuracy**: 1.00
- **Confusion Matrix**: [[10 0], [0 30]]
- ✔ Clear logic, fully interpretable
- ⚠ Needs pruning to prevent **overfitting**

---

7. Data Visualization

Included:

- Accuracy bar chart comparison
- Confusion matrix heatmaps
- ROC/AUC plots (optional in notebook)

---

8. Conclusion

- **Naive Bayes & Decision Tree** performed perfectly — but results must be validated
- **KNN** showed bias — struggled with class 0
- All models need **cross-validation** for real-world deployment

---

9. Tools & Libraries Used

- Python, Jupyter Notebook
- Libraries: Pandas, NumPy, Matplotlib, Seaborn, scikit-learn

10. Final Conclusion (As per Model Evaluation)

In this project, we trained three classification models to predict whether a food delivery would be fast or delayed.

- **Naive Bayes** and **Decision Tree** achieved **100% accuracy**, but caution is needed — such results can indicate overfitting or poor train-test splitting.

- **KNN**, while intuitive and flexible, performed weakly for fast deliveries, likely due to class imbalance.

📌 **Recommended Model**: **Decision Tree**
Because of its high accuracy and interpretability. But make sure to:

- Apply **pruning**
- Use **cross-validation**
- Test on **unseen data** before deployment.